

# **AM335x ARM® Cortex™-A8 Microprocessors (MPUs)**

## **Technical Reference Manual**



Literature Number: SPRUH73G  
October 2011–Revised November 2012

|   |            |
|---|------------|
| <b>Preface .....</b>  | <b>146</b> |
| <b>1    Introduction .....</b>  | <b>147</b> |
| 1.1    AM335x Family .....  | 147        |
| 1.1.1    Device Features .....  | 147        |
| 1.1.2    Device Identification .....  | 148        |
| 1.1.3    Feature Identification .....   | 148        |
| <b>2    Memory Map .....</b>  | <b>150</b> |
| 2.1    ARM Cortex-A8 Memory Map .....   | 150        |
| 2.2    ARM Cortex-M3 Memory Map .....   | 159        |
| <b>3    ARM MPU Subsystem .....</b>   | <b>161</b> |
| 3.1    ARM Cortex-A8 MPU Subsystem .....  | 162        |
| 3.1.1    Features .....   | 163        |
| 3.1.2    MPU Subsystem Integration .....  | 163        |
| 3.1.3    MPU Subsystem Clock and Reset Distribution .....                                       | 164        |
| 3.1.4    ARM Subchip .....  | 167        |
| 3.1.5    Interrupt Controller .....   | 168        |
| 3.1.6    Power Management .....   | 168        |
| 3.1.7    ARM Programming Model .....  | 171        |
| <b>4    Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) .....</b> | <b>173</b> |
| 4.1    Introduction .....   | 174        |
| <b>5    Graphics Accelerator (SGX) .....</b>  | <b>175</b> |
| 5.1    Introduction .....   | 176        |
| 5.1.1    POWERVR SGX Main Features .....  | 176        |
| 5.1.2    SGX 3D Features .....  | 176        |
| 5.1.3    Universal Scalable Shader Engine (USSE) – Key Features .....                           | 177        |
| 5.1.4    Unsupported Features .....   | 178        |
| 5.2    Integration .....  | 179        |
| 5.2.1    SGX530 Connectivity Attributes .....   | 179        |
| 5.2.2    SGX530 Clock and Reset Management .....  | 179        |
| 5.2.3    SGX530 Pin List .....  | 180        |
| 5.3    Functional Description .....   | 181        |
| 5.3.1    SGX Block Diagram .....  | 181        |
| 5.3.2    SGX Elements Description .....   | 181        |
| <b>6    Interrupts .....</b>  | <b>183</b> |
| 6.1    Functional Description .....   | 184        |
| 6.1.1    Interrupt Processing .....   | 185        |
| 6.1.2    Register Protection .....  | 186        |
| 6.1.3    Module Power Saving .....  | 186        |
| 6.1.4    Error Handling .....   | 186        |
| 6.1.5    Interrupt Handling .....   | 186        |
| 6.2    Basic Programming Model .....  | 187        |
| 6.2.1    Initialization Sequence .....  | 187        |
| 6.2.2    INTC Processing Sequence .....   | 187        |
| 6.2.3    INTC Preemptive Processing Sequence .....  | 191        |

|          |  |            |
|----------|--|------------|
| 6.2.4    | Interrupt Preemption .....                                 | 195        |
| 6.2.5    | ARM A8 INTC Spurious Interrupt Handling .....              | 195        |
| 6.3      | ARM Cortex-A8 Interrupts .....                             | 196        |
| 6.4      | ARM Cortex-M3 Interrupts .....                             | 200        |
| 6.5      | PWM Events .....   | 202        |
| 6.6      | Interrupt Controller Registers .....                       | 203        |
| 6.6.1    | INTC Registers .....                                       | 203        |
| <b>7</b> | <b>Memory Subsystem .....</b>                              | <b>249</b> |
| 7.1      | GPMC .....   | 250        |
| 7.1.1    | Introduction .....   | 250        |
| 7.1.2    | Integration .....  | 253        |
| 7.1.3    | Functional Description .....                               | 255        |
| 7.1.4    | Use Cases .....  | 354        |
| 7.1.5    | Registers .....  | 365        |
| 7.2      | OCMC-RAM .....   | 397        |
| 7.2.1    | Introduction .....   | 397        |
| 7.2.2    | Integration .....  | 398        |
| 7.3      | EMIF .....   | 399        |
| 7.3.1    | Introduction .....   | 399        |
| 7.3.2    | Integration .....  | 401        |
| 7.3.3    | Functional Description .....                               | 403        |
| 7.3.4    | Use Cases .....  | 421        |
| 7.3.5    | EMIF4D Registers .....                                     | 421        |
| 7.3.6    | DDR2/3/mDDR PHY Registers .....                            | 466        |
| 7.4      | ELM .....  | 475        |
| 7.4.1    | Introduction .....   | 475        |
| 7.4.2    | Integration .....  | 476        |
| 7.4.3    | Functional Description .....                               | 477        |
| 7.4.4    | Basic Programming Model .....                              | 480        |
| 7.4.5    | ELM Registers .....  | 486        |
| <b>8</b> | <b>Power, Reset, and Clock Management (PRCM) .....</b>     | <b>498</b> |
| 8.1      | Power, Reset, and Clock Management .....                   | 499        |
| 8.1.1    | Introduction .....   | 499        |
| 8.1.2    | Device Power-Management Architecture Building Blocks ..... | 499        |
| 8.1.3    | Clock Management .....                                     | 499        |
| 8.1.4    | Power Management .....                                     | 505        |
| 8.1.5    | PRCM Module Overview .....                                 | 513        |
| 8.1.6    | Clock Generation and Management .....                      | 514        |
| 8.1.7    | Reset Management .....                                     | 531        |
| 8.1.8    | Power-Up/Down Sequence .....                               | 540        |
| 8.1.9    | IO State .....   | 540        |
| 8.1.10   | Voltage and Power Domains .....                            | 540        |
| 8.1.11   | Device Modules and Power Management Attributes List .....  | 541        |
| 8.1.12   | Clock Module Registers .....                               | 544        |
| 8.1.13   | Power Management Registers .....                           | 701        |
| <b>9</b> | <b>Control Module .....</b>                                | <b>742</b> |
| 9.1      | Introduction .....   | 743        |
| 9.2      | Functional Description .....                               | 743        |
| 9.2.1    | Control Module Initialization .....                        | 743        |
| 9.2.2    | Pad Control Registers .....                                | 743        |
| 9.2.3    | EDMA Event Multiplexing .....                              | 744        |
| 9.2.4    | Device Control and Status .....                            | 745        |
| 9.2.5    | DDR PHY .....  | 752        |

|        |   |     |
|--------|---|-----|
| 9.3    | CONTROL_MODULE Registers .....  | 753 |
| 9.3.1  | control_revision Register (offset = 0h) [reset = 0h] .....            | 759 |
| 9.3.2  | device_id Register (offset = 600h) [reset = 0x] .....                 | 760 |
| 9.3.3  | control_hwinfo Register (offset = 4h) [reset = 0h] .....              | 761 |
| 9.3.4  | control_sysconfig Register (offset = 10h) [reset = 0h] .....          | 762 |
| 9.3.5  | control_status Register (offset = 40h) [reset = 0h] .....             | 763 |
| 9.3.6  | control_emif_sdram_config Register (offset = 110h) [reset = 0h] ..... | 764 |
| 9.3.7  | core_sldo_ctrl Register (offset = 428h) [reset = 0h] .....            | 766 |
| 9.3.8  | mpu_sldo_ctrl Register (offset = 42Ch) [reset = 0h] .....             | 767 |
| 9.3.9  | clk32kdivratio_ctrl Register (offset = 444h) [reset = 0h] .....       | 768 |
| 9.3.10 | bandgap_ctrl Register (offset = 448h) [reset = 0h] .....              | 769 |
| 9.3.11 | bandgap_trim Register (offset = 44Ch) [reset = 0h] .....              | 770 |
| 9.3.12 | pll_clkinpulow_ctrl Register (offset = 458h) [reset = 0h] .....       | 771 |
| 9.3.13 | mosc_ctrl Register (offset = 468h) [reset = 0h] .....                 | 772 |
| 9.3.14 | rcosc_ctrl Register (offset = 46Ch) [reset = 0h] .....                | 773 |
| 9.3.15 | deepsleep_ctrl Register (offset = 470h) [reset = 0h] .....            | 774 |
| 9.3.16 | dev_feature Register (offset = 604h) [reset = 0h] .....               | 775 |
| 9.3.17 | init_priority_0 Register (offset = 608h) [reset = 0h] .....           | 776 |
| 9.3.18 | init_priority_1 Register (offset = 60Ch) [reset = 0h] .....           | 777 |
| 9.3.19 | mmu_cfg Register (offset = 610h) [reset = 0h] .....                   | 778 |
| 9.3.20 | tptc_cfg Register (offset = 614h) [reset = 0h] .....                  | 779 |
| 9.3.21 | usb_ctrl0 Register (offset = 620h) [reset = 0h] .....                 | 780 |
| 9.3.22 | usb_sts0 Register (offset = 624h) [reset = 0h] .....                  | 782 |
| 9.3.23 | usb_ctrl1 Register (offset = 628h) [reset = 0h] .....                 | 783 |
| 9.3.24 | usb_sts1 Register (offset = 62Ch) [reset = 0h] .....                  | 785 |
| 9.3.25 | mac_id0_lo Register (offset = 630h) [reset = 0h] .....                | 786 |
| 9.3.26 | mac_id0_hi Register (offset = 634h) [reset = 0h] .....                | 787 |
| 9.3.27 | mac_id1_lo Register (offset = 638h) [reset = 0h] .....                | 788 |
| 9.3.28 | mac_id1_hi Register (offset = 63Ch) [reset = 0h] .....                | 789 |
| 9.3.29 | dcan_raminit Register (offset = 644h) [reset = 0h] .....              | 790 |
| 9.3.30 | usb_wkup_ctrl Register (offset = 648h) [reset = 0h] .....             | 791 |
| 9.3.31 | gmii_sel Register (offset = 650h) [reset = 0h] .....                  | 792 |
| 9.3.32 | pwmss_ctrl Register (offset = 664h) [reset = 0h] .....                | 793 |
| 9.3.33 | mreqprio_0 Register (offset = 670h) [reset = 0h] .....                | 794 |
| 9.3.34 | mreqprio_1 Register (offset = 674h) [reset = 0h] .....                | 795 |
| 9.3.35 | hw_event_sel_grp1 Register (offset = 690h) [reset = 0h] .....         | 796 |
| 9.3.36 | hw_event_sel_grp2 Register (offset = 694h) [reset = 0h] .....         | 797 |
| 9.3.37 | hw_event_sel_grp3 Register (offset = 698h) [reset = 0h] .....         | 798 |
| 9.3.38 | hw_event_sel_grp4 Register (offset = 69Ch) [reset = 0h] .....         | 799 |
| 9.3.39 | smrt_ctrl Register (offset = 6A0h) [reset = 0h] .....                 | 800 |
| 9.3.40 | mpuss_hw_debug_sel Register (offset = 6A4h) [reset = 0h] .....        | 801 |
| 9.3.41 | mpuss_hw_dbg_info Register (offset = 6A8h) [reset = 0h] .....         | 802 |
| 9.3.42 | vdd_mpu_opp_050 Register (offset = 770h) [reset = 0h] .....           | 803 |
| 9.3.43 | vdd_mpu_opp_100 Register (offset = 774h) [reset = 0h] .....           | 804 |
| 9.3.44 | vdd_mpu_opp_120 Register (offset = 778h) [reset = 0h] .....           | 805 |
| 9.3.45 | vdd_mpu_opp_turbo Register (offset = 77Ch) [reset = 0h] .....         | 806 |
| 9.3.46 | vdd_core_opp_050 Register (offset = 7B8h) [reset = 0h] .....          | 807 |
| 9.3.47 | vdd_core_opp_100 Register (offset = 7BCh) [reset = 0h] .....          | 808 |
| 9.3.48 | bb_scale Register (offset = 7D0h) [reset = 0h] .....                  | 809 |
| 9.3.49 | usb_vid_pid Register (offset = 7F4h) [reset = 4516141h] .....         | 810 |
| 9.3.50 | conf_<module>_<pin> Register (offset = 800h–A34h) .....               | 811 |
| 9.3.51 | cqdetect_status Register (offset = E00h) [reset = 0h] .....           | 812 |
| 9.3.52 | ddr_io_ctrl Register (offset = E04h) [reset = 0h] .....               | 813 |

|           |  |            |
|-----------|--|------------|
| 9.3.53    | vtp_ctrl Register (offset = E0Ch) [reset = 0h] .....           | 814        |
| 9.3.54    | vref_ctrl Register (offset = E14h) [reset = 0h] .....          | 815        |
| 9.3.55    | tpcc_evt_mux_0_3 Register (offset = F90h) [reset = 0h] .....   | 816        |
| 9.3.56    | tpcc_evt_mux_4_7 Register (offset = F94h) [reset = 0h] .....   | 817        |
| 9.3.57    | tpcc_evt_mux_8_11 Register (offset = F98h) [reset = 0h] .....  | 818        |
| 9.3.58    | tpcc_evt_mux_12_15 Register (offset = F9Ch) [reset = 0h] ..... | 819        |
| 9.3.59    | tpcc_evt_mux_16_19 Register (offset = FA0h) [reset = 0h] ..... | 820        |
| 9.3.60    | tpcc_evt_mux_20_23 Register (offset = FA4h) [reset = 0h] ..... | 821        |
| 9.3.61    | tpcc_evt_mux_24_27 Register (offset = FA8h) [reset = 0h] ..... | 822        |
| 9.3.62    | tpcc_evt_mux_28_31 Register (offset = FACH) [reset = 0h] ..... | 823        |
| 9.3.63    | tpcc_evt_mux_32_35 Register (offset = FB0h) [reset = 0h] ..... | 824        |
| 9.3.64    | tpcc_evt_mux_36_39 Register (offset = FB4h) [reset = 0h] ..... | 825        |
| 9.3.65    | tpcc_evt_mux_40_43 Register (offset = FB8h) [reset = 0h] ..... | 826        |
| 9.3.66    | tpcc_evt_mux_44_47 Register (offset = FBCh) [reset = 0h] ..... | 827        |
| 9.3.67    | tpcc_evt_mux_48_51 Register (offset = FC0h) [reset = 0h] ..... | 828        |
| 9.3.68    | tpcc_evt_mux_52_55 Register (offset = FC4h) [reset = 0h] ..... | 829        |
| 9.3.69    | tpcc_evt_mux_56_59 Register (offset = FC8h) [reset = 0h] ..... | 830        |
| 9.3.70    | tpcc_evt_mux_60_63 Register (offset = FCCh) [reset = 0h] ..... | 831        |
| 9.3.71    | timer_evt_capt Register (offset = FD0h) [reset = 0h] .....     | 832        |
| 9.3.72    | ecap_evt_capt Register (offset = FD4h) [reset = 0h] .....      | 833        |
| 9.3.73    | adc_evt_capt Register (offset = FD8h) [reset = 0h] .....       | 834        |
| 9.3.74    | reset_iso Register (offset = 1000h) [reset = 0h] .....         | 835        |
| 9.3.75    | ddr_cke_ctrl Register (offset = 131Ch) [reset = 0h] .....      | 836        |
| 9.3.76    | sma2 Register (offset = 1320h) [reset = 0h] .....              | 837        |
| 9.3.77    | m3_txev_eoi Register (offset = 1324h) [reset = 0h] .....       | 838        |
| 9.3.78    | ipc_msg_reg0 Register (offset = 1328h) [reset = 0h] .....      | 839        |
| 9.3.79    | ipc_msg_reg1 Register (offset = 132Ch) [reset = 0h] .....      | 840        |
| 9.3.80    | ipc_msg_reg2 Register (offset = 1330h) [reset = 0h] .....      | 841        |
| 9.3.81    | ipc_msg_reg3 Register (offset = 1334h) [reset = 0h] .....      | 842        |
| 9.3.82    | ipc_msg_reg4 Register (offset = 1338h) [reset = 0h] .....      | 843        |
| 9.3.83    | ipc_msg_reg5 Register (offset = 133Ch) [reset = 0h] .....      | 844        |
| 9.3.84    | ipc_msg_reg6 Register (offset = 1340h) [reset = 0h] .....      | 845        |
| 9.3.85    | ipc_msg_reg7 Register (offset = 1344h) [reset = 0h] .....      | 846        |
| 9.3.86    | ddr_cmd0_ioctrl Register (offset = 1404h) [reset = 0h] .....   | 847        |
| 9.3.87    | ddr_cmd1_ioctrl Register (offset = 1408h) [reset = 0h] .....   | 849        |
| 9.3.88    | ddr_cmd2_ioctrl Register (offset = 140Ch) [reset = 0h] .....   | 851        |
| 9.3.89    | ddr_data0_ioctrl Register (offset = 1440h) [reset = 0h] .....  | 853        |
| 9.3.90    | ddr_data1_ioctrl Register (offset = 1444h) [reset = 0h] .....  | 855        |
| <b>10</b> | <b>Interconnects .....</b>                                     | <b>857</b> |
| 10.1      | Introduction .....   | 858        |
| 10.1.1    | Terminology .....  | 858        |
| 10.1.2    | L3 Interconnect .....  | 858        |
| 10.1.3    | L4 Interconnect .....  | 861        |
| <b>11</b> | <b>Enhanced Direct Memory Access (EDMA) .....</b>              | <b>863</b> |
| 11.1      | Introduction .....   | 864        |
| 11.1.1    | EDMA3 Controller Block Diagram .....                           | 864        |
| 11.1.2    | Third-Party Channel Controller (TPCC) Overview .....           | 864        |
| 11.1.3    | Third-Party Transfer Controller (PTPC) Overview .....          | 865        |
| 11.2      | Integration .....  | 867        |
| 11.2.1    | Third-Party Channel Controller (TPCC) Integration .....        | 867        |
| 11.2.2    | Third-Party Transfer Controller (PTPC) Integration .....       | 868        |
| 11.3      | Functional Description .....                                   | 870        |
| 11.3.1    | Functional Overview .....                                      | 870        |

|           |   |             |
|-----------|---|-------------|
| 11.3.2    | Types of EDMA3 Transfers .....                        | 873         |
| 11.3.3    | Parameter RAM (PaRAM) .....                           | 875         |
| 11.3.4    | Initiating a DMA Transfer .....                       | 887         |
| 11.3.5    | Completion of a DMA Transfer .....                    | 890         |
| 11.3.6    | Event, Channel, and PaRAM Mapping .....               | 891         |
| 11.3.7    | EDMA3 Channel Controller Regions .....                | 893         |
| 11.3.8    | Chaining EDMA3 Channels .....                         | 895         |
| 11.3.9    | EDMA3 Interrupts .....                                | 896         |
| 11.3.10   | Memory Protection .....                               | 902         |
| 11.3.11   | Event Queue(s) .....                                  | 906         |
| 11.3.12   | EDMA3 Transfer Controller (EDMA3TC) .....             | 908         |
| 11.3.13   | Event Dataflow .....                                  | 911         |
| 11.3.14   | EDMA3 Prioritization .....                            | 911         |
| 11.3.15   | EDMA3 Operating Frequency (Clock Control) .....       | 912         |
| 11.3.16   | Reset Considerations .....                            | 912         |
| 11.3.17   | Power Management .....                                | 912         |
| 11.3.18   | Emulation Considerations .....                        | 912         |
| 11.3.19   | EDMA Transfer Examples .....                          | 914         |
| 11.3.20   | EDMA Events .....                                     | 930         |
| 11.4      | EDMA3 Registers .....                                 | 933         |
| 11.4.1    | EDMA3 Channel Controller Registers .....              | 933         |
| 11.4.2    | EDMA3 Transfer Controller Registers .....             | 987         |
| 11.5      | Appendix A .....                                      | 1011        |
| 11.5.1    | Debug Checklist .....                                 | 1011        |
| 11.5.2    | Miscellaneous Programming/Debug Tips .....            | 1012        |
| 11.5.3    | Setting Up a Transfer .....                           | 1013        |
| <b>12</b> | <b>Touchscreen Controller .....</b>                   | <b>1015</b> |
| 12.1      | Introduction .....                                    | 1016        |
| 12.1.1    | TSC_ADC Features .....                                | 1016        |
| 12.1.2    | Unsupported TSC_ADC_SS Features .....                 | 1016        |
| 12.2      | Integration .....                                     | 1017        |
| 12.2.1    | TSC_ADC Connectivity Attributes .....                 | 1017        |
| 12.2.2    | TSC_ADC Clock and Reset Management .....              | 1018        |
| 12.2.3    | TSC_ADC Pin List .....                                | 1018        |
| 12.3      | Functional Description .....                          | 1019        |
| 12.3.1    | HW Synchronized or SW Channels .....                  | 1019        |
| 12.3.2    | Open Delay and Sample Delay .....                     | 1019        |
| 12.3.3    | Averaging of Samples (1, 2, 4, 8, and 16) .....       | 1019        |
| 12.3.4    | One-Shot (Single) or Continuous Mode .....            | 1019        |
| 12.3.5    | Interrupts .....                                      | 1019        |
| 12.3.6    | DMA Requests .....                                    | 1020        |
| 12.3.7    | Analog Front End (AFE) Functional Block Diagram ..... | 1020        |
| 12.4      | Operational Modes .....                               | 1021        |
| 12.4.1    | PenCtrl and PenIRQ .....                              | 1022        |
| 12.5      | Touchscreen Controller Registers .....                | 1025        |
| 12.5.1    | TSC_ADC_SS Registers .....                            | 1025        |
| <b>13</b> | <b>LCD Controller .....</b>                           | <b>1089</b> |
| 13.1      | Introduction .....                                    | 1090        |
| 13.1.1    | Purpose of the Peripheral .....                       | 1090        |
| 13.1.2    | Features .....  | 1091        |
| 13.2      | Integration .....                                     | 1092        |
| 13.2.1    | LCD Controller Connectivity Attributes .....          | 1092        |
| 13.2.2    | LCD Controller Clock and Reset Management .....       | 1093        |

|           |   |             |
|-----------|---|-------------|
| 13.2.3    | LCD Controller Pin List .....                                 | 1093        |
| 13.3      | Functional Description .....                                  | 1094        |
| 13.3.1    | Clocking .....  | 1094        |
| 13.3.2    | LCD External I/O Signals .....                                | 1096        |
| 13.3.3    | DMA Engine .....  | 1097        |
| 13.3.4    | LIDD Controller .....   | 1098        |
| 13.3.5    | Raster Controller .....                                       | 1100        |
| 13.3.6    | Interrupt Conditions .....                                    | 1111        |
| 13.3.7    | DMA .....   | 1113        |
| 13.3.8    | Power Management .....  | 1113        |
| 13.4      | Programming Model .....                                       | 1114        |
| 13.4.1    | LCD Character Displays .....                                  | 1114        |
| 13.4.2    | Active Matrix Displays .....                                  | 1117        |
| 13.4.3    | System Interaction .....                                      | 1117        |
| 13.4.4    | Palette Lookup .....  | 1117        |
| 13.4.5    | Test Logic .....  | 1119        |
| 13.4.6    | Disable and Software Reset Sequence .....                     | 1119        |
| 13.4.7    | Precedence Order for Determining Frame Buffer Type .....      | 1120        |
| 13.5      | LCD Registers .....   | 1120        |
| 13.5.1    | PID Register (offset = 0h) [reset = 0h] .....                 | 1122        |
| 13.5.2    | CTRL Register (offset = 4h) [reset = 0h] .....                | 1123        |
| 13.5.3    | LIDD_CTRL Register (offset = Ch) [reset = 0h] .....           | 1124        |
| 13.5.4    | LIDD_CS0_CONF Register (offset = 10h) [reset = 0h] .....      | 1125        |
| 13.5.5    | LIDD_CS0_ADDR Register (offset = 14h) [reset = 0h] .....      | 1126        |
| 13.5.6    | LIDD_CS0_DATA Register (offset = 18h) [reset = 0h] .....      | 1127        |
| 13.5.7    | LIDD_CS1_CONF Register (offset = 1Ch) [reset = 0h] .....      | 1128        |
| 13.5.8    | LIDD_CS1_ADDR Register (offset = 20h) [reset = 0h] .....      | 1129        |
| 13.5.9    | LIDD_CS1_DATA Register (offset = 24h) [reset = 0h] .....      | 1130        |
| 13.5.10   | RASTER_CTRL Register (offset = 28h) [reset = 0h] .....        | 1131        |
| 13.5.11   | RASTER_TIMING_0 Register (offset = 2Ch) [reset = 0h] .....    | 1133        |
| 13.5.12   | RASTER_TIMING_1 Register (offset = 30h) [reset = 0h] .....    | 1134        |
| 13.5.13   | RASTER_TIMING_2 Register (offset = 34h) [reset = 0h] .....    | 1135        |
| 13.5.14   | RASTER_SUBPANEL Register (offset = 38h) [reset = 0h] .....    | 1137        |
| 13.5.15   | RASTER_SUBPANEL2 Register (offset = 3Ch) [reset = 0h] .....   | 1138        |
| 13.5.16   | LCDDMA_CTRL Register (offset = 40h) [reset = 0h] .....        | 1139        |
| 13.5.17   | LCDDMA_FB0_BASE Register (offset = 44h) [reset = 0h] .....    | 1140        |
| 13.5.18   | LCDDMA_FB0_CEILING Register (offset = 48h) [reset = 0h] ..... | 1141        |
| 13.5.19   | LCDDMA_FB1_BASE Register (offset = 4Ch) [reset = 0h] .....    | 1142        |
| 13.5.20   | LCDDMA_FB1_CEILING Register (offset = 50h) [reset = 0h] ..... | 1143        |
| 13.5.21   | SYSCONFIG Register (offset = 54h) [reset = 0h] .....          | 1144        |
| 13.5.22   | IRQSTATUS_RAW Register (offset = 58h) [reset = 0h] .....      | 1145        |
| 13.5.23   | IRQSTATUS Register (offset = 5Ch) [reset = 0h] .....          | 1147        |
| 13.5.24   | IRQENABLE_SET Register (offset = 60h) [reset = 0h] .....      | 1149        |
| 13.5.25   | IRQENABLE_CLEAR Register (offset = 64h) [reset = 0h] .....    | 1151        |
| 13.5.26   | CLKC_ENABLE Register (offset = 6Ch) [reset = 0h] .....        | 1153        |
| 13.5.27   | CLKC_RESET Register (offset = 70h) [reset = 0h] .....         | 1154        |
| <b>14</b> | <b>Ethernet Subsystem .....</b>                               | <b>1155</b> |
| 14.1      | Introduction .....  | 1156        |
| 14.1.1    | Features .....  | 1156        |
| 14.1.2    | Unsupported Features .....                                    | 1157        |
| 14.2      | Integration .....   | 1158        |
| 14.2.1    | Ethernet Switch Connectivity Attributes .....                 | 1158        |
| 14.2.2    | Ethernet Switch Clock and Reset Management .....              | 1160        |

|           |  |             |
|-----------|--|-------------|
| 14.2.3    | Ethernet Switch Pin List .....                           | 1161        |
| 14.2.4    | Ethernet Switch RMII Clocking Details .....              | 1161        |
| 14.2.5    | GMII Interface Signal Connections and Descriptions ..... | 1162        |
| 14.2.6    | RMII Signal Connections and Descriptions .....           | 1165        |
| 14.2.7    | RGMII Signal Connections and Descriptions .....          | 1166        |
| 14.3      | Functional Description .....                             | 1168        |
| 14.3.1    | CPSW_3G Subsystem .....                                  | 1168        |
| 14.3.2    | CPSW_3G .....  | 1173        |
| 14.3.3    | Ethernet Mac Sliver (CPGMAC_SL) .....                    | 1215        |
| 14.3.4    | Command IDLE .....                                       | 1217        |
| 14.3.5    | RMII Interface .....                                     | 1218        |
| 14.3.6    | RGMII Interface .....                                    | 1218        |
| 14.3.7    | Common Platform Time Sync (CPTS) .....                   | 1221        |
| 14.3.8    | MDIO .....   | 1226        |
| 14.4      | Software Operation .....                                 | 1228        |
| 14.4.1    | Transmit Operation .....                                 | 1228        |
| 14.4.2    | Receive Operation .....                                  | 1230        |
| 14.4.3    | Initializing the MDIO Module .....                       | 1231        |
| 14.4.4    | Writing Data to a PHY Register .....                     | 1231        |
| 14.4.5    | Reading Data from a PHY Register .....                   | 1232        |
| 14.4.6    | Initialization and Configuration of CPSW .....           | 1232        |
| 14.5      | Ethernet Subsystem Registers .....                       | 1233        |
| 14.5.1    | CPSW_ALE Registers .....                                 | 1233        |
| 14.5.2    | CPSW_CPDMA Registers .....                               | 1248        |
| 14.5.3    | CPSW_CPTS Registers .....                                | 1301        |
| 14.5.4    | CPSW_STATS Registers .....                               | 1314        |
| 14.5.5    | CPDMA_STATERAM Registers .....                           | 1314        |
| 14.5.6    | CPSW_PORT Registers .....                                | 1348        |
| 14.5.7    | CPSW_SL Registers .....                                  | 1403        |
| 14.5.8    | CPSW_SS Registers .....                                  | 1417        |
| 14.5.9    | CPSW_WR Registers .....                                  | 1430        |
| 14.5.10   | Management Data Input/Output (MDIO) Registers .....      | 1466        |
| <b>15</b> | <b>Pulse-Width Modulation Subsystem (PWMSS)</b> .....    | <b>1477</b> |
| 15.1      | Pulse-Width Modulation Subsystem (PWMSS) .....           | 1478        |
| 15.1.1    | Introduction .....                                       | 1478        |
| 15.1.2    | Integration .....  | 1480        |
| 15.1.3    | PWMSS Registers .....                                    | 1481        |
| 15.2      | Enhanced PWM (ePWM) Module .....                         | 1486        |
| 15.2.1    | Introduction .....                                       | 1486        |
| 15.2.2    | Functional Description .....                             | 1490        |
| 15.2.3    | Use Cases .....  | 1549        |
| 15.2.4    | Registers .....  | 1573        |
| 15.3      | Enhanced Capture (eCAP) Module .....                     | 1599        |
| 15.3.1    | Introduction .....                                       | 1599        |
| 15.3.2    | Functional Description .....                             | 1600        |
| 15.3.3    | Use Cases .....  | 1610        |
| 15.3.4    | Registers .....  | 1626        |
| 15.4      | Enhanced Quadrature Encoder Pulse (eQEP) Module .....    | 1638        |
| 15.4.1    | Introduction .....                                       | 1638        |
| 15.4.2    | Functional Description .....                             | 1641        |
| 15.4.3    | eQEP Registers .....                                     | 1659        |
| <b>16</b> | <b>Universal Serial Bus (USB)</b> .....                  | <b>1677</b> |
| 16.1      | Introduction .....                                       | 1678        |

|           |  |             |
|-----------|--|-------------|
| 16.1.1    | Acronyms, Abbreviations, and Definitions .....                 | 1678        |
| 16.1.2    | USB Features .....   | 1679        |
| 16.1.3    | Unsupported USB OTG and PHY Features .....                     | 1680        |
| 16.2      | Integration .....  | 1681        |
| 16.2.1    | USB Connectivity Attributes .....                              | 1681        |
| 16.2.2    | USB Clock and Reset Management .....                           | 1682        |
| 16.2.3    | USB Pin List .....   | 1682        |
| 16.2.4    | USB GPIO Details .....   | 1682        |
| 16.2.5    | USB Unbonded PHY Pads .....                                    | 1683        |
| 16.3      | Functional Description .....                                   | 1684        |
| 16.3.1    | VBUS Voltage Sourcing Control .....                            | 1684        |
| 16.3.2    | Pullup/PullDown Resistors .....                                | 1684        |
| 16.3.3    | Role Assuming Method .....                                     | 1685        |
| 16.3.4    | Clock, PLL, and PHY Initialization .....                       | 1685        |
| 16.3.5    | Indexed and Non-Indexed Register Spaces .....                  | 1685        |
| 16.3.6    | Dynamic FIFO Sizing .....                                      | 1685        |
| 16.3.7    | USB Controller Host and Peripheral Modes Operation .....       | 1686        |
| 16.3.8    | Protocol Description(s) .....                                  | 1688        |
| 16.3.9    | Communications Port Programming Interface (CPPI) 4.1 DMA ..... | 1721        |
| 16.3.10   | USB 2.0 Test Modes .....                                       | 1745        |
| 16.4      | Supported Use Cases .....                                      | 1746        |
| 16.5      | USB Registers .....  | 1747        |
| 16.5.1    | USBSS Registers .....  | 1747        |
| 16.5.2    | USB0_CTRL Registers .....                                      | 1790        |
| 16.5.3    | USB1_CTRL Registers .....                                      | 1840        |
| 16.5.4    | USB2PHY Registers .....  | 1888        |
| 16.5.5    | CPPI_DMA Registers .....                                       | 1912        |
| 16.5.6    | CPPI_DMA_SCHEDULER Registers .....                             | 2068        |
| 16.5.7    | QUEUE_MGR Registers .....                                      | 2071        |
| <b>17</b> | <b>Interprocessor Communication .....</b>                      | <b>3222</b> |
| 17.1      | Mailbox .....  | 3223        |
| 17.1.1    | Introduction .....   | 3223        |
| 17.1.2    | Integration .....  | 3224        |
| 17.1.3    | Functional Description .....                                   | 3225        |
| 17.1.4    | Programming Guide .....  | 3229        |
| 17.1.5    | MAILBOX Registers .....  | 3232        |
| 17.2      | Spinlock .....   | 3293        |
| 17.2.1    | SPINLOCK Registers .....                                       | 3293        |
| <b>18</b> | <b>Multimedia Card (MMC) .....</b>                             | <b>3331</b> |
| 18.1      | Introduction .....   | 3332        |
| 18.1.1    | MMCHS Features .....   | 3332        |
| 18.1.2    | Unsupported MMCHS Features .....                               | 3332        |
| 18.2      | Integration .....  | 3333        |
| 18.2.1    | MMCHS Connectivity Attributes .....                            | 3334        |
| 18.2.2    | MMCHS Clock and Reset Management .....                         | 3335        |
| 18.2.3    | MMCHS Pin List .....   | 3335        |
| 18.3      | Functional Description .....                                   | 3337        |
| 18.3.1    | MMC/SD/SDIO Functional Modes .....                             | 3337        |
| 18.3.2    | Resets .....   | 3344        |
| 18.3.3    | Power Management .....   | 3345        |
| 18.3.4    | Interrupt Requests .....                                       | 3348        |
| 18.3.5    | DMA Modes .....  | 3350        |
| 18.3.6    | Mode Selection .....   | 3353        |

|           |   |             |
|-----------|---|-------------|
| 18.3.7    | Buffer Management .....   | 3353        |
| 18.3.8    | Transfer Process .....  | 3356        |
| 18.3.9    | Transfer or Command Status and Error Reporting .....            | 3357        |
| 18.3.10   | Auto Command 12 Timings .....                                   | 3362        |
| 18.3.11   | Transfer Stop .....   | 3364        |
| 18.3.12   | Output Signals Generation .....                                 | 3365        |
| 18.3.13   | Card Boot Mode Management .....                                 | 3367        |
| 18.3.14   | CE-ATA Command Completion Disable Management .....              | 3369        |
| 18.3.15   | Test Registers .....  | 3369        |
| 18.3.16   | MMC/SD/SDIO Hardware Status Features .....                      | 3370        |
| 18.4      | Low-Level Programming Models .....                              | 3371        |
| 18.4.1    | Surrounding Modules Global Initialization .....                 | 3371        |
| 18.4.2    | MMC/SD/SDIO Controller Initialization Flow .....                | 3371        |
| 18.4.3    | Operational Modes Configuration .....                           | 3374        |
| 18.5      | Multimedia Card Registers .....                                 | 3376        |
| 18.5.1    | MULTIMEDIA_CARD Registers .....                                 | 3376        |
| <b>19</b> | <b>Universal Asynchronous Receiver/Transmitter (UART) .....</b> | <b>3433</b> |
| 19.1      | Introduction .....  | 3434        |
| 19.1.1    | UART Mode Features .....  | 3434        |
| 19.1.2    | IrDA Mode Features .....  | 3434        |
| 19.1.3    | CIR Mode Features .....   | 3434        |
| 19.1.4    | Unsupported UART Features .....                                 | 3434        |
| 19.2      | Integration .....   | 3436        |
| 19.2.1    | UART Connectivity Attributes .....                              | 3436        |
| 19.2.2    | UART Clock and Reset Management .....                           | 3437        |
| 19.2.3    | UART Pin List .....   | 3439        |
| 19.3      | Functional Description .....                                    | 3440        |
| 19.3.1    | Block Diagram .....   | 3440        |
| 19.3.2    | Clock Configuration .....                                       | 3441        |
| 19.3.3    | Software Reset .....  | 3441        |
| 19.3.4    | Power Management .....  | 3441        |
| 19.3.5    | Interrupt Requests .....  | 3443        |
| 19.3.6    | FIFO Management .....   | 3446        |
| 19.3.7    | Mode Selection .....  | 3454        |
| 19.3.8    | Protocol Formatting .....                                       | 3460        |
| 19.4      | UART/IrDA/CIR Basic Programming Model .....                     | 3483        |
| 19.4.1    | UART Programming Model .....                                    | 3483        |
| 19.4.2    | IrDA Programming Model .....                                    | 3489        |
| 19.5      | UART Registers .....  | 3492        |
| 19.5.1    | UART Registers .....  | 3492        |
| <b>20</b> | <b>Timers .....</b>   | <b>3537</b> |
| 20.1      | DMTimer .....   | 3538        |
| 20.1.1    | Introduction .....  | 3538        |
| 20.1.2    | Integration .....   | 3540        |
| 20.1.3    | Functional Description .....                                    | 3542        |
| 20.1.4    | Use Cases .....   | 3551        |
| 20.1.5    | TIMER Registers .....   | 3551        |
| 20.2      | DMTimer 1ms .....   | 3569        |
| 20.2.1    | Introduction .....  | 3569        |
| 20.2.2    | Integration .....   | 3571        |
| 20.2.3    | Functional Description .....                                    | 3573        |
| 20.2.4    | Use Cases .....   | 3581        |
| 20.2.5    | DMTIMER_1MS Registers .....                                     | 3581        |

|           |  |             |
|-----------|--|-------------|
| 20.3      | RTC_SS .....   | 3605        |
| 20.3.1    | Introduction .....                                       | 3605        |
| 20.3.2    | Integration .....  | 3606        |
| 20.3.3    | Functional Description .....                             | 3607        |
| 20.3.4    | Use Cases .....  | 3615        |
| 20.3.5    | RTC Registers .....                                      | 3615        |
| 20.4      | WATCHDOG .....   | 3653        |
| 20.4.1    | Introduction .....                                       | 3653        |
| 20.4.2    | Integration .....  | 3654        |
| 20.4.3    | Functional Description .....                             | 3655        |
| 20.4.4    | Watchdog Registers .....                                 | 3662        |
| <b>21</b> | <b>I2C .....</b>   | <b>3680</b> |
| 21.1      | Introduction .....                                       | 3681        |
| 21.1.1    | I2C Features .....                                       | 3681        |
| 21.1.2    | Unsupported I2C Features .....                           | 3681        |
| 21.2      | Integration .....  | 3682        |
| 21.2.1    | I2C Connectivity Attributes .....                        | 3682        |
| 21.2.2    | I2C Clock and Reset Management .....                     | 3683        |
| 21.2.3    | I2C Pin List .....                                       | 3683        |
| 21.3      | Functional Description .....                             | 3684        |
| 21.3.1    | Functional Block Diagram .....                           | 3684        |
| 21.3.2    | I2C Master/Slave Controller Signals .....                | 3684        |
| 21.3.3    | I2C Reset .....  | 3685        |
| 21.3.4    | Data Validity .....                                      | 3685        |
| 21.3.5    | START & STOP Conditions .....                            | 3687        |
| 21.3.6    | I2C Operation .....                                      | 3687        |
| 21.3.7    | Arbitration .....  | 3689        |
| 21.3.8    | I2C Clock Generation and I2C Clock Synchronization ..... | 3689        |
| 21.3.9    | Prescaler (SCLK/ICLK) .....                              | 3690        |
| 21.3.10   | Noise Filter .....                                       | 3690        |
| 21.3.11   | I2C Interrupts .....                                     | 3690        |
| 21.3.12   | DMA Events .....   | 3691        |
| 21.3.13   | Interrupt and DMA Events .....                           | 3691        |
| 21.3.14   | FIFO Management .....                                    | 3691        |
| 21.3.15   | How to Program I2C .....                                 | 3696        |
| 21.4      | I2C Registers .....                                      | 3698        |
| 21.4.1    | I2C Registers .....                                      | 3698        |
| <b>22</b> | <b>Multichannel Audio Serial Port (McASP) .....</b>      | <b>3750</b> |
| 22.1      | Introduction .....                                       | 3751        |
| 22.1.1    | Purpose of the Peripheral .....                          | 3751        |
| 22.1.2    | Features .....   | 3751        |
| 22.1.3    | Protocols Supported .....                                | 3751        |
| 22.1.4    | Unsupported McASP Features .....                         | 3752        |
| 22.2      | Integration .....  | 3753        |
| 22.2.1    | McASP Connectivity Attributes .....                      | 3753        |
| 22.2.2    | McASP Clock and Reset Management .....                   | 3754        |
| 22.2.3    | McASP Pin List .....                                     | 3754        |
| 22.3      | Functional Description .....                             | 3755        |
| 22.3.1    | Overview .....   | 3755        |
| 22.3.2    | Functional Block Diagram .....                           | 3756        |
| 22.3.3    | Industry Standard Compliance Statement .....             | 3759        |
| 22.3.4    | Definition of Terms .....                                | 3763        |
| 22.3.5    | Clock and Frame Sync Generators .....                    | 3765        |

|           |   |             |
|-----------|---|-------------|
| 22.3.6    | Signal Descriptions .....                               | 3769        |
| 22.3.7    | Pin Multiplexing .....                                  | 3769        |
| 22.3.8    | Transfer Modes .....                                    | 3770        |
| 22.3.9    | General Architecture .....                              | 3777        |
| 22.3.10   | Operation .....   | 3781        |
| 22.3.11   | Reset Considerations .....                              | 3798        |
| 22.3.12   | Setup and Initialization .....                          | 3798        |
| 22.3.13   | Interrupts .....  | 3803        |
| 22.3.14   | EDMA Event Support .....                                | 3805        |
| 22.3.15   | Power Management .....                                  | 3807        |
| 22.3.16   | Emulation Considerations .....                          | 3807        |
| 22.4      | McASP Registers .....                                   | 3808        |
| 22.4.1    | McASP CFG Registers .....                               | 3808        |
| 22.4.2    | McASP Data Port Registers .....                         | 3862        |
| <b>23</b> | <b>Controller Area Network (CAN) .....</b>              | <b>3863</b> |
| 23.1      | Introduction .....                                      | 3864        |
| 23.1.1    | DCAN Features .....                                     | 3864        |
| 23.1.2    | Unsupported DCAN Features .....                         | 3864        |
| 23.2      | Integration .....                                       | 3865        |
| 23.2.1    | DCAN Connectivity Attributes .....                      | 3865        |
| 23.2.2    | DCAN Clock and Reset Management .....                   | 3866        |
| 23.2.3    | DCAN Pin List .....                                     | 3866        |
| 23.3      | Functional Description .....                            | 3867        |
| 23.3.1    | CAN Core .....  | 3867        |
| 23.3.2    | Message Handler .....                                   | 3868        |
| 23.3.3    | Message RAM .....                                       | 3868        |
| 23.3.4    | Message RAM Interface .....                             | 3868        |
| 23.3.5    | Registers and Message Object Access .....               | 3868        |
| 23.3.6    | Module Interface .....                                  | 3868        |
| 23.3.7    | Dual Clock Source .....                                 | 3868        |
| 23.3.8    | CAN Operation .....                                     | 3869        |
| 23.3.9    | Dual Clock Source .....                                 | 3875        |
| 23.3.10   | Interrupt Functionality .....                           | 3876        |
| 23.3.11   | Local Power-Down Mode .....                             | 3878        |
| 23.3.12   | Parity Check Mechanism .....                            | 3880        |
| 23.3.13   | Debug/Suspend Mode .....                                | 3881        |
| 23.3.14   | Configuration of Message Objects .....                  | 3881        |
| 23.3.15   | Message Handling .....                                  | 3884        |
| 23.3.16   | CAN Bit Timing .....                                    | 3889        |
| 23.3.17   | Message Interface Register Sets .....                   | 3897        |
| 23.3.18   | Message RAM .....                                       | 3899        |
| 23.3.19   | GIO Support .....                                       | 3904        |
| 23.4      | DCAN Registers .....                                    | 3905        |
| 23.4.1    | DCAN Control Registers .....                            | 3905        |
| <b>24</b> | <b>Multichannel Serial Port Interface (McSPI) .....</b> | <b>3946</b> |
| 24.1      | Introduction .....                                      | 3947        |
| 24.1.1    | McSPI Features .....                                    | 3947        |
| 24.1.2    | Unsupported McSPI Features .....                        | 3947        |
| 24.2      | Integration .....                                       | 3947        |
| 24.2.1    | McSPI Connectivity Attributes .....                     | 3949        |
| 24.2.2    | McSPI Clock and Reset Management .....                  | 3949        |
| 24.2.3    | McSPI Pin List .....                                    | 3949        |
| 24.3      | Functional Description .....                            | 3950        |

---

|           |   |             |
|-----------|---|-------------|
| 24.3.1    | SPI Transmission .....                                  | 3950        |
| 24.3.2    | Master Mode .....                                       | 3957        |
| 24.3.3    | Slave Mode .....  | 3975        |
| 24.3.4    | Interrupts .....  | 3979        |
| 24.3.5    | DMA Requests .....                                      | 3980        |
| 24.3.6    | Emulation Mode .....                                    | 3981        |
| 24.3.7    | Power Saving Management .....                           | 3982        |
| 24.3.8    | System Test Mode .....                                  | 3983        |
| 24.3.9    | Reset .....   | 3983        |
| 24.3.10   | Access to Data Registers .....                          | 3984        |
| 24.3.11   | Programming Aid .....                                   | 3984        |
| 24.3.12   | Interrupt and DMA Events .....                          | 3985        |
| 24.4      | McSPI Registers .....                                   | 3985        |
| 24.4.1    | SPI Registers .....                                     | 3985        |
| <b>25</b> | <b>General-Purpose Input/Output .....</b>               | <b>4009</b> |
| 25.1      | Introduction .....                                      | 4010        |
| 25.1.1    | Purpose of the Peripheral .....                         | 4010        |
| 25.1.2    | GPIO Features .....                                     | 4010        |
| 25.1.3    | Unsupported GPIO Features .....                         | 4010        |
| 25.2      | Integration .....                                       | 4011        |
| 25.2.1    | GPIO Connectivity Attributes .....                      | 4011        |
| 25.2.2    | GPIO Clock and Reset Management .....                   | 4012        |
| 25.2.3    | GPIO Pin List .....                                     | 4013        |
| 25.3      | Functional Description .....                            | 4014        |
| 25.3.1    | Operating Modes .....                                   | 4014        |
| 25.3.2    | Clocking and Reset Strategy .....                       | 4014        |
| 25.3.3    | Interrupt Features .....                                | 4015        |
| 25.3.4    | General-Purpose Interface Basic Programming Model ..... | 4017        |
| 25.4      | GPIO Registers .....                                    | 4020        |
| 25.4.1    | GPIO Registers .....                                    | 4020        |
| <b>26</b> | <b>Initialization .....</b>                             | <b>4047</b> |
| 26.1      | Functional Description .....                            | 4048        |
| 26.1.1    | Architecture .....                                      | 4048        |
| 26.1.2    | Functionality .....                                     | 4048        |
| 26.1.3    | Memory Map .....  | 4049        |
| 26.1.4    | Start-up and Configuration .....                        | 4053        |
| 26.1.5    | Booting .....   | 4055        |
| 26.1.6    | Fast External Booting .....                             | 4064        |
| 26.1.7    | Memory Booting .....                                    | 4066        |
| 26.1.8    | Peripheral Booting .....                                | 4094        |
| 26.1.9    | Image Format .....                                      | 4099        |
| 26.1.10   | Code Execution .....                                    | 4100        |
| 26.1.11   | Wakeup .....  | 4101        |
| 26.1.12   | Tracing .....   | 4102        |
| <b>A</b>  | <b>Revision History .....</b>                           | <b>4106</b> |

## List of Figures

|       |  |     |
|-------|--|-----|
| 3-1.  | Microprocessor Unit (MPU) Subsystem.....                   | 162 |
| 3-2.  | Microprocessor Unit (MPU) Subsystem Signal Interface ..... | 164 |
| 3-3.  | MPU Subsystem Clocking Scheme .....                        | 165 |
| 3-4.  | Reset Scheme of the MPU Subsystem .....                    | 166 |
| 3-5.  | MPU Subsystem Power Domain Overview.....                   | 169 |
| 5-1.  | SGX530 Integration.....                                    | 179 |
| 5-2.  | SGX Block Diagram .....                                    | 181 |
| 6-1.  | Interrupt Controller Block Diagram .....                   | 184 |
| 6-2.  | IRQ/FIQ Processing Sequence .....                          | 190 |
| 6-3.  | Nested IRQ/FIQ Processing Sequence .....                   | 194 |
| 6-4.  | INTC_REVISION Register .....                               | 205 |
| 6-5.  | INTC_SYSCONFIG Register .....                              | 206 |
| 6-6.  | INTC_SYSSTATUS Register.....                               | 207 |
| 6-7.  | INTC_SIR_IRQ Register .....                                | 208 |
| 6-8.  | INTC_SIR_FIQ Register .....                                | 209 |
| 6-9.  | INTC_CONTROL Register.....                                 | 210 |
| 6-10. | INTC_PROTECTION Register .....                             | 211 |
| 6-11. | INTC_IDLE Register .....                                   | 212 |
| 6-12. | INTC_IRQ_PRIORITY Register.....                            | 213 |
| 6-13. | INTC_FIQ_PRIORITY Register .....                           | 214 |
| 6-14. | INTC_THRESHOLD Register .....                              | 215 |
| 6-15. | INTC_ITR0 Register .....                                   | 216 |
| 6-16. | INTC_MIR0 Register .....                                   | 217 |
| 6-17. | INTC_MIR_CLEAR0 Register .....                             | 218 |
| 6-18. | INTC_MIR_SET0 Register.....                                | 219 |
| 6-19. | INTC_ISR_SET0 Register .....                               | 220 |
| 6-20. | INTC_ISR_CLEAR0 Register .....                             | 221 |
| 6-21. | INTC_PENDING_IRQ0 Register .....                           | 222 |
| 6-22. | INTC_PENDING_FIQ0 Register .....                           | 223 |
| 6-23. | INTC_ITR1 Register .....                                   | 224 |
| 6-24. | INTC_MIR1 Register .....                                   | 225 |
| 6-25. | INTC_MIR_CLEAR1 Register .....                             | 226 |
| 6-26. | INTC_MIR_SET1 Register .....                               | 227 |
| 6-27. | INTC_ISR_SET1 Register .....                               | 228 |
| 6-28. | INTC_ISR_CLEAR1 Register .....                             | 229 |
| 6-29. | INTC_PENDING_IRQ1 Register .....                           | 230 |
| 6-30. | INTC_PENDING_FIQ1 Register .....                           | 231 |
| 6-31. | INTC_ITR2 Register .....                                   | 232 |
| 6-32. | INTC_MIR2 Register .....                                   | 233 |
| 6-33. | INTC_MIR_CLEAR2 Register .....                             | 234 |
| 6-34. | INTC_MIR_SET2 Register.....                                | 235 |
| 6-35. | INTC_ISR_SET2 Register .....                               | 236 |
| 6-36. | INTC_ISR_CLEAR2 Register .....                             | 237 |
| 6-37. | INTC_PENDING_IRQ2 Register .....                           | 238 |
| 6-38. | INTC_PENDING_FIQ2 Register .....                           | 239 |
| 6-39. | INTC_ITR3 Register .....                                   | 240 |
| 6-40. | INTC_MIR3 Register .....                                   | 241 |

|       |   |     |
|-------|---|-----|
| 6-41. | INTC_MIR_CLEAR3 Register .....  | 242 |
| 6-42. | INTC_MIR_SET3 Register .....  | 243 |
| 6-43. | INTC_ISR_SET3 Register .....  | 244 |
| 6-44. | INTC_ISR_CLEAR3 Register .....  | 245 |
| 6-45. | INTC_PENDING_IRQ3 Register .....  | 246 |
| 6-46. | INTC_PENDING_FIQ3 Register .....  | 247 |
| 6-47. | INTC_ILR0 to INTC_ILR127 Register .....   | 248 |
| 7-1.  | GPMC Block Diagram .....  | 252 |
| 7-2.  | GPMC Integration .....  | 253 |
| 7-3.  | GPMC to 16-Bit Address/Data-Multiplexed Memory .....  | 257 |
| 7-4.  | GPMC to 16-Bit Non-multiplexed Memory .....   | 258 |
| 7-5.  | GPMC to 8-Bit NAND Device .....   | 258 |
| 7-6.  | Chip-Select Address Mapping and Decoding Mask .....   | 263 |
| 7-7.  | Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDIVIDER = 1) .....   | 266 |
| 7-8.  | Wait Behavior During a Synchronous Read Burst Access .....  | 268 |
| 7-9.  | Read to Read for an Address-Data Multiplexed Device, On Different CS, Without Bus Turnaround (CS0n Attached to Fast Device) ..... | 270 |
| 7-10. | Read to Read / Write for an Address-Data Multiplexed Device, On Different CS, With Bus Turnaround....                             | 270 |
| 7-11. | Read to Read / Write for a Address-Data or AAD-Multiplexed Device, On Same CS, With Bus Turnaround.....                           | 271 |
| 7-12. | Asynchronous Single Read Operation on an Address/Data Multiplexed Device .....  | 280 |
| 7-13. | Two Asynchronous Single Read Accesses on an Address/Data Multiplexed Device (32-Bit Read Split Into 2 × 16-Bit Read) .....        | 281 |
| 7-14. | Asynchronous Single Write on an Address/Data-Multiplexed Device .....   | 282 |
| 7-15. | Asynchronous Single-Read on an AAD-Multiplexed Device .....   | 283 |
| 7-16. | Asynchronous Single Write on an AAD-Multiplexed Device .....  | 285 |
| 7-17. | Synchronous Single Read (GPMCFCLKDIVIDER = 0) .....   | 287 |
| 7-18. | Synchronous Single Read (GPMCFCLKDIVIDER = 1) .....   | 288 |
| 7-19. | Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0) .....   | 290 |
| 7-20. | Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1) .....   | 291 |
| 7-21. | Synchronous Single Write on an Address/Data-Multiplexed Device .....  | 292 |
| 7-22. | Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode .....   | 293 |
| 7-23. | Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode .....   | 294 |
| 7-24. | Asynchronous Single Read on an Address/Data-Nonmultiplexed Device .....   | 296 |
| 7-25. | Asynchronous Single Write on an Address/Data-Nonmultiplexed Device .....  | 297 |
| 7-26. | Asynchronous Multiple (Page Mode) Read.....   | 298 |
| 7-27. | NAND Command Latch Cycle.....   | 303 |
| 7-28. | NAND Address Latch Cycle .....  | 304 |
| 7-29. | NAND Data Read Cycle .....  | 305 |
| 7-30. | NAND Data Write Cycle.....  | 306 |
| 7-31. | Hamming Code Accumulation Algorithm (1 of 2) .....  | 310 |
| 7-32. | Hamming Code Accumulation Algorithm (2 of 2) .....  | 311 |
| 7-33. | ECC Computation for a 256-Byte Data Stream (Read or Write) .....  | 311 |
| 7-34. | ECC Computation for a 512-Byte Data Stream (Read or Write) .....  | 312 |
| 7-35. | 128 Word16 ECC Computation .....  | 313 |
| 7-36. | 256 Word16 ECC Computation .....  | 313 |
| 7-37. | Manual Mode Sequence and Mapping.....   | 318 |
| 7-38. | NAND Page Mapping and ECC: Per-Sector Schemes .....   | 323 |
| 7-39. | NAND Page Mapping and ECC: Pooled Spare Schemes.....  | 324 |
| 7-40. | NAND Page Mapping and ECC: Per-Sector Schemes, with Separate ECC.....   | 325 |

|       |   |     |
|-------|---|-----|
| 7-41. | NAND Read Cycle Optimization Timing Description .....                     | 332 |
| 7-42. | Programming Model Top-Level Diagram .....                                 | 335 |
| 7-43. | NOR Interfacing Timing Parameters Diagram .....                           | 342 |
| 7-44. | NAND Command Latch Cycle Timing Simplified Example .....                  | 346 |
| 7-45. | Synchronous NOR Single Read Simplified Example.....                       | 351 |
| 7-46. | Asynchronous NOR Single Write Simplified Example .....                    | 353 |
| 7-47. | GPMC Connection to an External NOR Flash Memory.....                      | 355 |
| 7-48. | Synchronous Burst Read Access (Timing Parameters in Clock Cycles) .....   | 357 |
| 7-49. | Asynchronous Single Read Access (Timing Parameters in Clock Cycles) ..... | 359 |
| 7-50. | Asynchronous Single Write Access (Timing Parameters in Clock Cycles)..... | 361 |
| 7-51. | GPMC_REVISION .....   | 366 |
| 7-52. | GPMC_SYSCONFIG .....  | 366 |
| 7-53. | GPMC_SYSSTATUS.....   | 367 |
| 7-54. | GPMC_IRQSTATUS .....  | 368 |
| 7-55. | GPMC_IRQENABLE .....  | 369 |
| 7-56. | GPMC_TIMEOUT_CONTROL .....  | 370 |
| 7-57. | GPMC_ERR_ADDRESS.....   | 370 |
| 7-58. | GPMC_ERR_TYPE.....  | 371 |
| 7-59. | GPMC_CONFIG .....   | 372 |
| 7-60. | GPMC_STATUS .....   | 373 |
| 7-61. | GPMC_CONFIG1_i.....   | 374 |
| 7-62. | GPMC_CONFIG2_i.....   | 376 |
| 7-63. | GPMC_CONFIG3_i.....   | 377 |
| 7-64. | GPMC_CONFIG4_i.....   | 379 |
| 7-65. | GPMC_CONFIG5_i.....   | 381 |
| 7-66. | GPMC_CONFIG6_i.....   | 382 |
| 7-67. | GPMC_CONFIG7_i.....   | 383 |
| 7-68. | GPMC_NAND_COMMAND_i .....   | 384 |
| 7-69. | GPMC_NAND_ADDRESS_i .....   | 384 |
| 7-70. | GPMC_NAND_DATA_i .....  | 384 |
| 7-71. | GPMC_PREFETCH_CONFIG1 .....   | 385 |
| 7-72. | GPMC_PREFETCH_CONFIG2 .....   | 387 |
| 7-73. | GPMC_PREFETCH_CONTROL.....  | 387 |
| 7-74. | GPMC_PREFETCH_STATUS .....  | 388 |
| 7-75. | GPMC_ECC_CONFIG .....   | 389 |
| 7-76. | GPMC_ECC_CONTROL .....  | 390 |
| 7-77. | GPMC_ECC_SIZE_CONFIG.....   | 391 |
| 7-78. | GPMC_ECCj_RESULT .....  | 393 |
| 7-79. | GPMC_BCH_RESULT0_i .....  | 394 |
| 7-80. | GPMC_BCH_RESULT1_i .....  | 394 |
| 7-81. | GPMC_BCH_RESULT2_i .....  | 394 |
| 7-82. | GPMC_BCH_RESULT3_i .....  | 395 |
| 7-83. | GPMC_BCH_SWDATA .....   | 395 |
| 7-84. | GPMC_BCH_RESULT4_i .....  | 395 |
| 7-85. | GPMC_BCH_RESULT5_i .....  | 396 |
| 7-86. | GPMC_BCH_RESULT6_i .....  | 396 |
| 7-87. | OCMC RAM Integration .....  | 398 |
| 7-88. | DDR2/3/mDDR Memory Controller Signals .....                               | 403 |
| 7-89. | DDR2/3/mDDR Subsystem Block Diagram .....                                 | 405 |

|        |  |     |
|--------|--|-----|
| 7-90.  | DDR2/3/mDDR Memory Controller FIFO Block Diagram.....  | 406 |
| 7-91.  | EMIF_MOD_ID_REV Register .....   | 423 |
| 7-92.  | STATUS Register .....  | 424 |
| 7-93.  | SDRAM_CONFIG Register .....  | 425 |
| 7-94.  | SDRAM_CONFIG_2 Register .....  | 427 |
| 7-95.  | SDRAM_REF_CTRL Register .....  | 428 |
| 7-96.  | SDRAM_REF_CTRL_SHDW Register.....  | 429 |
| 7-97.  | SDRAM_TIM_1 Register.....  | 430 |
| 7-98.  | SDRAM_TIM_1_SHDW Register .....  | 431 |
| 7-99.  | SDRAM_TIM_2 Register.....  | 432 |
| 7-100. | SDRAM_TIM_2_SHDW Register .....  | 433 |
| 7-101. | SDRAM_TIM_3 Register.....  | 434 |
| 7-102. | SDRAM_TIM_3_SHDW Register .....  | 435 |
| 7-103. | PWR_MGMT_CTRL Register .....   | 436 |
| 7-104. | PWR_MGMT_CTRL_SHDW Register .....  | 438 |
| 7-105. | Interface Configuration Register .....   | 439 |
| 7-106. | Interface Configuration Value 1 Register .....   | 440 |
| 7-107. | Interface Configuration Value 2 Register .....   | 441 |
| 7-108. | PERF_CNT_1 Register .....  | 442 |
| 7-109. | PERF_CNT_2 Register .....  | 443 |
| 7-110. | PERF_CNT_CFG Register .....  | 444 |
| 7-111. | PERF_CNT_SEL Register.....   | 445 |
| 7-112. | PERF_CNT_TIM Register .....  | 446 |
| 7-113. | READ_IDLE_CTRL Register .....  | 447 |
| 7-114. | READ_IDLE_CTRL_SHDW Register .....   | 448 |
| 7-115. | IRQSTATUS_RAW_SYS Register .....   | 449 |
| 7-116. | IRQSTATUS_SYS Register .....   | 450 |
| 7-117. | IRQENABLE_SET_SYS Register.....  | 451 |
| 7-118. | IRQENABLE_CLR_SYS Register .....   | 452 |
| 7-119. | ZQ_CONFIG Register .....   | 453 |
| 7-120. | Read-Write Leveling Ramp Window Register.....  | 454 |
| 7-121. | Read-Write Leveling Ramp Control Register.....   | 455 |
| 7-122. | Read-Write Leveling Control Register.....  | 456 |
| 7-123. | DDR_PHY_CTRL_1 Register .....  | 457 |
| 7-124. | DDR_PHY_CTRL_1_SHDW Register .....   | 459 |
| 7-125. | Priority to Class of Service Mapping Register.....   | 461 |
| 7-126. | Connection ID to Class of Service 1 Mapping Register.....  | 462 |
| 7-127. | Connection ID to Class of Service 2 Mapping Register.....  | 463 |
| 7-128. | Read Write Execution Threshold Register.....   | 465 |
| 7-129. | DDR PHY Command 0/1/2 Address/Command Slave Ratio Register<br>(CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0).....       | 468 |
| 7-130. | DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register(<br>CMD0/1/2_REG_PHY_DLL_LOCK_DIFF_0) ..... | 468 |
| 7-131. | DDR PHY Command 0/1/2 Invert Clockout Selection Register(<br>CMD0/1/2_REG_PHY_INVERT_CLKOUT_0) .....           | 469 |
| 7-132. | DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register<br>(DATA0/1_REG_PHY_RD_DQS_SLAVE_RATIO_0)).....           | 469 |
| 7-133. | DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (<br>DATA0/1_REG_PHY_WRLVL_INIT_RATIO_0) .....       | 470 |
| 7-134. | DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register   |     |

|   |     |
|---|-----|
| (DATA0/1_REG_PHY_WRLVL_INIT_MODE_0) .....   | 471 |
| 7-135. DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register<br>(DATA0_REG_PHY_GATELVL_INIT_RATIO_0).....                    | 471 |
| 7-136. DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register<br>(DATA0/1_REG_PHY_GATELVL_INIT_MODE_0) ..... | 472 |
| 7-137. DDR PHY Data Macro 0/1 DQS Gate Slave Ratio<br>Register(DATA0/1_REG_PHY_FIFO_WE_SLAVE_RATIO_0).....                        | 472 |
| 7-138. DDR PHY Data Macro 0/1 Write Data Slave Ratio Register<br>(DATA0/1_REG_PHY_WR_DATA_SLAVE_RATIO_0).....                     | 473 |
| 7-139. DDR PHY Data Macro 0/1 Delay Selection Register (DATA0/1_REG_PHY_USE_RANK0_DELAYS)....                                     | 474 |
| 7-140. ELM Integration .....  | 476 |
| 7-141. ELM Revision Register (ELM_REVISION) .....   | 487 |
| 7-142. ELM System Configuration Register (ELM_SYSCONFIG).....   | 487 |
| 7-143. ELM System Status Register (ELM_SYSSTATUS) .....   | 488 |
| 7-144. ELM Interrupt Status Register (ELM_IRQSTATUS).....   | 489 |
| 7-145. ELM Interrupt Enable Register (ELM_IRQENABLE) .....  | 491 |
| 7-146. ELM Location Configuration Register (ELM_LOCATION_CONFIG).....   | 492 |
| 7-147. ELM Page Definition Register (ELM_PAGE_CTRL) .....   | 493 |
| 7-148. ELM_SYNDROME_FRAGMENT_0_i Register .....   | 494 |
| 7-149. ELM_SYNDROME_FRAGMENT_1_i Register .....   | 494 |
| 7-150. ELM_SYNDROME_FRAGMENT_2_i Register .....   | 494 |
| 7-151. ELM_SYNDROME_FRAGMENT_3_i Register .....   | 495 |
| 7-152. ELM_SYNDROME_FRAGMENT_4_i Register .....   | 495 |
| 7-153. ELM_SYNDROME_FRAGMENT_5_i Register .....   | 495 |
| 7-154. ELM_SYNDROME_FRAGMENT_6_i Register .....   | 496 |
| 7-155. ELM_LOCATION_STATUS_i Register.....  | 496 |
| 7-156. ELM_ERROR_LOCATION_0-15_i Registers .....  | 497 |
| 8-1. Functional and Interface Clocks .....  | 499 |
| 8-2. Generic Clock Domain .....   | 504 |
| 8-3. Clock Domain State Transitions .....   | 504 |
| 8-4. Generic Power Domain Architecture .....  | 506 |
| 8-5. High Level System View for RTC-only Mode .....   | 509 |
| 8-6. System Level View of Power Management of Cortex A8 MPU and Cortex M3 .....   | 511 |
| 8-7. ADPLLs .....   | 516 |
| 8-8. Basic Structure of the ADPLLJ .....  | 518 |
| 8-9. Core PLL .....   | 521 |
| 8-10. Peripheral PLL Structure.....   | 524 |
| 8-11. MPU Subsystem PLL Structure .....   | 526 |
| 8-12. Display PLL Structure .....   | 527 |
| 8-13. DDR PLL Structure .....   | 528 |
| 8-14. CLKOUT Signals .....  | 529 |
| 8-15. Watchdog Timer Clock Selection .....  | 529 |
| 8-16. Timer Clock Selection .....   | 530 |
| 8-17. RTC, VTP, and Debounce Clock Selection .....  | 531 |
| 8-18. PORz .....  | 533 |
| 8-19. External System Reset .....   | 534 |
| 8-20. nRESETIN_OUT Waveform as Warm Reset Source .....  | 535 |
| 8-21. nRESETIN_OUT Waveform Not as Warm Reset Source .....  | 536 |
| 8-22. CM_PER_L4LS_CLKSTCTRL Register .....  | 546 |
| 8-23. CM_PER_L3S_CLKSTCTRL Register.....  | 548 |

|       |  |     |
|-------|--|-----|
| 8-24. | CM_PER_L4FW_CLKSTCTRL Register .....   | 549 |
| 8-25. | CM_PER_L3_CLKSTCTRL Register.....      | 550 |
| 8-26. | CM_PER_CPGMAC0_CLKCTRL Register.....   | 551 |
| 8-27. | CM_PER_LCDC_CLKCTRL Register.....      | 552 |
| 8-28. | CM_PER_USB0_CLKCTRL Register .....     | 553 |
| 8-29. | CM_PER_TPTC0_CLKCTRL Register.....     | 554 |
| 8-30. | CM_PER_EMIF_CLKCTRL Register.....      | 555 |
| 8-31. | CM_PER_OCMCRAM_CLKCTRL Register .....  | 556 |
| 8-32. | CM_PER_GPMC_CLKCTRL Register .....     | 557 |
| 8-33. | CM_PER_MCASP0_CLKCTRL Register .....   | 558 |
| 8-34. | CM_PER_UART5_CLKCTRL Register .....    | 559 |
| 8-35. | CM_PER_MMC0_CLKCTRL Register .....     | 560 |
| 8-36. | CM_PER_ELM_CLKCTRL Register.....       | 561 |
| 8-37. | CM_PER_I2C2_CLKCTRL Register .....     | 562 |
| 8-38. | CM_PER_I2C1_CLKCTRL Register .....     | 563 |
| 8-39. | CM_PER_SPI0_CLKCTRL Register .....     | 564 |
| 8-40. | CM_PER_SPI1_CLKCTRL Register .....     | 565 |
| 8-41. | CM_PER_L4LS_CLKCTRL Register .....     | 566 |
| 8-42. | CM_PER_L4FW_CLKCTRL Register .....     | 567 |
| 8-43. | CM_PER_MCASP1_CLKCTRL Register .....   | 568 |
| 8-44. | CM_PER_UART1_CLKCTRL Register .....    | 569 |
| 8-45. | CM_PER_UART2_CLKCTRL Register .....    | 570 |
| 8-46. | CM_PER_UART3_CLKCTRL Register .....    | 571 |
| 8-47. | CM_PER_UART4_CLKCTRL Register .....    | 572 |
| 8-48. | CM_PER_TIMER7_CLKCTRL Register .....   | 573 |
| 8-49. | CM_PER_TIMER2_CLKCTRL Register .....   | 574 |
| 8-50. | CM_PER_TIMER3_CLKCTRL Register .....   | 575 |
| 8-51. | CM_PER_TIMER4_CLKCTRL Register .....   | 576 |
| 8-52. | CM_PER_GPIO1_CLKCTRL Register .....    | 577 |
| 8-53. | CM_PER_GPIO2_CLKCTRL Register .....    | 578 |
| 8-54. | CM_PER_GPIO3_CLKCTRL Register .....    | 579 |
| 8-55. | CM_PER_TPCC_CLKCTRL Register .....     | 580 |
| 8-56. | CM_PER_DCAN0_CLKCTRL Register .....    | 581 |
| 8-57. | CM_PER_DCAN1_CLKCTRL Register .....    | 582 |
| 8-58. | CM_PER_EPWMSS1_CLKCTRL Register.....   | 583 |
| 8-59. | CM_PER_EMIF_FW_CLKCTRL Register .....  | 584 |
| 8-60. | CM_PER_EPWMSS0_CLKCTRL Register.....   | 585 |
| 8-61. | CM_PER_EPWMSS2_CLKCTRL Register.....   | 586 |
| 8-62. | CM_PER_L3_INSTR_CLKCTRL Register ..... | 587 |
| 8-63. | CM_PER_L3_CLKCTRL Register .....       | 588 |
| 8-64. | CM_PER_IEEE5000_CLKCTRL Register ..... | 589 |
| 8-65. | CM_PER_PRU_ICSS_CLKCTRL Register.....  | 590 |
| 8-66. | CM_PER_TIMER5_CLKCTRL Register .....   | 591 |
| 8-67. | CM_PER_TIMER6_CLKCTRL Register .....   | 592 |
| 8-68. | CM_PER_MMC1_CLKCTRL Register .....     | 593 |
| 8-69. | CM_PER_MMC2_CLKCTRL Register .....     | 594 |
| 8-70. | CM_PER_TPTC1_CLKCTRL Register.....     | 595 |
| 8-71. | CM_PER_TPTC2_CLKCTRL Register.....     | 596 |
| 8-72. | CM_PER_SPINLOCK_CLKCTRL Register ..... | 597 |

|   |     |
|---|-----|
| 8-73. CM_PER_MAILBOX0_CLKCTRL Register.....       | 598 |
| 8-74. CM_PER_L4HS_CLKSTCTRL Register .....        | 599 |
| 8-75. CM_PER_L4HS_CLKCTRL Register .....          | 600 |
| 8-76. CM_PER_OCPWP_L3_CLKSTCTRL Register .....    | 601 |
| 8-77. CM_PER_OCPWP_CLKCTRL Register .....         | 602 |
| 8-78. CM_PER_PRU_ICSS_CLKSTCTRL Register .....    | 603 |
| 8-79. CM_PER_CPSW_CLKSTCTRL Register.....         | 604 |
| 8-80. CM_PER_LCDC_CLKSTCTRL Register .....        | 605 |
| 8-81. CM_PER_CLKDIV32K_CLKCTRL Register.....      | 606 |
| 8-82. CM_PER_CLK_24MHZ_CLKSTCTRL Register .....   | 607 |
| 8-83. CM_WKUP_CLKSTCTRL Register.....             | 611 |
| 8-84. CM_WKUP_CONTROL_CLKCTRL Register .....      | 613 |
| 8-85. CM_WKUP_GPIO0_CLKCTRL Register .....        | 614 |
| 8-86. CM_WKUP_L4WKUP_CLKCTRL Register .....       | 615 |
| 8-87. CM_WKUP_TIMER0_CLKCTRL Register.....        | 616 |
| 8-88. CM_WKUP_DEBUGSS_CLKCTRL Register .....      | 617 |
| 8-89. CM_L3_AON_CLKSTCTRL Register .....          | 618 |
| 8-90. CM_AUTOIDLE_DPLL_MPU Register.....          | 619 |
| 8-91. CM_IDLEST_DPLL_MPU Register.....            | 620 |
| 8-92. CM_SSC_DELTAMSTEP_DPLL_MPU Register .....   | 621 |
| 8-93. CM_SSC_MODFREQDIV_DPLL_MPU Register .....   | 622 |
| 8-94. CM_CLKSEL_DPLL_MPU Register .....           | 623 |
| 8-95. CM_AUTOIDLE_DPLL_DDR Register.....          | 624 |
| 8-96. CM_IDLEST_DPLL_DDR Register.....            | 625 |
| 8-97. CM_SSC_DELTAMSTEP_DPLL_DDR Register .....   | 626 |
| 8-98. CM_SSC_MODFREQDIV_DPLL_DDR Register .....   | 627 |
| 8-99. CM_CLKSEL_DPLL_DDR Register .....           | 628 |
| 8-100. CM_AUTOIDLE_DPLL_DISP Register .....       | 629 |
| 8-101. CM_IDLEST_DPLL_DISP Register .....         | 630 |
| 8-102. CM_SSC_DELTAMSTEP_DPLL_DISP Register ..... | 631 |
| 8-103. CM_SSC_MODFREQDIV_DPLL_DISP Register.....  | 632 |
| 8-104. CM_CLKSEL_DPLL_DISP Register.....          | 633 |
| 8-105. CM_AUTOIDLE_DPLL_CORE Register.....        | 634 |
| 8-106. CM_IDLEST_DPLL_CORE Register .....         | 635 |
| 8-107. CM_SSC_DELTAMSTEP_DPLL_CORE Register ..... | 636 |
| 8-108. CM_SSC_MODFREQDIV_DPLL_CORE Register ..... | 637 |
| 8-109. CM_CLKSEL_DPLL_CORE Register .....         | 638 |
| 8-110. CM_AUTOIDLE_DPLL_PER Register .....        | 639 |
| 8-111. CM_IDLEST_DPLL_PER Register .....          | 640 |
| 8-112. CM_SSC_DELTAMSTEP_DPLL_PER Register.....   | 641 |
| 8-113. CM_SSC_MODFREQDIV_DPLL_PER Register .....  | 642 |
| 8-114. CM_CLKDCOLDO_DPLL_PER Register .....       | 643 |
| 8-115. CM_DIV_M4_DPLL_CORE Register.....          | 644 |
| 8-116. CM_DIV_M5_DPLL_CORE Register.....          | 645 |
| 8-117. CM_CLKMODE_DPLL_MPU Register .....         | 646 |
| 8-118. CM_CLKMODE_DPLL_PER Register .....         | 648 |
| 8-119. CM_CLKMODE_DPLL_CORE Register .....        | 649 |
| 8-120. CM_CLKMODE_DPLL_DDR Register .....         | 651 |
| 8-121. CM_CLKMODE_DPLL_DISP Register.....         | 653 |

|  |     |
|--|-----|
| 8-122. CM_CLKSEL_DPLL_PERIPH Register.....         | 655 |
| 8-123. CM_DIV_M2_DPLL_DDR Register.....            | 656 |
| 8-124. CM_DIV_M2_DPLL_DISP Register .....          | 657 |
| 8-125. CM_DIV_M2_DPLL_MPU Register .....           | 658 |
| 8-126. CM_DIV_M2_DPLL_PER Register.....            | 659 |
| 8-127. CM_WKUP_WKUP_M3_CLKCTRL Register .....      | 660 |
| 8-128. CM_WKUP_UART0_CLKCTRL Register.....         | 661 |
| 8-129. CM_WKUP_I2C0_CLKCTRL Register.....          | 662 |
| 8-130. CM_WKUP_ADC_TSC_CLKCTRL Register .....      | 663 |
| 8-131. CM_WKUP_SMARTREFLEX0_CLKCTRL Register ..... | 664 |
| 8-132. CM_WKUP_TIMER1_CLKCTRL Register.....        | 665 |
| 8-133. CM_WKUP_SMARTREFLEX1_CLKCTRL Register ..... | 666 |
| 8-134. CM_L4_WKUP_AON_CLKSTCTRL Register.....      | 667 |
| 8-135. CM_WKUP_WDT1_CLKCTRL Register.....          | 668 |
| 8-136. CM_DIV_M6_DPLL_CORE Register.....           | 669 |
| 8-137. CLKSEL_TIMER7_CLK Register .....            | 671 |
| 8-138. CLKSEL_TIMER2_CLK Register .....            | 672 |
| 8-139. CLKSEL_TIMER3_CLK Register .....            | 673 |
| 8-140. CLKSEL_TIMER4_CLK Register .....            | 674 |
| 8-141. CM_MAC_CLKSEL Register .....                | 675 |
| 8-142. CLKSEL_TIMER5_CLK Register .....            | 676 |
| 8-143. CLKSEL_TIMER6_CLK Register .....            | 677 |
| 8-144. CM_CPTS_RFT_CLKSEL Register .....           | 678 |
| 8-145. CLKSEL_TIMER1MS_CLK Register .....          | 679 |
| 8-146. CLKSEL_GFX_FCLK Register .....              | 680 |
| 8-147. CLKSEL_PRU_ICSS_OCP_CLK Register .....      | 681 |
| 8-148. CLKSEL_LCDC_PIXEL_CLK Register.....         | 682 |
| 8-149. CLKSEL_WDT1_CLK Register .....              | 683 |
| 8-150. CLKSEL_GPIO0_DBCLK Register .....           | 684 |
| 8-151. CM_MPUMPU_CLKSTCTRL Register.....           | 685 |
| 8-152. CM_MPUMPU_CLKCTRL Register.....             | 686 |
| 8-153. CM_CLKOUT_CTRL Register .....               | 688 |
| 8-154. CM_RTC_RTC_CLKCTRL Register.....            | 690 |
| 8-155. CM_RTC_CLKSTCTRL Register .....             | 691 |
| 8-156. CM_GFX_L3_CLKSTCTRL Register.....           | 693 |
| 8-157. CM_GFX_GFX_CLKCTRL Register.....            | 694 |
| 8-158. CM_GFX_L4LS_GFX_CLKSTCTRL Register .....    | 695 |
| 8-159. CM_GFX_MMUCFG_CLKCTRL Register .....        | 696 |
| 8-160. CM_GFX_MMUDATA_CLKCTRL Register .....       | 697 |
| 8-161. CM_CEFUSE_CLKSTCTRL Register.....           | 699 |
| 8-162. CM_CEFUSE_CEFUSE_CLKCTRL Register .....     | 700 |
| 8-163. REVISION_PRM Register .....                 | 702 |
| 8-164. PRM_IRQSTATUS_MPU Register .....            | 703 |
| 8-165. PRM_IRQENABLE_MPU Register .....            | 704 |
| 8-166. PRM_IRQSTATUS_M3 Register .....             | 705 |
| 8-167. PRM_IRQENABLE_M3 Register .....             | 706 |
| 8-168. RM_PER_RSTCTRL Register .....               | 708 |
| 8-169. PM_PER_PWRSTST Register.....                | 709 |
| 8-170. PM_PER_PWRSTCTRL Register .....             | 710 |

|   |     |
|---|-----|
| 8-171. RM_WKUP_RSTCTRL Register .....         | 712 |
| 8-172. PM_WKUP_PWRSTCTRL Register .....       | 713 |
| 8-173. PM_WKUP_PWRSTST Register .....         | 714 |
| 8-174. RM_WKUP_RSTST Register .....           | 715 |
| 8-175. PM_MPUPWRSTCTRL Register .....         | 717 |
| 8-176. PM_MPUPWRSTST Register .....           | 719 |
| 8-177. RM_MPUPRSTST Register .....            | 720 |
| 8-178. PRM_RSTCTRL Register .....             | 722 |
| 8-179. PRM_RSTTIME Register .....             | 723 |
| 8-180. PRM_RSTST Register .....               | 724 |
| 8-181. PRM_SRAM_COUNT Register .....          | 725 |
| 8-182. PRM_LDO_SRAM_CORE_SETUP Register ..... | 726 |
| 8-183. PRM_LDO_SRAM_CORE_CTRL Register .....  | 728 |
| 8-184. PRM_LDO_SRAM_MPUPSETUP Register .....  | 729 |
| 8-185. PRM_LDO_SRAM_MPUCONTROL Register ..... | 731 |
| 8-186. PM_RTC_PWRSTCTRL Register .....        | 733 |
| 8-187. PM_RTC_PWRSTST Register .....          | 734 |
| 8-188. PM_GFX_PWRSTCTRL Register .....        | 736 |
| 8-189. RM_GFX_RSTCTRL Register .....          | 737 |
| 8-190. PM_GFX_PWRSTST Register .....          | 738 |
| 8-191. RM_GFX_RSTST Register .....            | 739 |
| 8-192. PM_CEFUSE_PWRSTCTRL Register .....     | 740 |
| 8-193. PM_CEFUSE_PWRSTST Register .....       | 741 |
| 9-1. Event Crossbar .....                     | 745 |
| 9-2. USB Charger Detection .....              | 747 |
| 9-3. Timer Events .....                       | 751 |
| 9-4. control_revision Register .....          | 759 |
| 9-5. device_id Register .....                 | 760 |
| 9-6. control_hwinfo Register .....            | 761 |
| 9-7. control_sysconfig Register .....         | 762 |
| 9-8. control_status Register .....            | 763 |
| 9-9. control_emif_sdram_config Register ..... | 764 |
| 9-10. core_sldo_ctrl Register .....           | 766 |
| 9-11. mpu_sldo_ctrl Register .....            | 767 |
| 9-12. clk32kdivratio_ctrl Register .....      | 768 |
| 9-13. bandgap_ctrl Register .....             | 769 |
| 9-14. bandgap_trim Register .....             | 770 |
| 9-15. pll_clkinpulow_ctrl Register .....      | 771 |
| 9-16. mosc_ctrl Register .....                | 772 |
| 9-17. rcosc_ctrl Register .....               | 773 |
| 9-18. deepsleep_ctrl Register .....           | 774 |
| 9-19. dev_feature Register .....              | 775 |
| 9-20. init_priority_0 Register .....          | 776 |
| 9-21. init_priority_1 Register .....          | 777 |
| 9-22. mmu_cfg Register .....                  | 778 |
| 9-23. tptc_cfg Register .....                 | 779 |
| 9-24. usb_ctrl0 Register .....                | 780 |
| 9-25. usb_sts0 Register .....                 | 782 |
| 9-26. usb_ctrl1 Register .....                | 783 |

|       |                                    |     |
|-------|------------------------------------|-----|
| 9-27. | usb_sts1 Register .....            | 785 |
| 9-28. | mac_id0_lo Register .....          | 786 |
| 9-29. | mac_id0_hi Register .....          | 787 |
| 9-30. | mac_id1_lo Register .....          | 788 |
| 9-31. | mac_id1_hi Register .....          | 789 |
| 9-32. | dcan_raminit Register .....        | 790 |
| 9-33. | usb_wkup_ctrl Register .....       | 791 |
| 9-34. | gmii_sel Register .....            | 792 |
| 9-35. | pwmss_ctrl Register .....          | 793 |
| 9-36. | mreqprio_0 Register .....          | 794 |
| 9-37. | mreqprio_1 Register .....          | 795 |
| 9-38. | hw_event_sel_grp1 Register .....   | 796 |
| 9-39. | hw_event_sel_grp2 Register .....   | 797 |
| 9-40. | hw_event_sel_grp3 Register .....   | 798 |
| 9-41. | hw_event_sel_grp4 Register .....   | 799 |
| 9-42. | smrt_ctrl Register .....           | 800 |
| 9-43. | mpuss_hw_debug_sel Register .....  | 801 |
| 9-44. | mpuss_hw_dbg_info Register .....   | 802 |
| 9-45. | vdd_mpu_opp_050 Register .....     | 803 |
| 9-46. | vdd_mpu_opp_100 Register .....     | 804 |
| 9-47. | vdd_mpu_opp_120 Register .....     | 805 |
| 9-48. | vdd_mpu_opp_turbo Register .....   | 806 |
| 9-49. | vdd_core_opp_050 Register .....    | 807 |
| 9-50. | vdd_core_opp_100 Register .....    | 808 |
| 9-51. | bb_scale Register .....            | 809 |
| 9-52. | usb_vid_pid Register .....         | 810 |
| 9-53. | conf_<module>_<pin> Register ..... | 811 |
| 9-54. | cqdetect_status Register .....     | 812 |
| 9-55. | ddr_io_ctrl Register .....         | 813 |
| 9-56. | vtp_ctrl Register .....            | 814 |
| 9-57. | vref_ctrl Register .....           | 815 |
| 9-58. | tpcc_evt_mux_0_3 Register .....    | 816 |
| 9-59. | tpcc_evt_mux_4_7 Register .....    | 817 |
| 9-60. | tpcc_evt_mux_8_11 Register .....   | 818 |
| 9-61. | tpcc_evt_mux_12_15 Register .....  | 819 |
| 9-62. | tpcc_evt_mux_16_19 Register .....  | 820 |
| 9-63. | tpcc_evt_mux_20_23 Register .....  | 821 |
| 9-64. | tpcc_evt_mux_24_27 Register .....  | 822 |
| 9-65. | tpcc_evt_mux_28_31 Register .....  | 823 |
| 9-66. | tpcc_evt_mux_32_35 Register .....  | 824 |
| 9-67. | tpcc_evt_mux_36_39 Register .....  | 825 |
| 9-68. | tpcc_evt_mux_40_43 Register .....  | 826 |
| 9-69. | tpcc_evt_mux_44_47 Register .....  | 827 |
| 9-70. | tpcc_evt_mux_48_51 Register .....  | 828 |
| 9-71. | tpcc_evt_mux_52_55 Register .....  | 829 |
| 9-72. | tpcc_evt_mux_56_59 Register .....  | 830 |
| 9-73. | tpcc_evt_mux_60_63 Register .....  | 831 |
| 9-74. | timer_evt_capt Register .....      | 832 |
| 9-75. | ecap_evt_capt Register .....       | 833 |

|  |     |
|--|-----|
| 9-76. adc_evt_capt Register.....                                       | 834 |
| 9-77. reset_iso Register.....  | 835 |
| 9-78. ddr_cke_ctrl Register .....                                      | 836 |
| 9-79. sma2 Register .....  | 837 |
| 9-80. m3_txev_eoi Register .....                                       | 838 |
| 9-81. ipc_msg_reg0 Register.....                                       | 839 |
| 9-82. ipc_msg_reg1 Register.....                                       | 840 |
| 9-83. ipc_msg_reg2 Register.....                                       | 841 |
| 9-84. ipc_msg_reg3 Register.....                                       | 842 |
| 9-85. ipc_msg_reg4 Register.....                                       | 843 |
| 9-86. ipc_msg_reg5 Register.....                                       | 844 |
| 9-87. ipc_msg_reg6 Register.....                                       | 845 |
| 9-88. ipc_msg_reg7 Register.....                                       | 846 |
| 9-89. ddr_cmd0_ioctl Register .....                                    | 847 |
| 9-90. ddr_cmd1_ioctl Register .....                                    | 849 |
| 9-91. ddr_cmd2_ioctl Register .....                                    | 851 |
| 9-92. ddr_data0_ioctl Register .....                                   | 853 |
| 9-93. ddr_data1_ioctl Register .....                                   | 855 |
| 10-1. L3 Topology.....   | 859 |
| 10-2. L4 Topology.....   | 862 |
| 11-1. EDMA3 Controller Block Diagram .....                             | 864 |
| 11-2. TPCC Integration.....  | 867 |
| 11-3. TPTC Integration .....   | 868 |
| 11-4. EDMA3 Channel Controller (EDMA3CC) Block Diagram.....            | 871 |
| 11-5. EDMA3 Transfer Controller (EDMA3TC) Block Diagram .....          | 872 |
| 11-6. Definition of ACNT, BCNT, and CCNT .....                         | 873 |
| 11-7. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) .....    | 874 |
| 11-8. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3) .....   | 875 |
| 11-9. PaRAM Set .....  | 877 |
| 11-10. Channel Options Parameter (OPT).....                            | 879 |
| 11-11. Linked Transfer.....  | 886 |
| 11-12. Link-to-Self Transfer .....                                     | 887 |
| 11-13. DMA Channel and QDMA Channel to PaRAM Mapping .....             | 892 |
| 11-14. QDMA Channel to PaRAM Mapping .....                             | 893 |
| 11-15. Shadow Region Registers .....                                   | 894 |
| 11-16. Interrupt Diagram .....   | 898 |
| 11-17. Error Interrupt Operation .....                                 | 901 |
| 11-18. PaRAM Set Content for Proxy Memory Protection Example.....      | 905 |
| 11-19. Channel Options Parameter (OPT) Example .....                   | 905 |
| 11-20. Proxy Memory Protection Example .....                           | 906 |
| 11-21. EDMA3 Prioritization.....                                       | 913 |
| 11-22. Block Move Example .....  | 914 |
| 11-23. Block Move Example PaRAM Configuration .....                    | 914 |
| 11-24. Subframe Extraction Example .....                               | 915 |
| 11-25. Subframe Extraction Example PaRAM Configuration.....            | 915 |
| 11-26. Data Sorting Example .....                                      | 916 |
| 11-27. Data Sorting Example PaRAM Configuration .....                  | 917 |
| 11-28. Servicing Incoming McASP Data Example .....                     | 918 |
| 11-29. Servicing Incoming McASP Data Example PaRAM Configuration ..... | 918 |

|  |     |
|--|-----|
| 11-30. Servicing Peripheral Burst Example .....  | 919 |
| 11-31. Servicing Peripheral Burst Example PaRAM Configuration .....                      | 920 |
| 11-32. Servicing Continuous McASP Data Example .....                                     | 921 |
| 11-33. Servicing Continuous McASP Data Example PaRAM Configuration .....                 | 922 |
| 11-34. Servicing Continuous McASP Data Example Reload PaRAM Configuration .....          | 923 |
| 11-35. Ping-Pong Buffering for McASP Data Example .....                                  | 925 |
| 11-36. Ping-Pong Buffering for McASP Example PaRAM Configuration .....                   | 925 |
| 11-37. Ping-Pong Buffering for McASP Example Pong PaRAM Configuration .....              | 926 |
| 11-38. Ping-Pong Buffering for McASP Example Ping PaRAM Configuration .....              | 927 |
| 11-39. Intermediate Transfer Completion Chaining Example .....                           | 928 |
| 11-40. Single Large Block Transfer Example.....  | 929 |
| 11-41. Smaller Packet Data Transfers Example .....                                       | 929 |
| 11-42. Peripheral ID Register (PID).....   | 936 |
| 11-43. EDMA3CC Configuration Register (CCCFG) .....                                      | 937 |
| 11-44. EDMA3CC System Configuration Register (SYSCONFIG) .....                           | 939 |
| 11-45. DMA Channel Map <i>n</i> Registers (DCHMAP <i>n</i> ) .....                       | 940 |
| 11-46. QDMA Channel Map <i>n</i> Registers (QCHMAP <i>n</i> ) .....                      | 941 |
| 11-47. DMA Channel Queue <i>n</i> Number Registers (DMAQNUM <i>n</i> ) .....             | 942 |
| 11-48. QDMA Channel Queue Number Register (QDMAQNUM) .....                               | 943 |
| 11-49. Queue Priority Register (QUEPRI) .....  | 944 |
| 11-50. Event Missed Register (EMR).....  | 945 |
| 11-51. Event Missed Register High (EMRH) .....   | 945 |
| 11-52. Event Missed Clear Register (EMCR).....   | 946 |
| 11-53. Event Missed Clear Register High (EMCRH) .....                                    | 946 |
| 11-54. QDMA Event Missed Register (QEMR).....  | 947 |
| 11-55. QDMA Event Missed Clear Register (QEMCR).....                                     | 948 |
| 11-56. EDMA3CC Error Register (CCERR) .....  | 948 |
| 11-57. EDMA3CC Error Clear Register (CCERRCLR).....                                      | 949 |
| 11-58. Error Evaluation Register (EEVAL) .....   | 951 |
| 11-59. DMA Region Access Enable Register for Region <i>m</i> (DRAEm) .....               | 952 |
| 11-60. DMA Region Access Enable High Register for Region <i>m</i> (DRAEH <i>m</i> )..... | 952 |
| 11-61. QDMA Region Access Enable for Region <i>m</i> (QRAEm)32-bit, 2 Rows .....         | 953 |
| 11-62. Event Queue Entry Registers (QxEy) .....  | 954 |
| 11-63. Queue Status Register <i>n</i> (QSTAT <i>n</i> ).....                             | 955 |
| 11-64. Queue Watermark Threshold A Register (QWMTHRA) .....                              | 956 |
| 11-65. EDMA3CC Status Register (CCSTAT).....   | 957 |
| 11-66. Memory Protection Fault Address Register (MPFAR) .....                            | 959 |
| 11-67. Memory Protection Fault Status Register (MPFSR).....                              | 960 |
| 11-68. Memory Protection Fault Command Register (MPFCR).....                             | 961 |
| 11-69. Memory Protection Page Attribute Register (MPPAn) .....                           | 962 |
| 11-70. Event Register (ER) .....   | 964 |
| 11-71. Event Register High (ERH) .....   | 964 |
| 11-72. Event Clear Register (ECR) .....  | 965 |
| 11-73. Event Clear Register High (ECRH) .....  | 965 |
| 11-74. Event Set Register (ESR).....   | 966 |
| 11-75. Event Set Register High (ESRH) .....  | 967 |
| 11-76. Chained Event Register (CER).....   | 968 |
| 11-77. Chained Event Register High (CERH) .....  | 968 |
| 11-78. Event Enable Register (EER) .....   | 970 |

|  |      |
|--|------|
| 11-79. Event Enable Register High (EERH) .....                                   | 970  |
| 11-80. Event Enable Clear Register (EECR) .....                                  | 971  |
| 11-81. Event Enable Clear Register High (EECRH).....                             | 971  |
| 11-82. Event Enable Set Register (EESR).....                                     | 972  |
| 11-83. Event Enable Set Register High (EESRH) .....                              | 972  |
| 11-84. Secondary Event Register (SER) .....                                      | 973  |
| 11-85. Secondary Event Register High (SERH).....                                 | 973  |
| 11-86. Secondary Event Clear Register (SECR).....                                | 974  |
| 11-87. Secondary Event Clear Register High (SECRH) .....                         | 974  |
| 11-88. Interrupt Enable Register (IER) .....                                     | 975  |
| 11-89. Interrupt Enable Register High (IERTH) .....                              | 975  |
| 11-90. Interrupt Enable Clear Register (IECR) .....                              | 976  |
| 11-91. Interrupt Enable Clear Register High (IECRH).....                         | 976  |
| 11-92. Interrupt Enable Set Register (IESR).....                                 | 977  |
| 11-93. Interrupt Enable Set Register High (IESRH) .....                          | 977  |
| 11-94. Interrupt Pending Register (IPR).....                                     | 978  |
| 11-95. Interrupt Pending Register High (IPRH) .....                              | 978  |
| 11-96. Interrupt Clear Register (ICR) .....                                      | 979  |
| 11-97. Interrupt Clear Register High (ICRH) .....                                | 979  |
| 11-98. Interrupt Evaluate Register (IEVAL) .....                                 | 980  |
| 11-99. QDMA Event Register (QER) .....   | 981  |
| 11-100. QDMA Event Enable Register (QEER) .....                                  | 982  |
| 11-101. QDMA Event Enable Clear Register (QEECR).....                            | 983  |
| 11-102. QDMA Event Enable Set Register (QEESR) .....                             | 984  |
| 11-103. QDMA Secondary Event Register (QSER) .....                               | 985  |
| 11-104. QDMA Secondary Event Clear Register (QSECER) .....                       | 986  |
| 11-105. Peripheral ID Register (PID) .....                                       | 988  |
| 11-106. EDMA3TC Configuration Register (TCCFG).....                              | 989  |
| 11-107. EDMA3TC Channel Status Register (TCSTAT) .....                           | 990  |
| 11-108. Error Register (ERRSTAT) .....   | 992  |
| 11-109. Error Enable Register (ERREN) .....                                      | 993  |
| 11-110. Error Clear Register (ERRCLR) .....                                      | 994  |
| 11-111. Error Details Register (ERRDET) .....                                    | 995  |
| 11-112. Error Interrupt Command Register (ERRCMD).....                           | 996  |
| 11-113. Read Rate Register (RDRATE).....   | 997  |
| 11-114. Source Active Options Register (SAOPT).....                              | 998  |
| 11-115. Source Active Source Address Register (SASRC) .....                      | 1000 |
| 11-116. Source Active Count Register (SACNT) .....                               | 1000 |
| 11-117. Source Active Destination Address Register (SADST) .....                 | 1001 |
| 11-118. Source Active Source B-Dimension Index Register (SABIDX).....            | 1001 |
| 11-119. Source Active Memory Protection Proxy Register (SAMPPRXY) .....          | 1002 |
| 11-120. Source Active Count Reload Register (SACNTRLD).....                      | 1003 |
| 11-121. Source Active Source Address B-Reference Register (SASRCBREF) .....      | 1003 |
| 11-122. Source Active Destination Address B-Reference Register (SADSTBREF) ..... | 1004 |
| 11-123. Destination FIFO Options Register (DFOPT $n$ ) .....                     | 1005 |
| 11-124. Destination FIFO Source Address Register (DFSRC $n$ ) .....              | 1007 |
| 11-125. Destination FIFO Count Register (DFCNT $n$ ) .....                       | 1007 |
| 11-126. Destination FIFO Destination Address Register (DFDST $n$ ) .....         | 1008 |
| 11-127. Destination FIFO B-Index Register (DFBIDX $n$ ) .....                    | 1008 |

|  |      |
|--|------|
| 11-128. Destination FIFO Memory Protection Proxy Register (DFMPPRXY $n$ ) .....          | 1009 |
| 11-129. Destination FIFO Count Reload Register (DFCNTRLD $n$ ) .....                     | 1010 |
| 11-130. Destination FIFO Source Address B-Reference Register (DFSRCBREF $n$ ).....       | 1010 |
| 11-131. Destination FIFO Destination Address B-Reference Register (DFDSTBREF $n$ ) ..... | 1011 |
| 12-1. TSC_ADC Integration .....  | 1017 |
| 12-2. Functional Block Diagram .....   | 1020 |
| 12-3. Sequencer FSM .....  | 1023 |
| 12-4. Example Timing Diagram for Sequencer.....  | 1024 |
| 12-5. REVISION Register .....  | 1027 |
| 12-6. SYSCONFIG Register .....   | 1028 |
| 12-7. IRQSTATUS_RAW Register.....  | 1029 |
| 12-8. IRQSTATUS Register.....  | 1031 |
| 12-9. IRQENABLE_SET Register .....   | 1033 |
| 12-10. IRQENABLE_CLR Register.....   | 1035 |
| 12-11. IRQWAKEUP Register.....   | 1037 |
| 12-12. DMAENABLE_SET Register.....   | 1038 |
| 12-13. DMAENABLE_CLR Register.....   | 1039 |
| 12-14. CTRL Register .....   | 1040 |
| 12-15. ADCSTAT Register.....   | 1041 |
| 12-16. ADCRANGE Register.....  | 1042 |
| 12-17. ADC_CLKDIV Register .....   | 1043 |
| 12-18. ADC_MISC Register .....   | 1044 |
| 12-19. STEPENABLE Register .....   | 1045 |
| 12-20. IDLECONFIG Register.....  | 1046 |
| 12-21. TS_CHARGE_STEPCONFIG Register.....  | 1047 |
| 12-22. TS_CHARGE_DELAY Register .....  | 1048 |
| 12-23. STEPCONFIG1 Register .....  | 1049 |
| 12-24. STEPDELAY1 Register.....  | 1050 |
| 12-25. STEPCONFIG2 Register .....  | 1051 |
| 12-26. STEPDELAY2 Register.....  | 1052 |
| 12-27. STEPCONFIG3 Register .....  | 1053 |
| 12-28. STEPDELAY3 Register.....  | 1054 |
| 12-29. STEPCONFIG4 Register .....  | 1055 |
| 12-30. STEPDELAY4 Register.....  | 1056 |
| 12-31. STEPCONFIG5 Register .....  | 1057 |
| 12-32. STEPDELAY5 Register.....  | 1058 |
| 12-33. STEPCONFIG6 Register .....  | 1059 |
| 12-34. STEPDELAY6 Register.....  | 1060 |
| 12-35. STEPCONFIG7 Register .....  | 1061 |
| 12-36. STEPDELAY7 Register.....  | 1062 |
| 12-37. STEPCONFIG8 Register .....  | 1063 |
| 12-38. STEPDELAY8 Register.....  | 1064 |
| 12-39. STEPCONFIG9 Register .....  | 1065 |
| 12-40. STEPDELAY9 Register.....  | 1066 |
| 12-41. STEPCONFIG10 Register.....  | 1067 |
| 12-42. STEPDELAY10 Register .....  | 1068 |
| 12-43. STEPCONFIG11 Register.....  | 1069 |
| 12-44. STEPDELAY11 Register .....  | 1070 |
| 12-45. STEPCONFIG12 Register.....  | 1071 |

|   |      |
|---|------|
| 12-46. STEPDELAY12 Register .....   | 1072 |
| 12-47. STEPCONFIG13 Register.....   | 1073 |
| 12-48. STEPDELAY13 Register .....   | 1074 |
| 12-49. STEPCONFIG14 Register.....   | 1075 |
| 12-50. STEPDELAY14 Register .....   | 1076 |
| 12-51. STEPCONFIG15 Register.....   | 1077 |
| 12-52. STEPDELAY15 Register .....   | 1078 |
| 12-53. STEPCONFIG16 Register.....   | 1079 |
| 12-54. STEPDELAY16 Register .....   | 1080 |
| 12-55. FIFO0COUNT Register .....  | 1081 |
| 12-56. FIFO0THRESHOLD Register.....                                       | 1082 |
| 12-57. DMA0REQ Register.....  | 1083 |
| 12-58. FIFO1COUNT Register .....  | 1084 |
| 12-59. FIFO1THRESHOLD Register.....                                       | 1085 |
| 12-60. DMA1REQ Register.....  | 1086 |
| 12-61. FIFO0DATA Register .....   | 1087 |
| 12-62. FIFO1DATA Register .....   | 1088 |
| 13-1. LCD Controller .....  | 1090 |
| 13-2. LCD Controller Integration.....                                     | 1092 |
| 13-3. Input and Output Clocks .....                                       | 1094 |
| 13-4. Logical Data Path for Raster Controller .....                       | 1101 |
| 13-5. Frame Buffer Structure .....  | 1102 |
| 13-6. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP).....           | 1103 |
| 13-7. 256-Entry Palette/Buffer Format (8 BPP) .....                       | 1104 |
| 13-8. 16-BPP Data Memory Organization (TFT Mode Only)—Little Endian ..... | 1104 |
| 13-9. 12-BPP Data Memory Organization—Little Endian .....                 | 1105 |
| 13-10. 8-BPP Data Memory Organization .....                               | 1105 |
| 13-11. 4-BPP Data Memory Organization .....                               | 1105 |
| 13-12. 2-BPP Data Memory Organization .....                               | 1106 |
| 13-13. 1-BPP Data Memory Organization .....                               | 1106 |
| 13-14. Monochrome and Color Output.....                                   | 1108 |
| 13-15. Example of Subpicture .....  | 1109 |
| 13-16. Subpicture HOLS Bit .....  | 1109 |
| 13-17. Raster Mode Display Format .....                                   | 1110 |
| 13-18. Palette Lookup Examples .....                                      | 1118 |
| 13-19. PID Register.....  | 1122 |
| 13-20. CTRL Register .....  | 1123 |
| 13-21. LIDD_CTRL Register .....   | 1124 |
| 13-22. LIDD_CS0_CONF Register .....                                       | 1125 |
| 13-23. LIDD_CS0_ADDR Register .....                                       | 1126 |
| 13-24. LIDD_CS0_DATA Register.....  | 1127 |
| 13-25. LIDD_CS1_CONF Register .....                                       | 1128 |
| 13-26. LIDD_CS1_ADDR Register .....                                       | 1129 |
| 13-27. LIDD_CS1_DATA Register.....  | 1130 |
| 13-28. RASTER_CTRL Register.....  | 1131 |
| 13-29. RASTER_TIMING_0 Register .....                                     | 1133 |
| 13-30. RASTER_TIMING_1 Register .....                                     | 1134 |
| 13-31. RASTER_TIMING_2 Register .....                                     | 1135 |
| 13-32. RASTER_SUBPANEL Register .....                                     | 1137 |

|  |      |
|--|------|
| 13-33. RASTER_SUBPANEL2 Register .....         | 1138 |
| 13-34. LCDDMA_CTRL Register .....              | 1139 |
| 13-35. LCDDMA_FBO_BASE Register.....           | 1140 |
| 13-36. LCDDMA_FBO_CEILING Register .....       | 1141 |
| 13-37. LCDDMA_FBI_BASE Register.....           | 1142 |
| 13-38. LCDDMA_FBI_CEILING Register .....       | 1143 |
| 13-39. SYSCONFIG Register .....                | 1144 |
| 13-40. IRQSTATUS_RAW Register .....            | 1145 |
| 13-41. IRQSTATUS Register.....                 | 1147 |
| 13-42. IRQENABLE_SET Register .....            | 1149 |
| 13-43. IRQENABLE_CLEAR Register .....          | 1151 |
| 13-44. CLKC_ENABLE Register.....               | 1153 |
| 13-45. CLKC_RESET Register .....               | 1154 |
| 14-1. Ethernet Switch Integration .....        | 1158 |
| 14-2. Ethernet Switch RMII Clock Detail .....  | 1162 |
| 14-3. MII Interface Connections .....          | 1163 |
| 14-4. RMII Interface Connections .....         | 1165 |
| 14-5. RGMII Interface Connections .....        | 1166 |
| 14-6. CPSW_3G Block Diagram .....              | 1174 |
| 14-7. Tx Buffer Descriptor Format .....        | 1179 |
| 14-8. Rx Buffer Descriptor Format .....        | 1182 |
| 14-9. VLAN Header Encapsulation Word .....     | 1186 |
| 14-10. CPTS Block Diagram .....                | 1221 |
| 14-11. Event FIFO Misalignment Condition ..... | 1223 |
| 14-12. HW1/4_TSP_PUSH Connection.....          | 1224 |
| 14-13. Port TX State RAM Entry .....           | 1229 |
| 14-14. Port RX DMA State.....                  | 1230 |
| 14-15. IDVER Register.....                     | 1234 |
| 14-16. CONTROL Register .....                  | 1235 |
| 14-17. PRESCALE Register.....                  | 1237 |
| 14-18. UNKNOWN_VLAN Register .....             | 1238 |
| 14-19. TBLCTL Register .....                   | 1239 |
| 14-20. TBLW2 Register .....                    | 1240 |
| 14-21. TBLW1 Register .....                    | 1241 |
| 14-22. TBLW0 Register .....                    | 1242 |
| 14-23. PORTCTL0 Register .....                 | 1243 |
| 14-24. PORTCTL1 Register .....                 | 1244 |
| 14-25. PORTCTL2 Register .....                 | 1245 |
| 14-26. PORTCTL3 Register .....                 | 1246 |
| 14-27. PORTCTL4 Register .....                 | 1247 |
| 14-28. PORTCTL5 Register .....                 | 1248 |
| 14-29. TX_IDVER Register .....                 | 1251 |
| 14-30. TX_CONTROL Register .....               | 1252 |
| 14-31. TX_TEARDOWN Register .....              | 1253 |
| 14-32. RX_IDVER Register.....                  | 1254 |
| 14-33. RX_CONTROL Register.....                | 1255 |
| 14-34. RX_TEARDOWN Register .....              | 1256 |
| 14-35. CPDMA_SOFT_RESET Register .....         | 1257 |
| 14-36. DMACONTROL Register .....               | 1258 |

|   |      |
|---|------|
| 14-37. DMASTATUS Register .....           | 1260 |
| 14-38. RX_BUFFER_OFFSET Register .....    | 1262 |
| 14-39. EMCONTROL Register .....           | 1263 |
| 14-40. TX_PRI0_RATE Register .....        | 1264 |
| 14-41. TX_PRI1_RATE Register .....        | 1265 |
| 14-42. TX_PRI2_RATE Register .....        | 1266 |
| 14-43. TX_PRI3_RATE Register .....        | 1267 |
| 14-44. TX_PRI4_RATE Register .....        | 1268 |
| 14-45. TX_PRI5_RATE Register .....        | 1269 |
| 14-46. TX_PRI6_RATE Register .....        | 1270 |
| 14-47. TX_PRI7_RATE Register .....        | 1271 |
| 14-48. TX_INTSTAT_RAW Register .....      | 1272 |
| 14-49. TX_INTSTAT_MASKED Register .....   | 1273 |
| 14-50. TX_INTMASK_SET Register .....      | 1274 |
| 14-51. TX_INTMASK_CLEAR Register .....    | 1275 |
| 14-52. CPDMA_IN_VECTOR Register .....     | 1276 |
| 14-53. CPDMA_EOI_VECTOR Register .....    | 1277 |
| 14-54. RX_INTSTAT_RAW Register .....      | 1278 |
| 14-55. RX_INTSTAT_MASKED Register .....   | 1279 |
| 14-56. RX_INTMASK_SET Register .....      | 1280 |
| 14-57. RX_INTMASK_CLEAR Register .....    | 1281 |
| 14-58. DMA_INTSTAT_RAW Register .....     | 1282 |
| 14-59. DMA_INTSTAT_MASKED Register .....  | 1283 |
| 14-60. DMA_INTMASK_SET Register .....     | 1284 |
| 14-61. DMA_INTMASK_CLEAR Register .....   | 1285 |
| 14-62. RX0_PENDTHRESH Register .....      | 1286 |
| 14-63. RX1_PENDTHRESH Register .....      | 1287 |
| 14-64. RX2_PENDTHRESH Register .....      | 1288 |
| 14-65. RX3_PENDTHRESH Register .....      | 1289 |
| 14-66. RX4_PENDTHRESH Register .....      | 1290 |
| 14-67. RX5_PENDTHRESH Register .....      | 1291 |
| 14-68. RX6_PENDTHRESH Register .....      | 1292 |
| 14-69. RX7_PENDTHRESH Register .....      | 1293 |
| 14-70. RX0_FREEBUFFER Register .....      | 1294 |
| 14-71. RX1_FREEBUFFER Register .....      | 1295 |
| 14-72. RX2_FREEBUFFER Register .....      | 1296 |
| 14-73. RX3_FREEBUFFER Register .....      | 1297 |
| 14-74. RX4_FREEBUFFER Register .....      | 1298 |
| 14-75. RX5_FREEBUFFER Register .....      | 1299 |
| 14-76. RX6_FREEBUFFER Register .....      | 1300 |
| 14-77. RX7_FREEBUFFER Register .....      | 1301 |
| 14-78. CPTS_IDVER Register .....          | 1303 |
| 14-79. CPTS_CONTROL Register .....        | 1304 |
| 14-80. CPTS_TS_PUSH Register .....        | 1305 |
| 14-81. CPTS_TS_LOAD_VAL Register .....    | 1306 |
| 14-82. CPTS_TS_LOAD_EN Register .....     | 1307 |
| 14-83. CPTS_INTSTAT_RAW Register .....    | 1308 |
| 14-84. CPTS_INTSTAT_MASKED Register ..... | 1309 |
| 14-85. CPTS_INT_ENABLE Register .....     | 1310 |

---

|  |      |
|--|------|
| 14-86. CPTS_EVENT_POP Register .....       | 1311 |
| 14-87. CPTS_EVENT_LOW Register.....        | 1312 |
| 14-88. CPTS_EVENT_HIGH Register .....      | 1313 |
| 14-89. TX0_HDP Register .....              | 1317 |
| 14-90. TX1_HDP Register .....              | 1318 |
| 14-91. TX2_HDP Register .....              | 1319 |
| 14-92. TX3_HDP Register .....              | 1320 |
| 14-93. TX4_HDP Register .....              | 1321 |
| 14-94. TX5_HDP Register .....              | 1322 |
| 14-95. TX6_HDP Register .....              | 1323 |
| 14-96. TX7_HDP Register .....              | 1324 |
| 14-97. RX0_HDP Register.....               | 1325 |
| 14-98. RX1_HDP Register.....               | 1326 |
| 14-99. RX2_HDP Register.....               | 1327 |
| 14-100. RX3_HDP Register .....             | 1328 |
| 14-101. RX4_HDP Register .....             | 1329 |
| 14-102. RX5_HDP Register .....             | 1330 |
| 14-103. RX6_HDP Register .....             | 1331 |
| 14-104. RX7_HDP Register .....             | 1332 |
| 14-105. TX0_CP Register.....               | 1333 |
| 14-106. TX1_CP Register.....               | 1334 |
| 14-107. TX2_CP Register.....               | 1335 |
| 14-108. TX3_CP Register.....               | 1336 |
| 14-109. TX4_CP Register.....               | 1337 |
| 14-110. TX5_CP Register.....               | 1338 |
| 14-111. TX6_CP Register.....               | 1339 |
| 14-112. TX7_CP Register.....               | 1340 |
| 14-113. RX0_CP Register .....              | 1341 |
| 14-114. RX1_CP Register .....              | 1342 |
| 14-115. RX2_CP Register .....              | 1343 |
| 14-116. RX3_CP Register .....              | 1344 |
| 14-117. RX4_CP Register .....              | 1345 |
| 14-118. RX5_CP Register .....              | 1346 |
| 14-119. RX6_CP Register .....              | 1347 |
| 14-120. RX7_CP Register .....              | 1348 |
| 14-121. P0_CONTROL Register.....           | 1350 |
| 14-122. P0_MAX_BLKS Register.....          | 1351 |
| 14-123. P0_BLK_CNT Register.....           | 1352 |
| 14-124. P0_TX_IN_CTL Register.....         | 1353 |
| 14-125. P0_PORT_VLAN Register .....        | 1354 |
| 14-126. P0_TX_PRI_MAP Register .....       | 1355 |
| 14-127. P0_CPDMA_TX_PRI_MAP Register ..... | 1356 |
| 14-128. P0_CPDMA_RX_CH_MAP Register .....  | 1357 |
| 14-129. P0_RX_DSCP_PRI_MAP0 Register ..... | 1358 |
| 14-130. P0_RX_DSCP_PRI_MAP1 Register ..... | 1359 |
| 14-131. P0_RX_DSCP_PRI_MAP2 Register ..... | 1360 |
| 14-132. P0_RX_DSCP_PRI_MAP3 Register ..... | 1361 |
| 14-133. P0_RX_DSCP_PRI_MAP4 Register ..... | 1362 |
| 14-134. P0_RX_DSCP_PRI_MAP5 Register ..... | 1363 |

|  |      |
|--|------|
| 14-135. P0_RX_DSCP_PRI_MAP6 Register ..... | 1364 |
| 14-136. P0_RX_DSCP_PRI_MAP7 Register ..... | 1365 |
| 14-137. P1_CONTROL Register.....           | 1366 |
| 14-138. P1_MAX_BLKS Register.....          | 1368 |
| 14-139. P1_BLK_CNT Register.....           | 1369 |
| 14-140. P1_TX_IN_CTL Register.....         | 1370 |
| 14-141. P1_PORT_VLAN Register .....        | 1371 |
| 14-142. P1_TX_PRI_MAP Register .....       | 1372 |
| 14-143. P1_TS_SEQ_MTYPE Register .....     | 1373 |
| 14-144. P1_SA_LO Register .....            | 1374 |
| 14-145. P1_SA_HI Register .....            | 1375 |
| 14-146. P1_SEND_PERCENT Register .....     | 1376 |
| 14-147. P1_RX_DSCP_PRI_MAP0 Register ..... | 1377 |
| 14-148. P1_RX_DSCP_PRI_MAP1 Register ..... | 1378 |
| 14-149. P1_RX_DSCP_PRI_MAP2 Register ..... | 1379 |
| 14-150. P1_RX_DSCP_PRI_MAP3 Register ..... | 1380 |
| 14-151. P1_RX_DSCP_PRI_MAP4 Register ..... | 1381 |
| 14-152. P1_RX_DSCP_PRI_MAP5 Register ..... | 1382 |
| 14-153. P1_RX_DSCP_PRI_MAP6 Register ..... | 1383 |
| 14-154. P1_RX_DSCP_PRI_MAP7 Register ..... | 1384 |
| 14-155. P2_CONTROL Register.....           | 1385 |
| 14-156. P2_MAX_BLKS Register.....          | 1387 |
| 14-157. P2_BLK_CNT Register .....          | 1388 |
| 14-158. P2_TX_IN_CTL Register.....         | 1389 |
| 14-159. P2_PORT_VLAN Register .....        | 1390 |
| 14-160. P2_TX_PRI_MAP Register .....       | 1391 |
| 14-161. P2_TS_SEQ_MTYPE Register .....     | 1392 |
| 14-162. P2_SA_LO Register .....            | 1393 |
| 14-163. P2_SA_HI Register .....            | 1394 |
| 14-164. P2_SEND_PERCENT Register .....     | 1395 |
| 14-165. P2_RX_DSCP_PRI_MAP0 Register ..... | 1396 |
| 14-166. P2_RX_DSCP_PRI_MAP1 Register ..... | 1397 |
| 14-167. P2_RX_DSCP_PRI_MAP2 Register ..... | 1398 |
| 14-168. P2_RX_DSCP_PRI_MAP3 Register ..... | 1399 |
| 14-169. P2_RX_DSCP_PRI_MAP4 Register ..... | 1400 |
| 14-170. P2_RX_DSCP_PRI_MAP5 Register ..... | 1401 |
| 14-171. P2_RX_DSCP_PRI_MAP6 Register ..... | 1402 |
| 14-172. P2_RX_DSCP_PRI_MAP7 Register ..... | 1403 |
| 14-173. IDVER Register .....               | 1405 |
| 14-174. MACCONTROL Register.....           | 1406 |
| 14-175. MACSTATUS Register .....           | 1409 |
| 14-176. SOFT_RESET Register .....          | 1410 |
| 14-177. RX_MAXLEN Register .....           | 1411 |
| 14-178. BOFFTEST Register.....             | 1412 |
| 14-179. RX_PAUSE Register .....            | 1413 |
| 14-180. TX_PAUSE Register.....             | 1414 |
| 14-181. EMCONTROL Register .....           | 1415 |
| 14-182. RX_PRI_MAP Register.....           | 1416 |
| 14-183. TX_GAP Register .....              | 1417 |

|  |      |
|--|------|
| 14-184. ID_VER Register .....                | 1418 |
| 14-185. CONTROL Register.....                | 1419 |
| 14-186. SOFT_RESET Register .....            | 1420 |
| 14-187. STAT_PORT_EN Register.....           | 1421 |
| 14-188. PTYPE Register .....                 | 1422 |
| 14-189. SOFT_IDLE Register .....             | 1423 |
| 14-190. THRU_RATE Register.....              | 1424 |
| 14-191. GAP_THRESH Register .....            | 1425 |
| 14-192. TX_START_WDS Register .....          | 1426 |
| 14-193. FLOW_CONTROL Register .....          | 1427 |
| 14-194. VLAN_LTYPE Register.....             | 1428 |
| 14-195. TS_LTYPE Register .....              | 1429 |
| 14-196. DLR_LTYPE Register .....             | 1430 |
| 14-197. IDVER Register .....                 | 1432 |
| 14-198. SOFT_RESET Register .....            | 1433 |
| 14-199. CONTROL Register.....                | 1434 |
| 14-200. INT_CONTROL Register.....            | 1435 |
| 14-201. C0_RX_THRESH_EN Register .....       | 1436 |
| 14-202. C0_RX_EN Register .....              | 1437 |
| 14-203. C0_TX_EN Register .....              | 1438 |
| 14-204. C0_MISC_EN Register.....             | 1439 |
| 14-205. C1_RX_THRESH_EN Register .....       | 1440 |
| 14-206. C1_RX_EN Register .....              | 1441 |
| 14-207. C1_TX_EN Register .....              | 1442 |
| 14-208. C1_MISC_EN Register.....             | 1443 |
| 14-209. C2_RX_THRESH_EN Register .....       | 1444 |
| 14-210. C2_RX_EN Register .....              | 1445 |
| 14-211. C2_TX_EN Register .....              | 1446 |
| 14-212. C2_MISC_EN Register.....             | 1447 |
| 14-213. C0_RX_THRESH_STAT Register .....     | 1448 |
| 14-214. C0_RX_STAT Register.....             | 1449 |
| 14-215. C0_TX_STAT Register .....            | 1450 |
| 14-216. C0_MISC_STAT Register.....           | 1451 |
| 14-217. C1_RX_THRESH_STAT Register .....     | 1452 |
| 14-218. C1_RX_STAT Register.....             | 1453 |
| 14-219. C1_TX_STAT Register .....            | 1454 |
| 14-220. C1_MISC_STAT Register.....           | 1455 |
| 14-221. C2_RX_THRESH_STAT Register .....     | 1456 |
| 14-222. C2_RX_STAT Register .....            | 1457 |
| 14-223. C2_TX_STAT Register .....            | 1458 |
| 14-224. C2_MISC_STAT Register.....           | 1459 |
| 14-225. C0_RX_IMAX Register .....            | 1460 |
| 14-226. C0_TX_IMAX Register .....            | 1461 |
| 14-227. C1_RX_IMAX Register .....            | 1462 |
| 14-228. C1_TX_IMAX Register .....            | 1463 |
| 14-229. C2_RX_IMAX Register .....            | 1464 |
| 14-230. C2_TX_IMAX Register .....            | 1465 |
| 14-231. RGMII_CTL Register .....             | 1466 |
| 14-232. MDIO Version Register (MDIOVER)..... | 1467 |

|  |      |
|--|------|
| 14-233. MDIO Control Register (MDIOCONTROL) .....  | 1468 |
| 14-234. PHY Acknowledge Status Register (MDIOALIVE).....   | 1469 |
| 14-235. PHY Link Status Register (MDIOLINK) .....  | 1469 |
| 14-236. MDIO Link Status Change Interrupt Register (MDIOLINKINTRAW) .....  | 1470 |
| 14-237. MDIO Link Status Change Interrupt Register (Masked Value) (MDIOLINKINTMASKED) .....                                | 1470 |
| 14-238. MDIO User Command Complete Interrupt Register (Raw Value) (MDIOUSERINTRAW) .....                                   | 1471 |
| 14-239. MDIO User Command Complete Interrupt Register (Masked Value) (MDIOUSERINTMASKED) .....                             | 1471 |
| 14-240. MDIO User Command Complete Interrupt Mask Set Register (MDIOUSERINTMASKSET) .....                                  | 1472 |
| 14-241. MDIO User Command Complete Interrupt Mask Clear Register (MDIOUSERINTMASKCLR).....                                 | 1472 |
| 14-242. MDIO User Access Register 0 (MDIOUSERACCESS0).....   | 1473 |
| 14-243. MDIO User PHY Select Register 0 (MDIOUSERPHYSEL0).....   | 1474 |
| 14-244. MDIO User Access Register 1 (MDIOUSERACCESS1).....   | 1475 |
| 14-245. MDIO User PHY Select Register 1 (MDIOUSERPHYSEL1).....   | 1476 |
| 15-1. PWMSS Integration .....  | 1480 |
| 15-2. IDVER Register .....   | 1482 |
| 15-3. SYSCONFIG Register .....   | 1483 |
| 15-4. CLKCONFIG Register .....   | 1484 |
| 15-5. CLKSTATUS Register .....   | 1485 |
| 15-6. Multiple ePWM Modules .....  | 1487 |
| 15-7. Submodules and Signal Connections for an ePWM Module .....   | 1488 |
| 15-8. ePWM Submodules and Critical Internal Signal Interconnects.....  | 1489 |
| 15-9. Time-Base Submodule Block Diagram .....  | 1493 |
| 15-10. Time-Base Submodule Signals and Registers .....   | 1495 |
| 15-11. Time-Base Frequency and Period .....  | 1497 |
| 15-12. Time-Base Counter Synchronization Scheme 1 .....  | 1498 |
| 15-13. Time-Base Up-Count Mode Waveforms.....  | 1500 |
| 15-14. Time-Base Down-Count Mode Waveforms .....   | 1501 |
| 15-15. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event...                         | 1501 |
| 15-16. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event .....                        | 1502 |
| 15-17. Counter-Compare Submodule .....   | 1503 |
| 15-18. Counter-Compare Submodule Signals and Registers.....  | 1503 |
| 15-19. Counter-Compare Event Waveforms in Up-Count Mode .....  | 1506 |
| 15-20. Counter-Compare Events in Down-Count Mode .....   | 1506 |
| 15-21. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on Synchronization Event .....           | 1507 |
| 15-22. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event .....             | 1507 |
| 15-23. Action-Qualifier Submodule .....  | 1508 |
| 15-24. Action-Qualifier Submodule Inputs and Outputs .....   | 1509 |
| 15-25. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs.....  | 1510 |
| 15-26. Up-Down-Count Mode Symmetrical Waveform .....   | 1513 |
| 15-27. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High .....             | 1514 |
| 15-28. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low.....                | 1516 |
| 15-29. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA .....                           | 1518 |
| 15-30. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low .....    | 1520 |
| 15-31. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary ..... | 1522 |

|  |      |
|--|------|
| 15-32. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low .....    | 1524 |
| 15-33. Dead-Band Generator Submodule .....   | 1526 |
| 15-34. Configuration Options for the Dead-Band Generator Submodule .....                                       | 1527 |
| 15-35. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%) .....  | 1529 |
| 15-36. PWM-Chopper Submodule.....  | 1530 |
| 15-37. PWM-Chopper Submodule Signals and Registers .....   | 1531 |
| 15-38. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only .....                               | 1532 |
| 15-39. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses .....          | 1532 |
| 15-40. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses ..... | 1533 |
| 15-41. Trip-Zone Submodule .....   | 1534 |
| 15-42. Trip-Zone Submodule Mode Control Logic .....  | 1537 |
| 15-43. Trip-Zone Submodule Interrupt Logic.....  | 1537 |
| 15-44. Event-Trigger Submodule .....   | 1538 |
| 15-45. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller .....                                | 1539 |
| 15-46. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs.....                                 | 1539 |
| 15-47. Event-Trigger Interrupt Generator .....   | 1541 |
| 15-48. HRPWM System Interface .....  | 1542 |
| 15-49. Resolution Calculations for Conventionally Generated PWM .....  | 1543 |
| 15-50. Operating Logic Using MEP .....   | 1544 |
| 15-51. Required PWM Waveform for a Requested Duty = 40.5% .....  | 1546 |
| 15-52. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz.....                               | 1548 |
| 15-53. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz.....                              | 1548 |
| 15-54. Simplified ePWM Module.....   | 1549 |
| 15-55. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave .....                                 | 1550 |
| 15-56. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$ .....            | 1551 |
| 15-57. Buck Waveforms for (Note: Only three bucks shown here).....   | 1552 |
| 15-58. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$ ).....                               | 1554 |
| 15-59. Buck Waveforms for (Note: $F_{PWM2} = F_{PWM1}$ ).....  | 1555 |
| 15-60. Control of Two Half-H Bridge Stages ( $F_{PWM2} = N \times F_{PWM1}$ ).....                             | 1557 |
| 15-61. Half-H Bridge Waveforms for (Note: Here $F_{PWM2} = F_{PWM1}$ ) .....                                   | 1558 |
| 15-62. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control .....                      | 1560 |
| 15-63. 3-Phase Inverter Waveforms for (Only One Inverter Shown) .....  | 1561 |
| 15-64. Configuring Two PWM Modules for Phase Control .....   | 1564 |
| 15-65. Timing Waveforms Associated With Phase Control Between 2 Modules.....                                   | 1565 |
| 15-66. Control of a 3-Phase Interleaved DC/DC Converter .....  | 1566 |
| 15-67. 3-Phase Interleaved DC/DC Converter Waveforms for .....   | 1567 |
| 15-68. Controlling a Full-H Bridge Stage ( $F_{PWM2} = F_{PWM1}$ ) .....                                       | 1570 |
| 15-69. ZVS Full-H Bridge Waveforms.....  | 1571 |
| 15-70. Time-Base Control Register (TBCTL) .....  | 1574 |
| 15-71. Time-Base Status Register (TBSTS) .....   | 1576 |
| 15-72. Time-Base Phase Register (TBPHS) .....  | 1576 |
| 15-73. Time-Base Counter Register (TBCNT) .....  | 1577 |
| 15-74. Time-Base Period Register (TBPRD) .....   | 1578 |
| 15-75. Counter-Compare Control Register (CMPCTL) .....   | 1579 |
| 15-76. Counter-Compare A Register (CMPA) .....   | 1580 |
| 15-77. Counter-Compare B Register (CMPB).....  | 1581 |
| 15-78. Action-Qualifier Output A Control Register (AQCTLA) .....   | 1582 |

|  |      |
|--|------|
| 15-79. Action-Qualifier Output B Control Register (AQCTLB) .....                         | 1583 |
| 15-80. Action-Qualifier Software Force Register (AQSFRC).....                            | 1584 |
| 15-81. Action-Qualifier Continuous Software Force Register (AQCSFRC) .....               | 1585 |
| 15-82. Dead-Band Generator Control Register (DBCTL).....                                 | 1586 |
| 15-83. Dead-Band Generator Rising Edge Delay Register (DBRED) .....                      | 1587 |
| 15-84. Dead-Band Generator Falling Edge Delay Register (DBFED) .....                     | 1587 |
| 15-85. Trip-Zone Select Register (TZSEL) .....   | 1588 |
| 15-86. Trip-Zone Control Register (TZCTL) .....  | 1589 |
| 15-87. Trip-Zone Enable Interrupt Register (TZEINT) .....                                | 1589 |
| 15-88. Trip-Zone Flag Register (TZFLG) .....   | 1590 |
| 15-89. Trip-Zone Clear Register (TZCLR) .....  | 1591 |
| 15-90. Trip-Zone Force Register (TZFRC) .....  | 1591 |
| 15-91. Event-Trigger Selection Register (ETSEL).....                                     | 1592 |
| 15-92. Event-Trigger Prescale Register (ETPS).....                                       | 1593 |
| 15-93. Event-Trigger Flag Register (ETFLG) .....   | 1594 |
| 15-94. Event-Trigger Clear Register (ETCLR) .....  | 1594 |
| 15-95. Event-Trigger Force Register (ETFRC) .....  | 1595 |
| 15-96. PWM-Chopper Control Register (PCCTL) .....  | 1596 |
| 15-97. Time-Base Phase High-Resolution Register (TBPHSHR) .....                          | 1597 |
| 15-98. Counter-Compare A High-Resolution Register (CMPAHR).....                          | 1597 |
| 15-99. HRPWM Control Register (HRCTL).....   | 1598 |
| 15-100. Multiple eCAP Modules .....  | 1600 |
| 15-101. Capture and APWM Modes of Operation .....  | 1601 |
| 15-102. Capture Function Diagram .....   | 1602 |
| 15-103. Event Prescale Control.....  | 1603 |
| 15-104. Prescale Function Waveforms .....  | 1603 |
| 15-105. Continuous/One-shot Block Diagram .....  | 1604 |
| 15-106. Counter and Synchronization Block Diagram .....                                  | 1605 |
| 15-107. Interrupts in eCAP Module .....  | 1607 |
| 15-108. PWM Waveform Details Of APWM Mode Operation .....                                | 1608 |
| 15-109. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect.....                | 1611 |
| 15-110. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect.....    | 1613 |
| 15-111. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect .....             | 1615 |
| 15-112. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect ..... | 1617 |
| 15-113. PWM Waveform Details of APWM Mode Operation.....                                 | 1619 |
| 15-114. Multichannel PWM Example Using 4 eCAP Modules .....                              | 1621 |
| 15-115. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules .....          | 1624 |
| 15-116. Time-Stamp Counter Register (TSCTR) .....  | 1626 |
| 15-117. Counter Phase Control Register (CTRPHS) .....                                    | 1627 |
| 15-118. Capture 1 Register (CAP1) .....  | 1627 |
| 15-119. Capture 2 Register (CAP2) .....  | 1628 |
| 15-120. Capture 3 Register (CAP3) .....  | 1628 |
| 15-121. Capture 4 Register (CAP4) .....  | 1629 |
| 15-122. ECAP Control Register 1 (ECCTL1).....  | 1629 |
| 15-123. ECAP Control Register 2 (ECCTL2).....  | 1631 |
| 15-124. ECAP Interrupt Enable Register (ECEINT) .....                                    | 1633 |
| 15-125. ECAP Interrupt Flag Register (ECFLG) .....                                       | 1634 |
| 15-126. ECAP Interrupt Clear Register (ECCLR).....                                       | 1635 |
| 15-127. ECAP Interrupt Forcing Register (ECFRC) .....                                    | 1636 |

|  |      |
|--|------|
| 15-128. Revision ID Register (REVID) .....   | 1637 |
| 15-129. Optical Encoder Disk .....   | 1638 |
| 15-130. QEP Encoder Output Signal for Forward/Reverse Movement .....                               | 1639 |
| 15-131. Index Pulse Example .....  | 1639 |
| 15-132. Functional Block Diagram of the eQEP Peripheral .....                                      | 1642 |
| 15-133. Functional Block Diagram of Decoder Unit .....   | 1643 |
| 15-134. Quadrature Decoder State Machine .....   | 1645 |
| 15-135. Quadrature-clock and Direction Decoding .....  | 1645 |
| 15-136. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or F9Fh) ..... | 1647 |
| 15-137. Position Counter Underflow/Overflow (QPOSMAX = 4) .....                                    | 1648 |
| 15-138. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1) .....                        | 1650 |
| 15-139. Strobe Event Latch (QEPCTL[SEL] = 1) .....   | 1651 |
| 15-140. eQEP Position-compare Unit .....   | 1652 |
| 15-141. eQEP Position-compare Event Generation Points .....  | 1653 |
| 15-142. eQEP Position-compare Sync Output Pulse Stretcher.....                                     | 1653 |
| 15-143. eQEP Edge Capture Unit .....   | 1655 |
| 15-144. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010).....                  | 1655 |
| 15-145. eQEP Edge Capture Unit - Timing Details.....   | 1656 |
| 15-146. eQEP Watchdog Timer .....  | 1657 |
| 15-147. eQEP Unit Time Base .....  | 1658 |
| 15-148. EQEP Interrupt Generation .....  | 1658 |
| 15-149. eQEP Position Counter Register (QPOS_CNT).....   | 1660 |
| 15-150. eQEP Position Counter Initialization Register (QPOS_INIT) .....                            | 1660 |
| 15-151. eQEP Maximum Position Count Register (QPOSMAX) .....                                       | 1660 |
| 15-152. eQEP Position-Compare Register (QPOS_CMP).....   | 1661 |
| 15-153. eQEP Index Position Latch Register (QPOS_INDEX_LATCH) .....                                | 1661 |
| 15-154. eQEP Strobe Position Latch Register (QPOSSLATCH) .....                                     | 1661 |
| 15-155. eQEP Position Counter Latch Register (QPOS_LATCH) .....                                    | 1662 |
| 15-156. eQEP Unit Timer Register (QUTMR).....  | 1662 |
| 15-157. eQEP Unit Period Register (QUPRD) .....  | 1662 |
| 15-158. eQEP Watchdog Timer Register (QWDTMR) .....  | 1663 |
| 15-159. eQEP Watchdog Period Register (QWDPRD) .....   | 1663 |
| 15-160. QEP Decoder Control Register (QDECCTL) .....   | 1664 |
| 15-161. eQEP Control Register (QEPCTL) .....   | 1665 |
| 15-162. eQEP Capture Control Register (QCAPCTL) .....  | 1667 |
| 15-163. eQEP Position-Compare Control Register (QPOSCTL) .....                                     | 1668 |
| 15-164. eQEP Interrupt Enable Register (QEINT).....  | 1669 |
| 15-165. eQEP Interrupt Flag Register (QFLG).....   | 1670 |
| 15-166. eQEP Interrupt Clear Register (QCLR) .....   | 1671 |
| 15-167. eQEP Interrupt Force Register (QFRC).....  | 1673 |
| 15-168. eQEP Status Register (QEPSTS).....   | 1674 |
| 15-169. eQEP Capture Timer Register (QCTMR) .....  | 1675 |
| 15-170. eQEP Capture Period Register (QCPRD) .....   | 1675 |
| 15-171. eQEP Capture Timer Latch Register (QCTMRLAT).....  | 1675 |
| 15-172. eQEP Capture Period Latch Register (QCPRDLAT).....   | 1676 |
| 15-173. eQEP Revision ID Register (REVID) .....  | 1676 |
| 16-1. USB Integration .....  | 1681 |
| 16-2. USB GPIO Integration .....   | 1683 |
| 16-3. CPU Actions at Transfer Phases .....   | 1691 |

|  |      |
|--|------|
| 16-4. Sequence of Transfer .....   | 1692 |
| 16-5. Flow Chart of Setup Stage of a Control Transfer in Peripheral Mode .....                                   | 1694 |
| 16-6. Flow Chart of Transmit Data Stage of a Control Transfer in Peripheral Mode .....                           | 1695 |
| 16-7. Flow Chart of Receive Data Stage of a Control Transfer in Peripheral Mode .....                            | 1696 |
| 16-8. Flow Chart of Setup Stage of a Control Transfer in Host Mode .....   | 1707 |
| 16-9. Flow Chart of Data Stage (IN Data Phase) of a Control Transfer in Host Mode .....                          | 1708 |
| 16-10. Flow Chart of Data Stage (OUT Data Phase) of a Control Transfer in Host Mode .....                        | 1710 |
| 16-11. Flow Chart of Status Stage of Zero Data Request or Write Request of a Control Transfer in Host Mode ..... | 1711 |
| 16-12. Chart of Status Stage of a Read Request of a Control Transfer in Host Mode .....                          | 1713 |
| 16-13. Packet Descriptor Layout .....  | 1723 |
| 16-14. Buffer Descriptor (BD) Layout .....   | 1726 |
| 16-15. Teardown Descriptor Layout .....  | 1728 |
| 16-16. Relationship Between Memory Regions and Linking RAM .....   | 1733 |
| 16-17. High-level Transmit and Receive Data Transfer Example .....   | 1738 |
| 16-18. Transmit Descriptors and Queue Status Configuration .....   | 1740 |
| 16-19. Transmit USB Data Flow Example (Initialization) .....   | 1741 |
| 16-20. Receive Buffer Descriptors and Queue Status Configuration .....   | 1743 |
| 16-21. Receive USB Data Flow Example (Initialization) .....  | 1744 |
| 16-22. REVREG Register .....   | 1749 |
| 16-23. SYS CONFIG Register .....   | 1750 |
| 16-24. IRQSTATRAW Register .....   | 1751 |
| 16-25. IRQSTAT Register .....  | 1752 |
| 16-26. IRQENABLER Register .....   | 1753 |
| 16-27. IRQCLEARR Register .....  | 1754 |
| 16-28. IRQDMATHOLDTX00 Register .....  | 1755 |
| 16-29. IRQDMATHOLDTX01 Register .....  | 1756 |
| 16-30. IRQDMATHOLDTX02 Register .....  | 1757 |
| 16-31. IRQDMATHOLDTX03 Register .....  | 1758 |
| 16-32. IRQDMATHOLDRX00 Register .....  | 1759 |
| 16-33. IRQDMATHOLDRX01 Register .....  | 1760 |
| 16-34. IRQDMATHOLDRX02 Register .....  | 1761 |
| 16-35. IRQDMATHOLDRX03 Register .....  | 1762 |
| 16-36. IRQDMATHOLDTX10 Register .....  | 1763 |
| 16-37. IRQDMATHOLDTX11 Register .....  | 1764 |
| 16-38. IRQDMATHOLDTX12 Register .....  | 1765 |
| 16-39. IRQDMATHOLDTX13 Register .....  | 1766 |
| 16-40. IRQDMATHOLDRX10 Register .....  | 1767 |
| 16-41. IRQDMATHOLDRX11 Register .....  | 1768 |
| 16-42. IRQDMATHOLDRX12 Register .....  | 1769 |
| 16-43. IRQDMATHOLDRX13 Register .....  | 1770 |
| 16-44. IRQDMAENABLE0 Register .....  | 1771 |
| 16-45. IRQDMAENABLE1 Register .....  | 1772 |
| 16-46. IRQFRAMETHOLDTX00 Register .....  | 1773 |
| 16-47. IRQFRAMETHOLDTX01 Register .....  | 1774 |
| 16-48. IRQFRAMETHOLDTX02 Register .....  | 1775 |
| 16-49. IRQFRAMETHOLDTX03 Register .....  | 1776 |
| 16-50. IRQFRAMETHOLDRX00 Register .....  | 1777 |
| 16-51. IRQFRAMETHOLDRX01 Register .....  | 1778 |
| 16-52. IRQFRAMETHOLDRX02 Register .....  | 1779 |

|   |      |
|---|------|
| 16-53. IRQFRAMETHOLDRX03 Register ..... | 1780 |
| 16-54. IRQFRAMETHOLDTX10 Register ..... | 1781 |
| 16-55. IRQFRAMETHOLDTX11 Register ..... | 1782 |
| 16-56. IRQFRAMETHOLDTX12 Register ..... | 1783 |
| 16-57. IRQFRAMETHOLDTX13 Register ..... | 1784 |
| 16-58. IRQFRAMETHOLDRX10 Register ..... | 1785 |
| 16-59. IRQFRAMETHOLDRX11 Register ..... | 1786 |
| 16-60. IRQFRAMETHOLDRX12 Register ..... | 1787 |
| 16-61. IRQFRAMETHOLDRX13 Register ..... | 1788 |
| 16-62. IRQFRAMEENABLE0 Register.....    | 1789 |
| 16-63. IRQFRAMEENABLE1 Register.....    | 1790 |
| 16-64. USB0REV Register.....            | 1792 |
| 16-65. USB0CTRL Register .....          | 1793 |
| 16-66. USB0STAT Register .....          | 1795 |
| 16-67. USB0IRQMSTAT Register .....      | 1796 |
| 16-68. USB0IRQSTATRAW0 Register.....    | 1797 |
| 16-69. USB0IRQSTATRAW1 Register.....    | 1799 |
| 16-70. USB0IRQSTAT0 Register .....      | 1801 |
| 16-71. USB0IRQSTAT1 Register .....      | 1803 |
| 16-72. USB0IRQENABLESET0 Register ..... | 1805 |
| 16-73. USB0IRQENABLESET1 Register ..... | 1807 |
| 16-74. USB0IRQENABLECLR0 Register ..... | 1809 |
| 16-75. USB0IRQENABLECLR1 Register ..... | 1811 |
| 16-76. USB0TXMODE Register.....         | 1813 |
| 16-77. USB0RXMODE Register .....        | 1815 |
| 16-78. USB0GENRNDISEP1 Register.....    | 1819 |
| 16-79. USB0GENRNDISEP2 Register.....    | 1820 |
| 16-80. USB0GENRNDISEP3 Register.....    | 1821 |
| 16-81. USB0GENRNDISEP4 Register.....    | 1822 |
| 16-82. USB0GENRNDISEP5 Register.....    | 1823 |
| 16-83. USB0GENRNDISEP6 Register.....    | 1824 |
| 16-84. USB0GENRNDISEP7 Register.....    | 1825 |
| 16-85. USB0GENRNDISEP8 Register.....    | 1826 |
| 16-86. USB0GENRNDISEP9 Register.....    | 1827 |
| 16-87. USB0GENRNDISEP10 Register .....  | 1828 |
| 16-88. USB0GENRNDISEP11 Register .....  | 1829 |
| 16-89. USB0GENRNDISEP12 Register .....  | 1830 |
| 16-90. USB0GENRNDISEP13 Register .....  | 1831 |
| 16-91. USB0GENRNDISEP14 Register .....  | 1832 |
| 16-92. USB0GENRNDISEP15 Register .....  | 1833 |
| 16-93. USB0AUTOREQ Register.....        | 1834 |
| 16-94. USB0SRPFIXTIME Register.....     | 1836 |
| 16-95. USB0_TDOWN Register .....        | 1837 |
| 16-96. USB0UTMI Register .....          | 1838 |
| 16-97. USB0MGCUTMILB Register .....     | 1839 |
| 16-98. USB0MODE Register .....          | 1840 |
| 16-99. USB1REV Register.....            | 1842 |
| 16-100. USB1CTRL Register.....          | 1843 |
| 16-101. USB1STAT Register.....          | 1845 |

|  |      |
|--|------|
| 16-102. USB1IRQMSTAT Register .....          | 1846 |
| 16-103. USB1IRQSTATRAW0 Register .....       | 1847 |
| 16-104. USB1IRQSTATRAW1 Register .....       | 1849 |
| 16-105. USB1IRQSTAT0 Register .....          | 1851 |
| 16-106. USB1IRQSTAT1 Register .....          | 1853 |
| 16-107. USB1IRQENABLESET0 Register .....     | 1855 |
| 16-108. USB1IRQENABLESET1 Register .....     | 1857 |
| 16-109. USB1IRQENABLECLR0 Register .....     | 1859 |
| 16-110. USB1IRQENABLECLR1 Register .....     | 1861 |
| 16-111. USB1TXMODE Register .....            | 1863 |
| 16-112. USB1RXMODE Register .....            | 1865 |
| 16-113. USB1GENRNDISEP1 Register .....       | 1867 |
| 16-114. USB1GENRNDISEP2 Register .....       | 1868 |
| 16-115. USB1GENRNDISEP3 Register .....       | 1869 |
| 16-116. USB1GENRNDISEP4 Register .....       | 1870 |
| 16-117. USB1GENRNDISEP5 Register .....       | 1871 |
| 16-118. USB1GENRNDISEP6 Register .....       | 1872 |
| 16-119. USB1GENRNDISEP7 Register .....       | 1873 |
| 16-120. USB1GENRNDISEP8 Register .....       | 1874 |
| 16-121. USB1GENRNDISEP9 Register .....       | 1875 |
| 16-122. USB1GENRNDISEP10 Register .....      | 1876 |
| 16-123. USB1GENRNDISEP11 Register .....      | 1877 |
| 16-124. USB1GENRNDISEP12 Register .....      | 1878 |
| 16-125. USB1GENRNDISEP13 Register .....      | 1879 |
| 16-126. USB1GENRNDISEP14 Register .....      | 1880 |
| 16-127. USB1GENRNDISEP15 Register .....      | 1881 |
| 16-128. USB1AUTOREQ Register .....           | 1882 |
| 16-129. USB1SRPFIXTIME Register .....        | 1884 |
| 16-130. USB1TDOWN Register .....             | 1885 |
| 16-131. USB1UTMI Register .....              | 1886 |
| 16-132. USB1UTMILB Register .....            | 1887 |
| 16-133. USB1MODE Register .....              | 1888 |
| 16-134. Termination_control Register .....   | 1890 |
| 16-135. RX_CALIB Register .....              | 1891 |
| 16-136. DLLHS_2 Register .....               | 1893 |
| 16-137. RX_TEST_2 Register .....             | 1894 |
| 16-138. CHRG_DET Register .....              | 1895 |
| 16-139. PWR_CNTL Register .....              | 1897 |
| 16-140. UTMI_INTERFACE_CNTL_1 Register ..... | 1898 |
| 16-141. UTMI_INTERFACE_CNTL_2 Register ..... | 1899 |
| 16-142. BIST Register .....                  | 1901 |
| 16-143. BIST_CRC Register .....              | 1902 |
| 16-144. CDR_BIST2 Register .....             | 1903 |
| 16-145. GPIO Register .....                  | 1904 |
| 16-146. DLLHS Register .....                 | 1905 |
| 16-147. USB2PHYCM_CONFIG Register .....      | 1906 |
| 16-148. AD_INTERFACE_REG1 Register .....     | 1907 |
| 16-149. AD_INTERFACE_REG2 Register .....     | 1909 |
| 16-150. AD_INTERFACE_REG3 Register .....     | 1911 |

|                                    |      |
|------------------------------------|------|
| 16-151. ANA_CONFIG2 Register ..... | 1912 |
| 16-152. DMAREVID Register .....    | 1916 |
| 16-153. TDFDQ Register .....       | 1917 |
| 16-154. DMAEMU Register .....      | 1918 |
| 16-155. TXGCR0 Register .....      | 1919 |
| 16-156. RXGCR0 Register .....      | 1920 |
| 16-157. RXHPCRA0 Register .....    | 1922 |
| 16-158. RXHPCRB0 Register .....    | 1923 |
| 16-159. TXGCR1 Register .....      | 1924 |
| 16-160. RXGCR1 Register .....      | 1925 |
| 16-161. RXHPCRA1 Register .....    | 1927 |
| 16-162. RXHPCRB1 Register .....    | 1928 |
| 16-163. TXGCR2 Register .....      | 1929 |
| 16-164. RXGCR2 Register .....      | 1930 |
| 16-165. RXHPCRA2 Register .....    | 1932 |
| 16-166. RXHPCRB2 Register .....    | 1933 |
| 16-167. TXGCR3 Register .....      | 1934 |
| 16-168. RXGCR3 Register .....      | 1935 |
| 16-169. RXHPCRA3 Register .....    | 1937 |
| 16-170. RXHPCRB3 Register .....    | 1938 |
| 16-171. TXGCR4 Register .....      | 1939 |
| 16-172. RXGCR4 Register .....      | 1940 |
| 16-173. RXHPCRA4 Register .....    | 1942 |
| 16-174. RXHPCRB4 Register .....    | 1943 |
| 16-175. TXGCR5 Register .....      | 1944 |
| 16-176. RXGCR5 Register .....      | 1945 |
| 16-177. RXHPCRA5 Register .....    | 1947 |
| 16-178. RXHPCRB5 Register .....    | 1948 |
| 16-179. TXGCR6 Register .....      | 1949 |
| 16-180. RXGCR6 Register .....      | 1950 |
| 16-181. RXHPCRA6 Register .....    | 1952 |
| 16-182. RXHPCRB6 Register .....    | 1953 |
| 16-183. TXGCR7 Register .....      | 1954 |
| 16-184. RXGCR7 Register .....      | 1955 |
| 16-185. RXHPCRA7 Register .....    | 1957 |
| 16-186. RXHPCRB7 Register .....    | 1958 |
| 16-187. TXGCR8 Register .....      | 1959 |
| 16-188. RXGCR8 Register .....      | 1960 |
| 16-189. RXHPCRA8 Register .....    | 1962 |
| 16-190. RXHPCRB8 Register .....    | 1963 |
| 16-191. TXGCR9 Register .....      | 1964 |
| 16-192. RXGCR9 Register .....      | 1965 |
| 16-193. RXHPCRA9 Register .....    | 1967 |
| 16-194. RXHPCRB9 Register .....    | 1968 |
| 16-195. TXGCR10 Register .....     | 1969 |
| 16-196. RXGCR10 Register .....     | 1970 |
| 16-197. RXHPCRA10 Register .....   | 1972 |
| 16-198. RXHPCRB10 Register .....   | 1973 |
| 16-199. TXGCR11 Register .....     | 1974 |

|                                 |      |
|---------------------------------|------|
| 16-200. RXGCR11 Register .....  | 1975 |
| 16-201. RXHPCRA11 Register..... | 1977 |
| 16-202. RXHPCRB11 Register..... | 1978 |
| 16-203. TXGCR12 Register .....  | 1979 |
| 16-204. RXGCR12 Register .....  | 1980 |
| 16-205. RXHPCRA12 Register..... | 1982 |
| 16-206. RXHPCRB12 Register..... | 1983 |
| 16-207. TXGCR13 Register .....  | 1984 |
| 16-208. RXGCR13 Register .....  | 1985 |
| 16-209. RXHPCRA13 Register..... | 1987 |
| 16-210. RXHPCRB13 Register..... | 1988 |
| 16-211. TXGCR14 Register .....  | 1989 |
| 16-212. RXGCR14 Register .....  | 1990 |
| 16-213. RXHPCRA14 Register..... | 1992 |
| 16-214. RXHPCRB14 Register..... | 1993 |
| 16-215. TXGCR15 Register .....  | 1994 |
| 16-216. RXGCR15 Register .....  | 1995 |
| 16-217. RXHPCRA15 Register..... | 1997 |
| 16-218. RXHPCRB15 Register..... | 1998 |
| 16-219. TXGCR16 Register .....  | 1999 |
| 16-220. RXGCR16 Register .....  | 2000 |
| 16-221. RXHPCRA16 Register..... | 2002 |
| 16-222. RXHPCRB16 Register..... | 2003 |
| 16-223. TXGCR17 Register .....  | 2004 |
| 16-224. RXGCR17 Register .....  | 2005 |
| 16-225. RXHPCRA17 Register..... | 2007 |
| 16-226. RXHPCRB17 Register..... | 2008 |
| 16-227. TXGCR18 Register .....  | 2009 |
| 16-228. RXGCR18 Register .....  | 2010 |
| 16-229. RXHPCRA18 Register..... | 2012 |
| 16-230. RXHPCRB18 Register..... | 2013 |
| 16-231. TXGCR19 Register .....  | 2014 |
| 16-232. RXGCR19 Register .....  | 2015 |
| 16-233. RXHPCRA19 Register..... | 2017 |
| 16-234. RXHPCRB19 Register..... | 2018 |
| 16-235. TXGCR20 Register .....  | 2019 |
| 16-236. RXGCR20 Register .....  | 2020 |
| 16-237. RXHPCRA20 Register..... | 2022 |
| 16-238. RXHPCRB20 Register..... | 2023 |
| 16-239. TXGCR21 Register .....  | 2024 |
| 16-240. RXGCR21 Register .....  | 2025 |
| 16-241. RXHPCRA21 Register..... | 2027 |
| 16-242. RXHPCRB21 Register..... | 2028 |
| 16-243. TXGCR22 Register .....  | 2029 |
| 16-244. RXGCR22 Register .....  | 2030 |
| 16-245. RXHPCRA22 Register..... | 2032 |
| 16-246. RXHPCRB22 Register..... | 2033 |
| 16-247. TXGCR23 Register .....  | 2034 |
| 16-248. RXGCR23 Register .....  | 2035 |

|  |      |
|--|------|
| 16-249. RXHPCRA23 Register.....        | 2037 |
| 16-250. RXHPCRB23 Register.....        | 2038 |
| 16-251. TXGCR24 Register .....         | 2039 |
| 16-252. RXGCR24 Register .....         | 2040 |
| 16-253. RXHPCRA24 Register.....        | 2042 |
| 16-254. RXHPCRB24 Register.....        | 2043 |
| 16-255. TXGCR25 Register .....         | 2044 |
| 16-256. RXGCR25 Register .....         | 2045 |
| 16-257. RXHPCRA25 Register.....        | 2047 |
| 16-258. RXHPCRB25 Register.....        | 2048 |
| 16-259. TXGCR26 Register .....         | 2049 |
| 16-260. RXGCR26 Register .....         | 2050 |
| 16-261. RXHPCRA26 Register.....        | 2052 |
| 16-262. RXHPCRB26 Register.....        | 2053 |
| 16-263. TXGCR27 Register .....         | 2054 |
| 16-264. RXGCR27 Register .....         | 2055 |
| 16-265. RXHPCRA27 Register.....        | 2057 |
| 16-266. RXHPCRB27 Register.....        | 2058 |
| 16-267. TXGCR28 Register .....         | 2059 |
| 16-268. RXGCR28 Register .....         | 2060 |
| 16-269. RXHPCRA28 Register.....        | 2062 |
| 16-270. RXHPCRB28 Register.....        | 2063 |
| 16-271. TXGCR29 Register .....         | 2064 |
| 16-272. RXGCR29 Register .....         | 2065 |
| 16-273. RXHPCRA29 Register.....        | 2067 |
| 16-274. RXHPCRB29 Register.....        | 2068 |
| 16-275. DMA_SCHED_CTRL Register .....  | 2069 |
| 16-276. WORD0 to WORD63 Register ..... | 2070 |
| 16-277. QMGRREVID Register .....       | 2096 |
| 16-278. QMGRRST Register .....         | 2097 |
| 16-279. FDBSC0 Register .....          | 2098 |
| 16-280. FDBSC1 Register .....          | 2099 |
| 16-281. FDBSC2 Register .....          | 2100 |
| 16-282. FDBSC3 Register .....          | 2101 |
| 16-283. FDBSC4 Register .....          | 2102 |
| 16-284. FDBSC5 Register .....          | 2103 |
| 16-285. FDBSC6 Register .....          | 2104 |
| 16-286. FDBSC7 Register .....          | 2105 |
| 16-287. LRAM0BASE Register.....        | 2106 |
| 16-288. LRAM0SIZE Register.....        | 2107 |
| 16-289. LRAM1BASE Register.....        | 2108 |
| 16-290. PEND0 Register.....            | 2109 |
| 16-291. PEND1 Register.....            | 2110 |
| 16-292. PEND2 Register.....            | 2111 |
| 16-293. PEND3 Register.....            | 2112 |
| 16-294. PEND4 Register.....            | 2113 |
| 16-295. QMEMRBASE0 Register.....       | 2114 |
| 16-296. QMEMCTRL0 Register.....        | 2115 |
| 16-297. QMEMRBASE1 Register.....       | 2116 |

|                                  |      |
|----------------------------------|------|
| 16-298. QMEMCTRL1 Register.....  | 2117 |
| 16-299. QMEMRBASE2 Register..... | 2118 |
| 16-300. QMEMCTRL2 Register.....  | 2119 |
| 16-301. QMEMRBASE3 Register..... | 2120 |
| 16-302. QMEMCTRL3 Register.....  | 2121 |
| 16-303. QMEMRBASE4 Register..... | 2122 |
| 16-304. QMEMCTRL4 Register.....  | 2123 |
| 16-305. QMEMRBASE5 Register..... | 2124 |
| 16-306. QMEMCTRL5 Register.....  | 2125 |
| 16-307. QMEMRBASE6 Register..... | 2126 |
| 16-308. QMEMCTRL6 Register.....  | 2127 |
| 16-309. QMEMRBASE7 Register..... | 2128 |
| 16-310. QMEMCTRL7 Register.....  | 2129 |
| 16-311. QUEUE_0_A Register.....  | 2130 |
| 16-312. QUEUE_0_B Register.....  | 2131 |
| 16-313. QUEUE_0_C Register.....  | 2132 |
| 16-314. QUEUE_0_D Register.....  | 2133 |
| 16-315. QUEUE_1_A Register.....  | 2134 |
| 16-316. QUEUE_1_B Register.....  | 2135 |
| 16-317. QUEUE_1_C Register.....  | 2136 |
| 16-318. QUEUE_1_D Register.....  | 2137 |
| 16-319. QUEUE_2_A Register.....  | 2138 |
| 16-320. QUEUE_2_B Register.....  | 2139 |
| 16-321. QUEUE_2_C Register.....  | 2140 |
| 16-322. QUEUE_2_D Register.....  | 2141 |
| 16-323. QUEUE_3_A Register.....  | 2142 |
| 16-324. QUEUE_3_B Register.....  | 2143 |
| 16-325. QUEUE_3_C Register.....  | 2144 |
| 16-326. QUEUE_3_D Register.....  | 2145 |
| 16-327. QUEUE_4_A Register.....  | 2146 |
| 16-328. QUEUE_4_B Register.....  | 2147 |
| 16-329. QUEUE_4_C Register.....  | 2148 |
| 16-330. QUEUE_4_D Register.....  | 2149 |
| 16-331. QUEUE_5_A Register.....  | 2150 |
| 16-332. QUEUE_5_B Register.....  | 2151 |
| 16-333. QUEUE_5_C Register.....  | 2152 |
| 16-334. QUEUE_5_D Register.....  | 2153 |
| 16-335. QUEUE_6_A Register.....  | 2154 |
| 16-336. QUEUE_6_B Register.....  | 2155 |
| 16-337. QUEUE_6_C Register.....  | 2156 |
| 16-338. QUEUE_6_D Register.....  | 2157 |
| 16-339. QUEUE_7_A Register.....  | 2158 |
| 16-340. QUEUE_7_B Register.....  | 2159 |
| 16-341. QUEUE_7_C Register.....  | 2160 |
| 16-342. QUEUE_7_D Register.....  | 2161 |
| 16-343. QUEUE_8_A Register.....  | 2162 |
| 16-344. QUEUE_8_B Register.....  | 2163 |
| 16-345. QUEUE_8_C Register.....  | 2164 |
| 16-346. QUEUE_8_D Register.....  | 2165 |

---

|                                   |      |
|-----------------------------------|------|
| 16-347. QUEUE_9_A Register .....  | 2166 |
| 16-348. QUEUE_9_B Register .....  | 2167 |
| 16-349. QUEUE_9_C Register .....  | 2168 |
| 16-350. QUEUE_9_D Register .....  | 2169 |
| 16-351. QUEUE_10_A Register ..... | 2170 |
| 16-352. QUEUE_10_B Register ..... | 2171 |
| 16-353. QUEUE_10_C Register ..... | 2172 |
| 16-354. QUEUE_10_D Register ..... | 2173 |
| 16-355. QUEUE_11_A Register ..... | 2174 |
| 16-356. QUEUE_11_B Register ..... | 2175 |
| 16-357. QUEUE_11_C Register ..... | 2176 |
| 16-358. QUEUE_11_D Register ..... | 2177 |
| 16-359. QUEUE_12_A Register ..... | 2178 |
| 16-360. QUEUE_12_B Register ..... | 2179 |
| 16-361. QUEUE_12_C Register ..... | 2180 |
| 16-362. QUEUE_12_D Register ..... | 2181 |
| 16-363. QUEUE_13_A Register ..... | 2182 |
| 16-364. QUEUE_13_B Register ..... | 2183 |
| 16-365. QUEUE_13_C Register ..... | 2184 |
| 16-366. QUEUE_13_D Register ..... | 2185 |
| 16-367. QUEUE_14_A Register ..... | 2186 |
| 16-368. QUEUE_14_B Register ..... | 2187 |
| 16-369. QUEUE_14_C Register ..... | 2188 |
| 16-370. QUEUE_14_D Register ..... | 2189 |
| 16-371. QUEUE_15_A Register ..... | 2190 |
| 16-372. QUEUE_15_B Register ..... | 2191 |
| 16-373. QUEUE_15_C Register ..... | 2192 |
| 16-374. QUEUE_15_D Register ..... | 2193 |
| 16-375. QUEUE_16_A Register ..... | 2194 |
| 16-376. QUEUE_16_B Register ..... | 2195 |
| 16-377. QUEUE_16_C Register ..... | 2196 |
| 16-378. QUEUE_16_D Register ..... | 2197 |
| 16-379. QUEUE_17_A Register ..... | 2198 |
| 16-380. QUEUE_17_B Register ..... | 2199 |
| 16-381. QUEUE_17_C Register ..... | 2200 |
| 16-382. QUEUE_17_D Register ..... | 2201 |
| 16-383. QUEUE_18_A Register ..... | 2202 |
| 16-384. QUEUE_18_B Register ..... | 2203 |
| 16-385. QUEUE_18_C Register ..... | 2204 |
| 16-386. QUEUE_18_D Register ..... | 2205 |
| 16-387. QUEUE_19_A Register ..... | 2206 |
| 16-388. QUEUE_19_B Register ..... | 2207 |
| 16-389. QUEUE_19_C Register ..... | 2208 |
| 16-390. QUEUE_19_D Register ..... | 2209 |
| 16-391. QUEUE_20_A Register ..... | 2210 |
| 16-392. QUEUE_20_B Register ..... | 2211 |
| 16-393. QUEUE_20_C Register ..... | 2212 |
| 16-394. QUEUE_20_D Register ..... | 2213 |
| 16-395. QUEUE_21_A Register ..... | 2214 |

|                                   |      |
|-----------------------------------|------|
| 16-396. QUEUE_21_B Register ..... | 2215 |
| 16-397. QUEUE_21_C Register ..... | 2216 |
| 16-398. QUEUE_21_D Register ..... | 2217 |
| 16-399. QUEUE_22_A Register ..... | 2218 |
| 16-400. QUEUE_22_B Register ..... | 2219 |
| 16-401. QUEUE_22_C Register ..... | 2220 |
| 16-402. QUEUE_22_D Register ..... | 2221 |
| 16-403. QUEUE_23_A Register ..... | 2222 |
| 16-404. QUEUE_23_B Register ..... | 2223 |
| 16-405. QUEUE_23_C Register ..... | 2224 |
| 16-406. QUEUE_23_D Register ..... | 2225 |
| 16-407. QUEUE_24_A Register ..... | 2226 |
| 16-408. QUEUE_24_B Register ..... | 2227 |
| 16-409. QUEUE_24_C Register ..... | 2228 |
| 16-410. QUEUE_24_D Register ..... | 2229 |
| 16-411. QUEUE_25_A Register ..... | 2230 |
| 16-412. QUEUE_25_B Register ..... | 2231 |
| 16-413. QUEUE_25_C Register ..... | 2232 |
| 16-414. QUEUE_25_D Register ..... | 2233 |
| 16-415. QUEUE_26_A Register ..... | 2234 |
| 16-416. QUEUE_26_B Register ..... | 2235 |
| 16-417. QUEUE_26_C Register ..... | 2236 |
| 16-418. QUEUE_26_D Register ..... | 2237 |
| 16-419. QUEUE_27_A Register ..... | 2238 |
| 16-420. QUEUE_27_B Register ..... | 2239 |
| 16-421. QUEUE_27_C Register ..... | 2240 |
| 16-422. QUEUE_27_D Register ..... | 2241 |
| 16-423. QUEUE_28_A Register ..... | 2242 |
| 16-424. QUEUE_28_B Register ..... | 2243 |
| 16-425. QUEUE_28_C Register ..... | 2244 |
| 16-426. QUEUE_28_D Register ..... | 2245 |
| 16-427. QUEUE_29_A Register ..... | 2246 |
| 16-428. QUEUE_29_B Register ..... | 2247 |
| 16-429. QUEUE_29_C Register ..... | 2248 |
| 16-430. QUEUE_29_D Register ..... | 2249 |
| 16-431. QUEUE_30_A Register ..... | 2250 |
| 16-432. QUEUE_30_B Register ..... | 2251 |
| 16-433. QUEUE_30_C Register ..... | 2252 |
| 16-434. QUEUE_30_D Register ..... | 2253 |
| 16-435. QUEUE_31_A Register ..... | 2254 |
| 16-436. QUEUE_31_B Register ..... | 2255 |
| 16-437. QUEUE_31_C Register ..... | 2256 |
| 16-438. QUEUE_31_D Register ..... | 2257 |
| 16-439. QUEUE_32_A Register ..... | 2258 |
| 16-440. QUEUE_32_B Register ..... | 2259 |
| 16-441. QUEUE_32_C Register ..... | 2260 |
| 16-442. QUEUE_32_D Register ..... | 2261 |
| 16-443. QUEUE_33_A Register ..... | 2262 |
| 16-444. QUEUE_33_B Register ..... | 2263 |

---

|                                   |      |
|-----------------------------------|------|
| 16-445. QUEUE_33_C Register ..... | 2264 |
| 16-446. QUEUE_33_D Register ..... | 2265 |
| 16-447. QUEUE_34_A Register ..... | 2266 |
| 16-448. QUEUE_34_B Register ..... | 2267 |
| 16-449. QUEUE_34_C Register ..... | 2268 |
| 16-450. QUEUE_34_D Register ..... | 2269 |
| 16-451. QUEUE_35_A Register ..... | 2270 |
| 16-452. QUEUE_35_B Register ..... | 2271 |
| 16-453. QUEUE_35_C Register ..... | 2272 |
| 16-454. QUEUE_35_D Register ..... | 2273 |
| 16-455. QUEUE_36_A Register ..... | 2274 |
| 16-456. QUEUE_36_B Register ..... | 2275 |
| 16-457. QUEUE_36_C Register ..... | 2276 |
| 16-458. QUEUE_36_D Register ..... | 2277 |
| 16-459. QUEUE_37_A Register ..... | 2278 |
| 16-460. QUEUE_37_B Register ..... | 2279 |
| 16-461. QUEUE_37_C Register ..... | 2280 |
| 16-462. QUEUE_37_D Register ..... | 2281 |
| 16-463. QUEUE_38_A Register ..... | 2282 |
| 16-464. QUEUE_38_B Register ..... | 2283 |
| 16-465. QUEUE_38_C Register ..... | 2284 |
| 16-466. QUEUE_38_D Register ..... | 2285 |
| 16-467. QUEUE_39_A Register ..... | 2286 |
| 16-468. QUEUE_39_B Register ..... | 2287 |
| 16-469. QUEUE_39_C Register ..... | 2288 |
| 16-470. QUEUE_39_D Register ..... | 2289 |
| 16-471. QUEUE_40_A Register ..... | 2290 |
| 16-472. QUEUE_40_B Register ..... | 2291 |
| 16-473. QUEUE_40_C Register ..... | 2292 |
| 16-474. QUEUE_40_D Register ..... | 2293 |
| 16-475. QUEUE_41_A Register ..... | 2294 |
| 16-476. QUEUE_41_B Register ..... | 2295 |
| 16-477. QUEUE_41_C Register ..... | 2296 |
| 16-478. QUEUE_41_D Register ..... | 2297 |
| 16-479. QUEUE_42_A Register ..... | 2298 |
| 16-480. QUEUE_42_B Register ..... | 2299 |
| 16-481. QUEUE_42_C Register ..... | 2300 |
| 16-482. QUEUE_42_D Register ..... | 2301 |
| 16-483. QUEUE_43_A Register ..... | 2302 |
| 16-484. QUEUE_43_B Register ..... | 2303 |
| 16-485. QUEUE_43_C Register ..... | 2304 |
| 16-486. QUEUE_43_D Register ..... | 2305 |
| 16-487. QUEUE_44_A Register ..... | 2306 |
| 16-488. QUEUE_44_B Register ..... | 2307 |
| 16-489. QUEUE_44_C Register ..... | 2308 |
| 16-490. QUEUE_44_D Register ..... | 2309 |
| 16-491. QUEUE_45_A Register ..... | 2310 |
| 16-492. QUEUE_45_B Register ..... | 2311 |
| 16-493. QUEUE_45_C Register ..... | 2312 |

---

|                                   |      |
|-----------------------------------|------|
| 16-494. QUEUE_45_D Register ..... | 2313 |
| 16-495. QUEUE_46_A Register ..... | 2314 |
| 16-496. QUEUE_46_B Register ..... | 2315 |
| 16-497. QUEUE_46_C Register ..... | 2316 |
| 16-498. QUEUE_46_D Register ..... | 2317 |
| 16-499. QUEUE_47_A Register ..... | 2318 |
| 16-500. QUEUE_47_B Register ..... | 2319 |
| 16-501. QUEUE_47_C Register ..... | 2320 |
| 16-502. QUEUE_47_D Register ..... | 2321 |
| 16-503. QUEUE_48_A Register ..... | 2322 |
| 16-504. QUEUE_48_B Register ..... | 2323 |
| 16-505. QUEUE_48_C Register ..... | 2324 |
| 16-506. QUEUE_48_D Register ..... | 2325 |
| 16-507. QUEUE_49_A Register ..... | 2326 |
| 16-508. QUEUE_49_B Register ..... | 2327 |
| 16-509. QUEUE_49_C Register ..... | 2328 |
| 16-510. QUEUE_49_D Register ..... | 2329 |
| 16-511. QUEUE_50_A Register ..... | 2330 |
| 16-512. QUEUE_50_B Register ..... | 2331 |
| 16-513. QUEUE_50_C Register ..... | 2332 |
| 16-514. QUEUE_50_D Register ..... | 2333 |
| 16-515. QUEUE_51_A Register ..... | 2334 |
| 16-516. QUEUE_51_B Register ..... | 2335 |
| 16-517. QUEUE_51_C Register ..... | 2336 |
| 16-518. QUEUE_51_D Register ..... | 2337 |
| 16-519. QUEUE_52_A Register ..... | 2338 |
| 16-520. QUEUE_52_B Register ..... | 2339 |
| 16-521. QUEUE_52_C Register ..... | 2340 |
| 16-522. QUEUE_52_D Register ..... | 2341 |
| 16-523. QUEUE_53_A Register ..... | 2342 |
| 16-524. QUEUE_53_B Register ..... | 2343 |
| 16-525. QUEUE_53_C Register ..... | 2344 |
| 16-526. QUEUE_53_D Register ..... | 2345 |
| 16-527. QUEUE_54_A Register ..... | 2346 |
| 16-528. QUEUE_54_B Register ..... | 2347 |
| 16-529. QUEUE_54_C Register ..... | 2348 |
| 16-530. QUEUE_54_D Register ..... | 2349 |
| 16-531. QUEUE_55_A Register ..... | 2350 |
| 16-532. QUEUE_55_B Register ..... | 2351 |
| 16-533. QUEUE_55_C Register ..... | 2352 |
| 16-534. QUEUE_55_D Register ..... | 2353 |
| 16-535. QUEUE_56_A Register ..... | 2354 |
| 16-536. QUEUE_56_B Register ..... | 2355 |
| 16-537. QUEUE_56_C Register ..... | 2356 |
| 16-538. QUEUE_56_D Register ..... | 2357 |
| 16-539. QUEUE_57_A Register ..... | 2358 |
| 16-540. QUEUE_57_B Register ..... | 2359 |
| 16-541. QUEUE_57_C Register ..... | 2360 |
| 16-542. QUEUE_57_D Register ..... | 2361 |

---

|                                   |      |
|-----------------------------------|------|
| 16-543. QUEUE_58_A Register ..... | 2362 |
| 16-544. QUEUE_58_B Register ..... | 2363 |
| 16-545. QUEUE_58_C Register ..... | 2364 |
| 16-546. QUEUE_58_D Register ..... | 2365 |
| 16-547. QUEUE_59_A Register ..... | 2366 |
| 16-548. QUEUE_59_B Register ..... | 2367 |
| 16-549. QUEUE_59_C Register ..... | 2368 |
| 16-550. QUEUE_59_D Register ..... | 2369 |
| 16-551. QUEUE_60_A Register ..... | 2370 |
| 16-552. QUEUE_60_B Register ..... | 2371 |
| 16-553. QUEUE_60_C Register ..... | 2372 |
| 16-554. QUEUE_60_D Register ..... | 2373 |
| 16-555. QUEUE_61_A Register ..... | 2374 |
| 16-556. QUEUE_61_B Register ..... | 2375 |
| 16-557. QUEUE_61_C Register ..... | 2376 |
| 16-558. QUEUE_61_D Register ..... | 2377 |
| 16-559. QUEUE_62_A Register ..... | 2378 |
| 16-560. QUEUE_62_B Register ..... | 2379 |
| 16-561. QUEUE_62_C Register ..... | 2380 |
| 16-562. QUEUE_62_D Register ..... | 2381 |
| 16-563. QUEUE_63_A Register ..... | 2382 |
| 16-564. QUEUE_63_B Register ..... | 2383 |
| 16-565. QUEUE_63_C Register ..... | 2384 |
| 16-566. QUEUE_63_D Register ..... | 2385 |
| 16-567. QUEUE_64_A Register ..... | 2386 |
| 16-568. QUEUE_64_B Register ..... | 2387 |
| 16-569. QUEUE_64_C Register ..... | 2388 |
| 16-570. QUEUE_64_D Register ..... | 2389 |
| 16-571. QUEUE_65_A Register ..... | 2390 |
| 16-572. QUEUE_65_B Register ..... | 2391 |
| 16-573. QUEUE_65_C Register ..... | 2392 |
| 16-574. QUEUE_65_D Register ..... | 2393 |
| 16-575. QUEUE_66_A Register ..... | 2394 |
| 16-576. QUEUE_66_B Register ..... | 2395 |
| 16-577. QUEUE_66_C Register ..... | 2396 |
| 16-578. QUEUE_66_D Register ..... | 2397 |
| 16-579. QUEUE_67_A Register ..... | 2398 |
| 16-580. QUEUE_67_B Register ..... | 2399 |
| 16-581. QUEUE_67_C Register ..... | 2400 |
| 16-582. QUEUE_67_D Register ..... | 2401 |
| 16-583. QUEUE_68_A Register ..... | 2402 |
| 16-584. QUEUE_68_B Register ..... | 2403 |
| 16-585. QUEUE_68_C Register ..... | 2404 |
| 16-586. QUEUE_68_D Register ..... | 2405 |
| 16-587. QUEUE_69_A Register ..... | 2406 |
| 16-588. QUEUE_69_B Register ..... | 2407 |
| 16-589. QUEUE_69_C Register ..... | 2408 |
| 16-590. QUEUE_69_D Register ..... | 2409 |
| 16-591. QUEUE_70_A Register ..... | 2410 |

---

|                                   |      |
|-----------------------------------|------|
| 16-592. QUEUE_70_B Register ..... | 2411 |
| 16-593. QUEUE_70_C Register ..... | 2412 |
| 16-594. QUEUE_70_D Register ..... | 2413 |
| 16-595. QUEUE_71_A Register ..... | 2414 |
| 16-596. QUEUE_71_B Register ..... | 2415 |
| 16-597. QUEUE_71_C Register ..... | 2416 |
| 16-598. QUEUE_71_D Register ..... | 2417 |
| 16-599. QUEUE_72_A Register ..... | 2418 |
| 16-600. QUEUE_72_B Register ..... | 2419 |
| 16-601. QUEUE_72_C Register ..... | 2420 |
| 16-602. QUEUE_72_D Register ..... | 2421 |
| 16-603. QUEUE_73_A Register ..... | 2422 |
| 16-604. QUEUE_73_B Register ..... | 2423 |
| 16-605. QUEUE_73_C Register ..... | 2424 |
| 16-606. QUEUE_73_D Register ..... | 2425 |
| 16-607. QUEUE_74_A Register ..... | 2426 |
| 16-608. QUEUE_74_B Register ..... | 2427 |
| 16-609. QUEUE_74_C Register ..... | 2428 |
| 16-610. QUEUE_74_D Register ..... | 2429 |
| 16-611. QUEUE_75_A Register ..... | 2430 |
| 16-612. QUEUE_75_B Register ..... | 2431 |
| 16-613. QUEUE_75_C Register ..... | 2432 |
| 16-614. QUEUE_75_D Register ..... | 2433 |
| 16-615. QUEUE_76_A Register ..... | 2434 |
| 16-616. QUEUE_76_B Register ..... | 2435 |
| 16-617. QUEUE_76_C Register ..... | 2436 |
| 16-618. QUEUE_76_D Register ..... | 2437 |
| 16-619. QUEUE_77_A Register ..... | 2438 |
| 16-620. QUEUE_77_B Register ..... | 2439 |
| 16-621. QUEUE_77_C Register ..... | 2440 |
| 16-622. QUEUE_77_D Register ..... | 2441 |
| 16-623. QUEUE_78_A Register ..... | 2442 |
| 16-624. QUEUE_78_B Register ..... | 2443 |
| 16-625. QUEUE_78_C Register ..... | 2444 |
| 16-626. QUEUE_78_D Register ..... | 2445 |
| 16-627. QUEUE_79_A Register ..... | 2446 |
| 16-628. QUEUE_79_B Register ..... | 2447 |
| 16-629. QUEUE_79_C Register ..... | 2448 |
| 16-630. QUEUE_79_D Register ..... | 2449 |
| 16-631. QUEUE_80_A Register ..... | 2450 |
| 16-632. QUEUE_80_B Register ..... | 2451 |
| 16-633. QUEUE_80_C Register ..... | 2452 |
| 16-634. QUEUE_80_D Register ..... | 2453 |
| 16-635. QUEUE_81_A Register ..... | 2454 |
| 16-636. QUEUE_81_B Register ..... | 2455 |
| 16-637. QUEUE_81_C Register ..... | 2456 |
| 16-638. QUEUE_81_D Register ..... | 2457 |
| 16-639. QUEUE_82_A Register ..... | 2458 |
| 16-640. QUEUE_82_B Register ..... | 2459 |

---

|                                   |      |
|-----------------------------------|------|
| 16-641. QUEUE_82_C Register ..... | 2460 |
| 16-642. QUEUE_82_D Register ..... | 2461 |
| 16-643. QUEUE_83_A Register ..... | 2462 |
| 16-644. QUEUE_83_B Register ..... | 2463 |
| 16-645. QUEUE_83_C Register ..... | 2464 |
| 16-646. QUEUE_83_D Register ..... | 2465 |
| 16-647. QUEUE_84_A Register ..... | 2466 |
| 16-648. QUEUE_84_B Register ..... | 2467 |
| 16-649. QUEUE_84_C Register ..... | 2468 |
| 16-650. QUEUE_84_D Register ..... | 2469 |
| 16-651. QUEUE_85_A Register ..... | 2470 |
| 16-652. QUEUE_85_B Register ..... | 2471 |
| 16-653. QUEUE_85_C Register ..... | 2472 |
| 16-654. QUEUE_85_D Register ..... | 2473 |
| 16-655. QUEUE_86_A Register ..... | 2474 |
| 16-656. QUEUE_86_B Register ..... | 2475 |
| 16-657. QUEUE_86_C Register ..... | 2476 |
| 16-658. QUEUE_86_D Register ..... | 2477 |
| 16-659. QUEUE_87_A Register ..... | 2478 |
| 16-660. QUEUE_87_B Register ..... | 2479 |
| 16-661. QUEUE_87_C Register ..... | 2480 |
| 16-662. QUEUE_87_D Register ..... | 2481 |
| 16-663. QUEUE_88_A Register ..... | 2482 |
| 16-664. QUEUE_88_B Register ..... | 2483 |
| 16-665. QUEUE_88_C Register ..... | 2484 |
| 16-666. QUEUE_88_D Register ..... | 2485 |
| 16-667. QUEUE_89_A Register ..... | 2486 |
| 16-668. QUEUE_89_B Register ..... | 2487 |
| 16-669. QUEUE_89_C Register ..... | 2488 |
| 16-670. QUEUE_89_D Register ..... | 2489 |
| 16-671. QUEUE_90_A Register ..... | 2490 |
| 16-672. QUEUE_90_B Register ..... | 2491 |
| 16-673. QUEUE_90_C Register ..... | 2492 |
| 16-674. QUEUE_90_D Register ..... | 2493 |
| 16-675. QUEUE_91_A Register ..... | 2494 |
| 16-676. QUEUE_91_B Register ..... | 2495 |
| 16-677. QUEUE_91_C Register ..... | 2496 |
| 16-678. QUEUE_91_D Register ..... | 2497 |
| 16-679. QUEUE_92_A Register ..... | 2498 |
| 16-680. QUEUE_92_B Register ..... | 2499 |
| 16-681. QUEUE_92_C Register ..... | 2500 |
| 16-682. QUEUE_92_D Register ..... | 2501 |
| 16-683. QUEUE_93_A Register ..... | 2502 |
| 16-684. QUEUE_93_B Register ..... | 2503 |
| 16-685. QUEUE_93_C Register ..... | 2504 |
| 16-686. QUEUE_93_D Register ..... | 2505 |
| 16-687. QUEUE_94_A Register ..... | 2506 |
| 16-688. QUEUE_94_B Register ..... | 2507 |
| 16-689. QUEUE_94_C Register ..... | 2508 |

|                                    |      |
|------------------------------------|------|
| 16-690. QUEUE_94_D Register .....  | 2509 |
| 16-691. QUEUE_95_A Register .....  | 2510 |
| 16-692. QUEUE_95_B Register .....  | 2511 |
| 16-693. QUEUE_95_C Register .....  | 2512 |
| 16-694. QUEUE_95_D Register .....  | 2513 |
| 16-695. QUEUE_96_A Register .....  | 2514 |
| 16-696. QUEUE_96_B Register .....  | 2515 |
| 16-697. QUEUE_96_C Register .....  | 2516 |
| 16-698. QUEUE_96_D Register .....  | 2517 |
| 16-699. QUEUE_97_A Register .....  | 2518 |
| 16-700. QUEUE_97_B Register .....  | 2519 |
| 16-701. QUEUE_97_C Register .....  | 2520 |
| 16-702. QUEUE_97_D Register .....  | 2521 |
| 16-703. QUEUE_98_A Register .....  | 2522 |
| 16-704. QUEUE_98_B Register .....  | 2523 |
| 16-705. QUEUE_98_C Register .....  | 2524 |
| 16-706. QUEUE_98_D Register .....  | 2525 |
| 16-707. QUEUE_99_A Register .....  | 2526 |
| 16-708. QUEUE_99_B Register .....  | 2527 |
| 16-709. QUEUE_99_C Register .....  | 2528 |
| 16-710. QUEUE_99_D Register .....  | 2529 |
| 16-711. QUEUE_100_A Register ..... | 2530 |
| 16-712. QUEUE_100_B Register ..... | 2531 |
| 16-713. QUEUE_100_C Register ..... | 2532 |
| 16-714. QUEUE_100_D Register ..... | 2533 |
| 16-715. QUEUE_101_A Register ..... | 2534 |
| 16-716. QUEUE_101_B Register ..... | 2535 |
| 16-717. QUEUE_101_C Register ..... | 2536 |
| 16-718. QUEUE_101_D Register ..... | 2537 |
| 16-719. QUEUE_102_A Register ..... | 2538 |
| 16-720. QUEUE_102_B Register ..... | 2539 |
| 16-721. QUEUE_102_C Register ..... | 2540 |
| 16-722. QUEUE_102_D Register ..... | 2541 |
| 16-723. QUEUE_103_A Register ..... | 2542 |
| 16-724. QUEUE_103_B Register ..... | 2543 |
| 16-725. QUEUE_103_C Register ..... | 2544 |
| 16-726. QUEUE_103_D Register ..... | 2545 |
| 16-727. QUEUE_104_A Register ..... | 2546 |
| 16-728. QUEUE_104_B Register ..... | 2547 |
| 16-729. QUEUE_104_C Register ..... | 2548 |
| 16-730. QUEUE_104_D Register ..... | 2549 |
| 16-731. QUEUE_105_A Register ..... | 2550 |
| 16-732. QUEUE_105_B Register ..... | 2551 |
| 16-733. QUEUE_105_C Register ..... | 2552 |
| 16-734. QUEUE_105_D Register ..... | 2553 |
| 16-735. QUEUE_106_A Register ..... | 2554 |
| 16-736. QUEUE_106_B Register ..... | 2555 |
| 16-737. QUEUE_106_C Register ..... | 2556 |
| 16-738. QUEUE_106_D Register ..... | 2557 |

---

|                                    |      |
|------------------------------------|------|
| 16-739. QUEUE_107_A Register ..... | 2558 |
| 16-740. QUEUE_107_B Register ..... | 2559 |
| 16-741. QUEUE_107_C Register.....  | 2560 |
| 16-742. QUEUE_107_D Register.....  | 2561 |
| 16-743. QUEUE_108_A Register ..... | 2562 |
| 16-744. QUEUE_108_B Register ..... | 2563 |
| 16-745. QUEUE_108_C Register.....  | 2564 |
| 16-746. QUEUE_108_D Register.....  | 2565 |
| 16-747. QUEUE_109_A Register ..... | 2566 |
| 16-748. QUEUE_109_B Register ..... | 2567 |
| 16-749. QUEUE_109_C Register.....  | 2568 |
| 16-750. QUEUE_109_D Register.....  | 2569 |
| 16-751. QUEUE_110_A Register ..... | 2570 |
| 16-752. QUEUE_110_B Register ..... | 2571 |
| 16-753. QUEUE_110_C Register.....  | 2572 |
| 16-754. QUEUE_110_D Register.....  | 2573 |
| 16-755. QUEUE_111_A Register ..... | 2574 |
| 16-756. QUEUE_111_B Register ..... | 2575 |
| 16-757. QUEUE_111_C Register.....  | 2576 |
| 16-758. QUEUE_111_D Register.....  | 2577 |
| 16-759. QUEUE_112_A Register ..... | 2578 |
| 16-760. QUEUE_112_B Register ..... | 2579 |
| 16-761. QUEUE_112_C Register.....  | 2580 |
| 16-762. QUEUE_112_D Register.....  | 2581 |
| 16-763. QUEUE_113_A Register ..... | 2582 |
| 16-764. QUEUE_113_B Register ..... | 2583 |
| 16-765. QUEUE_113_C Register.....  | 2584 |
| 16-766. QUEUE_113_D Register.....  | 2585 |
| 16-767. QUEUE_114_A Register ..... | 2586 |
| 16-768. QUEUE_114_B Register ..... | 2587 |
| 16-769. QUEUE_114_C Register.....  | 2588 |
| 16-770. QUEUE_114_D Register.....  | 2589 |
| 16-771. QUEUE_115_A Register ..... | 2590 |
| 16-772. QUEUE_115_B Register ..... | 2591 |
| 16-773. QUEUE_115_C Register.....  | 2592 |
| 16-774. QUEUE_115_D Register.....  | 2593 |
| 16-775. QUEUE_116_A Register ..... | 2594 |
| 16-776. QUEUE_116_B Register ..... | 2595 |
| 16-777. QUEUE_116_C Register.....  | 2596 |
| 16-778. QUEUE_116_D Register.....  | 2597 |
| 16-779. QUEUE_117_A Register ..... | 2598 |
| 16-780. QUEUE_117_B Register ..... | 2599 |
| 16-781. QUEUE_117_C Register.....  | 2600 |
| 16-782. QUEUE_117_D Register.....  | 2601 |
| 16-783. QUEUE_118_A Register ..... | 2602 |
| 16-784. QUEUE_118_B Register ..... | 2603 |
| 16-785. QUEUE_118_C Register.....  | 2604 |
| 16-786. QUEUE_118_D Register.....  | 2605 |
| 16-787. QUEUE_119_A Register ..... | 2606 |

|                                    |      |
|------------------------------------|------|
| 16-788. QUEUE_119_B Register ..... | 2607 |
| 16-789. QUEUE_119_C Register ..... | 2608 |
| 16-790. QUEUE_119_D Register ..... | 2609 |
| 16-791. QUEUE_120_A Register ..... | 2610 |
| 16-792. QUEUE_120_B Register ..... | 2611 |
| 16-793. QUEUE_120_C Register ..... | 2612 |
| 16-794. QUEUE_120_D Register ..... | 2613 |
| 16-795. QUEUE_121_A Register ..... | 2614 |
| 16-796. QUEUE_121_B Register ..... | 2615 |
| 16-797. QUEUE_121_C Register ..... | 2616 |
| 16-798. QUEUE_121_D Register ..... | 2617 |
| 16-799. QUEUE_122_A Register ..... | 2618 |
| 16-800. QUEUE_122_B Register ..... | 2619 |
| 16-801. QUEUE_122_C Register ..... | 2620 |
| 16-802. QUEUE_122_D Register ..... | 2621 |
| 16-803. QUEUE_123_A Register ..... | 2622 |
| 16-804. QUEUE_123_B Register ..... | 2623 |
| 16-805. QUEUE_123_C Register ..... | 2624 |
| 16-806. QUEUE_123_D Register ..... | 2625 |
| 16-807. QUEUE_124_A Register ..... | 2626 |
| 16-808. QUEUE_124_B Register ..... | 2627 |
| 16-809. QUEUE_124_C Register ..... | 2628 |
| 16-810. QUEUE_124_D Register ..... | 2629 |
| 16-811. QUEUE_125_A Register ..... | 2630 |
| 16-812. QUEUE_125_B Register ..... | 2631 |
| 16-813. QUEUE_125_C Register ..... | 2632 |
| 16-814. QUEUE_125_D Register ..... | 2633 |
| 16-815. QUEUE_126_A Register ..... | 2634 |
| 16-816. QUEUE_126_B Register ..... | 2635 |
| 16-817. QUEUE_126_C Register ..... | 2636 |
| 16-818. QUEUE_126_D Register ..... | 2637 |
| 16-819. QUEUE_127_A Register ..... | 2638 |
| 16-820. QUEUE_127_B Register ..... | 2639 |
| 16-821. QUEUE_127_C Register ..... | 2640 |
| 16-822. QUEUE_127_D Register ..... | 2641 |
| 16-823. QUEUE_128_A Register ..... | 2642 |
| 16-824. QUEUE_128_B Register ..... | 2643 |
| 16-825. QUEUE_128_C Register ..... | 2644 |
| 16-826. QUEUE_128_D Register ..... | 2645 |
| 16-827. QUEUE_129_A Register ..... | 2646 |
| 16-828. QUEUE_129_B Register ..... | 2647 |
| 16-829. QUEUE_129_C Register ..... | 2648 |
| 16-830. QUEUE_129_D Register ..... | 2649 |
| 16-831. QUEUE_130_A Register ..... | 2650 |
| 16-832. QUEUE_130_B Register ..... | 2651 |
| 16-833. QUEUE_130_C Register ..... | 2652 |
| 16-834. QUEUE_130_D Register ..... | 2653 |
| 16-835. QUEUE_131_A Register ..... | 2654 |
| 16-836. QUEUE_131_B Register ..... | 2655 |

---

|                                   |      |
|-----------------------------------|------|
| 16-837. QUEUE_131_C Register..... | 2656 |
| 16-838. QUEUE_131_D Register..... | 2657 |
| 16-839. QUEUE_132_A Register..... | 2658 |
| 16-840. QUEUE_132_B Register..... | 2659 |
| 16-841. QUEUE_132_C Register..... | 2660 |
| 16-842. QUEUE_132_D Register..... | 2661 |
| 16-843. QUEUE_133_A Register..... | 2662 |
| 16-844. QUEUE_133_B Register..... | 2663 |
| 16-845. QUEUE_133_C Register..... | 2664 |
| 16-846. QUEUE_133_D Register..... | 2665 |
| 16-847. QUEUE_134_A Register..... | 2666 |
| 16-848. QUEUE_134_B Register..... | 2667 |
| 16-849. QUEUE_134_C Register..... | 2668 |
| 16-850. QUEUE_134_D Register..... | 2669 |
| 16-851. QUEUE_135_A Register..... | 2670 |
| 16-852. QUEUE_135_B Register..... | 2671 |
| 16-853. QUEUE_135_C Register..... | 2672 |
| 16-854. QUEUE_135_D Register..... | 2673 |
| 16-855. QUEUE_136_A Register..... | 2674 |
| 16-856. QUEUE_136_B Register..... | 2675 |
| 16-857. QUEUE_136_C Register..... | 2676 |
| 16-858. QUEUE_136_D Register..... | 2677 |
| 16-859. QUEUE_137_A Register..... | 2678 |
| 16-860. QUEUE_137_B Register..... | 2679 |
| 16-861. QUEUE_137_C Register..... | 2680 |
| 16-862. QUEUE_137_D Register..... | 2681 |
| 16-863. QUEUE_138_A Register..... | 2682 |
| 16-864. QUEUE_138_B Register..... | 2683 |
| 16-865. QUEUE_138_C Register..... | 2684 |
| 16-866. QUEUE_138_D Register..... | 2685 |
| 16-867. QUEUE_139_A Register..... | 2686 |
| 16-868. QUEUE_139_B Register..... | 2687 |
| 16-869. QUEUE_139_C Register..... | 2688 |
| 16-870. QUEUE_139_D Register..... | 2689 |
| 16-871. QUEUE_140_A Register..... | 2690 |
| 16-872. QUEUE_140_B Register..... | 2691 |
| 16-873. QUEUE_140_C Register..... | 2692 |
| 16-874. QUEUE_140_D Register..... | 2693 |
| 16-875. QUEUE_141_A Register..... | 2694 |
| 16-876. QUEUE_141_B Register..... | 2695 |
| 16-877. QUEUE_141_C Register..... | 2696 |
| 16-878. QUEUE_141_D Register..... | 2697 |
| 16-879. QUEUE_142_A Register..... | 2698 |
| 16-880. QUEUE_142_B Register..... | 2699 |
| 16-881. QUEUE_142_C Register..... | 2700 |
| 16-882. QUEUE_142_D Register..... | 2701 |
| 16-883. QUEUE_143_A Register..... | 2702 |
| 16-884. QUEUE_143_B Register..... | 2703 |
| 16-885. QUEUE_143_C Register..... | 2704 |

|                                   |      |
|-----------------------------------|------|
| 16-886. QUEUE_143_D Register..... | 2705 |
| 16-887. QUEUE_144_A Register..... | 2706 |
| 16-888. QUEUE_144_B Register..... | 2707 |
| 16-889. QUEUE_144_C Register..... | 2708 |
| 16-890. QUEUE_144_D Register..... | 2709 |
| 16-891. QUEUE_145_A Register..... | 2710 |
| 16-892. QUEUE_145_B Register..... | 2711 |
| 16-893. QUEUE_145_C Register..... | 2712 |
| 16-894. QUEUE_145_D Register..... | 2713 |
| 16-895. QUEUE_146_A Register..... | 2714 |
| 16-896. QUEUE_146_B Register..... | 2715 |
| 16-897. QUEUE_146_C Register..... | 2716 |
| 16-898. QUEUE_146_D Register..... | 2717 |
| 16-899. QUEUE_147_A Register..... | 2718 |
| 16-900. QUEUE_147_B Register..... | 2719 |
| 16-901. QUEUE_147_C Register..... | 2720 |
| 16-902. QUEUE_147_D Register..... | 2721 |
| 16-903. QUEUE_148_A Register..... | 2722 |
| 16-904. QUEUE_148_B Register..... | 2723 |
| 16-905. QUEUE_148_C Register..... | 2724 |
| 16-906. QUEUE_148_D Register..... | 2725 |
| 16-907. QUEUE_149_A Register..... | 2726 |
| 16-908. QUEUE_149_B Register..... | 2727 |
| 16-909. QUEUE_149_C Register..... | 2728 |
| 16-910. QUEUE_149_D Register..... | 2729 |
| 16-911. QUEUE_150_A Register..... | 2730 |
| 16-912. QUEUE_150_B Register..... | 2731 |
| 16-913. QUEUE_150_C Register..... | 2732 |
| 16-914. QUEUE_150_D Register..... | 2733 |
| 16-915. QUEUE_151_A Register..... | 2734 |
| 16-916. QUEUE_151_B Register..... | 2735 |
| 16-917. QUEUE_151_C Register..... | 2736 |
| 16-918. QUEUE_151_D Register..... | 2737 |
| 16-919. QUEUE_152_A Register..... | 2738 |
| 16-920. QUEUE_152_B Register..... | 2739 |
| 16-921. QUEUE_152_C Register..... | 2740 |
| 16-922. QUEUE_152_D Register..... | 2741 |
| 16-923. QUEUE_153_A Register..... | 2742 |
| 16-924. QUEUE_153_B Register..... | 2743 |
| 16-925. QUEUE_153_C Register..... | 2744 |
| 16-926. QUEUE_153_D Register..... | 2745 |
| 16-927. QUEUE_154_A Register..... | 2746 |
| 16-928. QUEUE_154_B Register..... | 2747 |
| 16-929. QUEUE_154_C Register..... | 2748 |
| 16-930. QUEUE_154_D Register..... | 2749 |
| 16-931. QUEUE_155_A Register..... | 2750 |
| 16-932. QUEUE_155_B Register..... | 2751 |
| 16-933. QUEUE_155_C Register..... | 2752 |
| 16-934. QUEUE_155_D Register..... | 2753 |

---

|   |      |
|---|------|
| 16-935. QUEUE_0_STATUS_A Register.....  | 2754 |
| 16-936. QUEUE_0_STATUS_B Register.....  | 2755 |
| 16-937. QUEUE_0_STATUS_C Register.....  | 2756 |
| 16-938. QUEUE_1_STATUS_A Register.....  | 2757 |
| 16-939. QUEUE_1_STATUS_B Register.....  | 2758 |
| 16-940. QUEUE_1_STATUS_C Register.....  | 2759 |
| 16-941. QUEUE_2_STATUS_A Register.....  | 2760 |
| 16-942. QUEUE_2_STATUS_B Register.....  | 2761 |
| 16-943. QUEUE_2_STATUS_C Register.....  | 2762 |
| 16-944. QUEUE_3_STATUS_A Register.....  | 2763 |
| 16-945. QUEUE_3_STATUS_B Register.....  | 2764 |
| 16-946. QUEUE_3_STATUS_C Register.....  | 2765 |
| 16-947. QUEUE_4_STATUS_A Register.....  | 2766 |
| 16-948. QUEUE_4_STATUS_B Register.....  | 2767 |
| 16-949. QUEUE_4_STATUS_C Register.....  | 2768 |
| 16-950. QUEUE_5_STATUS_A Register.....  | 2769 |
| 16-951. QUEUE_5_STATUS_B Register.....  | 2770 |
| 16-952. QUEUE_5_STATUS_C Register.....  | 2771 |
| 16-953. QUEUE_6_STATUS_A Register.....  | 2772 |
| 16-954. QUEUE_6_STATUS_B Register.....  | 2773 |
| 16-955. QUEUE_6_STATUS_C Register.....  | 2774 |
| 16-956. QUEUE_7_STATUS_A Register.....  | 2775 |
| 16-957. QUEUE_7_STATUS_B Register.....  | 2776 |
| 16-958. QUEUE_7_STATUS_C Register.....  | 2777 |
| 16-959. QUEUE_8_STATUS_A Register.....  | 2778 |
| 16-960. QUEUE_8_STATUS_B Register.....  | 2779 |
| 16-961. QUEUE_8_STATUS_C Register.....  | 2780 |
| 16-962. QUEUE_9_STATUS_A Register.....  | 2781 |
| 16-963. QUEUE_9_STATUS_B Register.....  | 2782 |
| 16-964. QUEUE_9_STATUS_C Register.....  | 2783 |
| 16-965. QUEUE_10_STATUS_A Register..... | 2784 |
| 16-966. QUEUE_10_STATUS_B Register..... | 2785 |
| 16-967. QUEUE_10_STATUS_C Register..... | 2786 |
| 16-968. QUEUE_11_STATUS_A Register..... | 2787 |
| 16-969. QUEUE_11_STATUS_B Register..... | 2788 |
| 16-970. QUEUE_11_STATUS_C Register..... | 2789 |
| 16-971. QUEUE_12_STATUS_A Register..... | 2790 |
| 16-972. QUEUE_12_STATUS_B Register..... | 2791 |
| 16-973. QUEUE_12_STATUS_C Register..... | 2792 |
| 16-974. QUEUE_13_STATUS_A Register..... | 2793 |
| 16-975. QUEUE_13_STATUS_B Register..... | 2794 |
| 16-976. QUEUE_13_STATUS_C Register..... | 2795 |
| 16-977. QUEUE_14_STATUS_A Register..... | 2796 |
| 16-978. QUEUE_14_STATUS_B Register..... | 2797 |
| 16-979. QUEUE_14_STATUS_C Register..... | 2798 |
| 16-980. QUEUE_15_STATUS_A Register..... | 2799 |
| 16-981. QUEUE_15_STATUS_B Register..... | 2800 |
| 16-982. QUEUE_15_STATUS_C Register..... | 2801 |
| 16-983. QUEUE_16_STATUS_A Register..... | 2802 |

|   |      |
|---|------|
| 16-984. QUEUE_16_STATUS_B Register .....  | 2803 |
| 16-985. QUEUE_16_STATUS_C Register .....  | 2804 |
| 16-986. QUEUE_17_STATUS_A Register .....  | 2805 |
| 16-987. QUEUE_17_STATUS_B Register .....  | 2806 |
| 16-988. QUEUE_17_STATUS_C Register .....  | 2807 |
| 16-989. QUEUE_18_STATUS_A Register .....  | 2808 |
| 16-990. QUEUE_18_STATUS_B Register .....  | 2809 |
| 16-991. QUEUE_18_STATUS_C Register .....  | 2810 |
| 16-992. QUEUE_19_STATUS_A Register .....  | 2811 |
| 16-993. QUEUE_19_STATUS_B Register .....  | 2812 |
| 16-994. QUEUE_19_STATUS_C Register .....  | 2813 |
| 16-995. QUEUE_20_STATUS_A Register .....  | 2814 |
| 16-996. QUEUE_20_STATUS_B Register .....  | 2815 |
| 16-997. QUEUE_20_STATUS_C Register .....  | 2816 |
| 16-998. QUEUE_21_STATUS_A Register .....  | 2817 |
| 16-999. QUEUE_21_STATUS_B Register .....  | 2818 |
| 16-1000. QUEUE_21_STATUS_C Register ..... | 2819 |
| 16-1001. QUEUE_22_STATUS_A Register ..... | 2820 |
| 16-1002. QUEUE_22_STATUS_B Register ..... | 2821 |
| 16-1003. QUEUE_22_STATUS_C Register ..... | 2822 |
| 16-1004. QUEUE_23_STATUS_A Register ..... | 2823 |
| 16-1005. QUEUE_23_STATUS_B Register ..... | 2824 |
| 16-1006. QUEUE_23_STATUS_C Register ..... | 2825 |
| 16-1007. QUEUE_24_STATUS_A Register ..... | 2826 |
| 16-1008. QUEUE_24_STATUS_B Register ..... | 2827 |
| 16-1009. QUEUE_24_STATUS_C Register ..... | 2828 |
| 16-1010. QUEUE_25_STATUS_A Register ..... | 2829 |
| 16-1011. QUEUE_25_STATUS_B Register ..... | 2830 |
| 16-1012. QUEUE_25_STATUS_C Register ..... | 2831 |
| 16-1013. QUEUE_26_STATUS_A Register ..... | 2832 |
| 16-1014. QUEUE_26_STATUS_B Register ..... | 2833 |
| 16-1015. QUEUE_26_STATUS_C Register ..... | 2834 |
| 16-1016. QUEUE_27_STATUS_A Register ..... | 2835 |
| 16-1017. QUEUE_27_STATUS_B Register ..... | 2836 |
| 16-1018. QUEUE_27_STATUS_C Register ..... | 2837 |
| 16-1019. QUEUE_28_STATUS_A Register ..... | 2838 |
| 16-1020. QUEUE_28_STATUS_B Register ..... | 2839 |
| 16-1021. QUEUE_28_STATUS_C Register ..... | 2840 |
| 16-1022. QUEUE_29_STATUS_A Register ..... | 2841 |
| 16-1023. QUEUE_29_STATUS_B Register ..... | 2842 |
| 16-1024. QUEUE_29_STATUS_C Register ..... | 2843 |
| 16-1025. QUEUE_30_STATUS_A Register ..... | 2844 |
| 16-1026. QUEUE_30_STATUS_B Register ..... | 2845 |
| 16-1027. QUEUE_30_STATUS_C Register ..... | 2846 |
| 16-1028. QUEUE_31_STATUS_A Register ..... | 2847 |
| 16-1029. QUEUE_31_STATUS_B Register ..... | 2848 |
| 16-1030. QUEUE_31_STATUS_C Register ..... | 2849 |
| 16-1031. QUEUE_32_STATUS_A Register ..... | 2850 |
| 16-1032. QUEUE_32_STATUS_B Register ..... | 2851 |

---

|  |      |
|--|------|
| 16-1033. QUEUE_32_STATUS_C Register..... | 2852 |
| 16-1034. QUEUE_33_STATUS_A Register..... | 2853 |
| 16-1035. QUEUE_33_STATUS_B Register..... | 2854 |
| 16-1036. QUEUE_33_STATUS_C Register..... | 2855 |
| 16-1037. QUEUE_34_STATUS_A Register..... | 2856 |
| 16-1038. QUEUE_34_STATUS_B Register..... | 2857 |
| 16-1039. QUEUE_34_STATUS_C Register..... | 2858 |
| 16-1040. QUEUE_35_STATUS_A Register..... | 2859 |
| 16-1041. QUEUE_35_STATUS_B Register..... | 2860 |
| 16-1042. QUEUE_35_STATUS_C Register..... | 2861 |
| 16-1043. QUEUE_36_STATUS_A Register..... | 2862 |
| 16-1044. QUEUE_36_STATUS_B Register..... | 2863 |
| 16-1045. QUEUE_36_STATUS_C Register..... | 2864 |
| 16-1046. QUEUE_37_STATUS_A Register..... | 2865 |
| 16-1047. QUEUE_37_STATUS_B Register..... | 2866 |
| 16-1048. QUEUE_37_STATUS_C Register..... | 2867 |
| 16-1049. QUEUE_38_STATUS_A Register..... | 2868 |
| 16-1050. QUEUE_38_STATUS_B Register..... | 2869 |
| 16-1051. QUEUE_38_STATUS_C Register..... | 2870 |
| 16-1052. QUEUE_39_STATUS_A Register..... | 2871 |
| 16-1053. QUEUE_39_STATUS_B Register..... | 2872 |
| 16-1054. QUEUE_39_STATUS_C Register..... | 2873 |
| 16-1055. QUEUE_40_STATUS_A Register..... | 2874 |
| 16-1056. QUEUE_40_STATUS_B Register..... | 2875 |
| 16-1057. QUEUE_40_STATUS_C Register..... | 2876 |
| 16-1058. QUEUE_41_STATUS_A Register..... | 2877 |
| 16-1059. QUEUE_41_STATUS_B Register..... | 2878 |
| 16-1060. QUEUE_41_STATUS_C Register..... | 2879 |
| 16-1061. QUEUE_42_STATUS_A Register..... | 2880 |
| 16-1062. QUEUE_42_STATUS_B Register..... | 2881 |
| 16-1063. QUEUE_42_STATUS_C Register..... | 2882 |
| 16-1064. QUEUE_43_STATUS_A Register..... | 2883 |
| 16-1065. QUEUE_43_STATUS_B Register..... | 2884 |
| 16-1066. QUEUE_43_STATUS_C Register..... | 2885 |
| 16-1067. QUEUE_44_STATUS_A Register..... | 2886 |
| 16-1068. QUEUE_44_STATUS_B Register..... | 2887 |
| 16-1069. QUEUE_44_STATUS_C Register..... | 2888 |
| 16-1070. QUEUE_45_STATUS_A Register..... | 2889 |
| 16-1071. QUEUE_45_STATUS_B Register..... | 2890 |
| 16-1072. QUEUE_45_STATUS_C Register..... | 2891 |
| 16-1073. QUEUE_46_STATUS_A Register..... | 2892 |
| 16-1074. QUEUE_46_STATUS_B Register..... | 2893 |
| 16-1075. QUEUE_46_STATUS_C Register..... | 2894 |
| 16-1076. QUEUE_47_STATUS_A Register..... | 2895 |
| 16-1077. QUEUE_47_STATUS_B Register..... | 2896 |
| 16-1078. QUEUE_47_STATUS_C Register..... | 2897 |
| 16-1079. QUEUE_48_STATUS_A Register..... | 2898 |
| 16-1080. QUEUE_48_STATUS_B Register..... | 2899 |
| 16-1081. QUEUE_48_STATUS_C Register..... | 2900 |

|  |      |
|--|------|
| 16-1082. QUEUE_49_STATUS_A Register..... | 2901 |
| 16-1083. QUEUE_49_STATUS_B Register..... | 2902 |
| 16-1084. QUEUE_49_STATUS_C Register..... | 2903 |
| 16-1085. QUEUE_50_STATUS_A Register..... | 2904 |
| 16-1086. QUEUE_50_STATUS_B Register..... | 2905 |
| 16-1087. QUEUE_50_STATUS_C Register..... | 2906 |
| 16-1088. QUEUE_51_STATUS_A Register..... | 2907 |
| 16-1089. QUEUE_51_STATUS_B Register..... | 2908 |
| 16-1090. QUEUE_51_STATUS_C Register..... | 2909 |
| 16-1091. QUEUE_52_STATUS_A Register..... | 2910 |
| 16-1092. QUEUE_52_STATUS_B Register..... | 2911 |
| 16-1093. QUEUE_52_STATUS_C Register..... | 2912 |
| 16-1094. QUEUE_53_STATUS_A Register..... | 2913 |
| 16-1095. QUEUE_53_STATUS_B Register..... | 2914 |
| 16-1096. QUEUE_53_STATUS_C Register..... | 2915 |
| 16-1097. QUEUE_54_STATUS_A Register..... | 2916 |
| 16-1098. QUEUE_54_STATUS_B Register..... | 2917 |
| 16-1099. QUEUE_54_STATUS_C Register..... | 2918 |
| 16-1100. QUEUE_55_STATUS_A Register..... | 2919 |
| 16-1101. QUEUE_55_STATUS_B Register..... | 2920 |
| 16-1102. QUEUE_55_STATUS_C Register..... | 2921 |
| 16-1103. QUEUE_56_STATUS_A Register..... | 2922 |
| 16-1104. QUEUE_56_STATUS_B Register..... | 2923 |
| 16-1105. QUEUE_56_STATUS_C Register..... | 2924 |
| 16-1106. QUEUE_57_STATUS_A Register..... | 2925 |
| 16-1107. QUEUE_57_STATUS_B Register..... | 2926 |
| 16-1108. QUEUE_57_STATUS_C Register..... | 2927 |
| 16-1109. QUEUE_58_STATUS_A Register..... | 2928 |
| 16-1110. QUEUE_58_STATUS_B Register..... | 2929 |
| 16-1111. QUEUE_58_STATUS_C Register..... | 2930 |
| 16-1112. QUEUE_59_STATUS_A Register..... | 2931 |
| 16-1113. QUEUE_59_STATUS_B Register..... | 2932 |
| 16-1114. QUEUE_59_STATUS_C Register..... | 2933 |
| 16-1115. QUEUE_60_STATUS_A Register..... | 2934 |
| 16-1116. QUEUE_60_STATUS_B Register..... | 2935 |
| 16-1117. QUEUE_60_STATUS_C Register..... | 2936 |
| 16-1118. QUEUE_61_STATUS_A Register..... | 2937 |
| 16-1119. QUEUE_61_STATUS_B Register..... | 2938 |
| 16-1120. QUEUE_61_STATUS_C Register..... | 2939 |
| 16-1121. QUEUE_62_STATUS_A Register..... | 2940 |
| 16-1122. QUEUE_62_STATUS_B Register..... | 2941 |
| 16-1123. QUEUE_62_STATUS_C Register..... | 2942 |
| 16-1124. QUEUE_63_STATUS_A Register..... | 2943 |
| 16-1125. QUEUE_63_STATUS_B Register..... | 2944 |
| 16-1126. QUEUE_63_STATUS_C Register..... | 2945 |
| 16-1127. QUEUE_64_STATUS_A Register..... | 2946 |
| 16-1128. QUEUE_64_STATUS_B Register..... | 2947 |
| 16-1129. QUEUE_64_STATUS_C Register..... | 2948 |
| 16-1130. QUEUE_65_STATUS_A Register..... | 2949 |

---

|  |      |
|--|------|
| 16-1131. QUEUE_65_STATUS_B Register..... | 2950 |
| 16-1132. QUEUE_65_STATUS_C Register..... | 2951 |
| 16-1133. QUEUE_66_STATUS_A Register..... | 2952 |
| 16-1134. QUEUE_66_STATUS_B Register..... | 2953 |
| 16-1135. QUEUE_66_STATUS_C Register..... | 2954 |
| 16-1136. QUEUE_67_STATUS_A Register..... | 2955 |
| 16-1137. QUEUE_67_STATUS_B Register..... | 2956 |
| 16-1138. QUEUE_67_STATUS_C Register..... | 2957 |
| 16-1139. QUEUE_68_STATUS_A Register..... | 2958 |
| 16-1140. QUEUE_68_STATUS_B Register..... | 2959 |
| 16-1141. QUEUE_68_STATUS_C Register..... | 2960 |
| 16-1142. QUEUE_69_STATUS_A Register..... | 2961 |
| 16-1143. QUEUE_69_STATUS_B Register..... | 2962 |
| 16-1144. QUEUE_69_STATUS_C Register..... | 2963 |
| 16-1145. QUEUE_70_STATUS_A Register..... | 2964 |
| 16-1146. QUEUE_70_STATUS_B Register..... | 2965 |
| 16-1147. QUEUE_70_STATUS_C Register..... | 2966 |
| 16-1148. QUEUE_71_STATUS_A Register..... | 2967 |
| 16-1149. QUEUE_71_STATUS_B Register..... | 2968 |
| 16-1150. QUEUE_71_STATUS_C Register..... | 2969 |
| 16-1151. QUEUE_72_STATUS_A Register..... | 2970 |
| 16-1152. QUEUE_72_STATUS_B Register..... | 2971 |
| 16-1153. QUEUE_72_STATUS_C Register..... | 2972 |
| 16-1154. QUEUE_73_STATUS_A Register..... | 2973 |
| 16-1155. QUEUE_73_STATUS_B Register..... | 2974 |
| 16-1156. QUEUE_73_STATUS_C Register..... | 2975 |
| 16-1157. QUEUE_74_STATUS_A Register..... | 2976 |
| 16-1158. QUEUE_74_STATUS_B Register..... | 2977 |
| 16-1159. QUEUE_74_STATUS_C Register..... | 2978 |
| 16-1160. QUEUE_75_STATUS_A Register..... | 2979 |
| 16-1161. QUEUE_75_STATUS_B Register..... | 2980 |
| 16-1162. QUEUE_75_STATUS_C Register..... | 2981 |
| 16-1163. QUEUE_76_STATUS_A Register..... | 2982 |
| 16-1164. QUEUE_76_STATUS_B Register..... | 2983 |
| 16-1165. QUEUE_76_STATUS_C Register..... | 2984 |
| 16-1166. QUEUE_77_STATUS_A Register..... | 2985 |
| 16-1167. QUEUE_77_STATUS_B Register..... | 2986 |
| 16-1168. QUEUE_77_STATUS_C Register..... | 2987 |
| 16-1169. QUEUE_78_STATUS_A Register..... | 2988 |
| 16-1170. QUEUE_78_STATUS_B Register..... | 2989 |
| 16-1171. QUEUE_78_STATUS_C Register..... | 2990 |
| 16-1172. QUEUE_79_STATUS_A Register..... | 2991 |
| 16-1173. QUEUE_79_STATUS_B Register..... | 2992 |
| 16-1174. QUEUE_79_STATUS_C Register..... | 2993 |
| 16-1175. QUEUE_80_STATUS_A Register..... | 2994 |
| 16-1176. QUEUE_80_STATUS_B Register..... | 2995 |
| 16-1177. QUEUE_80_STATUS_C Register..... | 2996 |
| 16-1178. QUEUE_81_STATUS_A Register..... | 2997 |
| 16-1179. QUEUE_81_STATUS_B Register..... | 2998 |

|  |      |
|--|------|
| 16-1180. QUEUE_81_STATUS_C Register..... | 2999 |
| 16-1181. QUEUE_82_STATUS_A Register..... | 3000 |
| 16-1182. QUEUE_82_STATUS_B Register..... | 3001 |
| 16-1183. QUEUE_82_STATUS_C Register..... | 3002 |
| 16-1184. QUEUE_83_STATUS_A Register..... | 3003 |
| 16-1185. QUEUE_83_STATUS_B Register..... | 3004 |
| 16-1186. QUEUE_83_STATUS_C Register..... | 3005 |
| 16-1187. QUEUE_84_STATUS_A Register..... | 3006 |
| 16-1188. QUEUE_84_STATUS_B Register..... | 3007 |
| 16-1189. QUEUE_84_STATUS_C Register..... | 3008 |
| 16-1190. QUEUE_85_STATUS_A Register..... | 3009 |
| 16-1191. QUEUE_85_STATUS_B Register..... | 3010 |
| 16-1192. QUEUE_85_STATUS_C Register..... | 3011 |
| 16-1193. QUEUE_86_STATUS_A Register..... | 3012 |
| 16-1194. QUEUE_86_STATUS_B Register..... | 3013 |
| 16-1195. QUEUE_86_STATUS_C Register..... | 3014 |
| 16-1196. QUEUE_87_STATUS_A Register..... | 3015 |
| 16-1197. QUEUE_87_STATUS_B Register..... | 3016 |
| 16-1198. QUEUE_87_STATUS_C Register..... | 3017 |
| 16-1199. QUEUE_88_STATUS_A Register..... | 3018 |
| 16-1200. QUEUE_88_STATUS_B Register..... | 3019 |
| 16-1201. QUEUE_88_STATUS_C Register..... | 3020 |
| 16-1202. QUEUE_89_STATUS_A Register..... | 3021 |
| 16-1203. QUEUE_89_STATUS_B Register..... | 3022 |
| 16-1204. QUEUE_89_STATUS_C Register..... | 3023 |
| 16-1205. QUEUE_90_STATUS_A Register..... | 3024 |
| 16-1206. QUEUE_90_STATUS_B Register..... | 3025 |
| 16-1207. QUEUE_90_STATUS_C Register..... | 3026 |
| 16-1208. QUEUE_91_STATUS_A Register..... | 3027 |
| 16-1209. QUEUE_91_STATUS_B Register..... | 3028 |
| 16-1210. QUEUE_91_STATUS_C Register..... | 3029 |
| 16-1211. QUEUE_92_STATUS_A Register..... | 3030 |
| 16-1212. QUEUE_92_STATUS_B Register..... | 3031 |
| 16-1213. QUEUE_92_STATUS_C Register..... | 3032 |
| 16-1214. QUEUE_93_STATUS_A Register..... | 3033 |
| 16-1215. QUEUE_93_STATUS_B Register..... | 3034 |
| 16-1216. QUEUE_93_STATUS_C Register..... | 3035 |
| 16-1217. QUEUE_94_STATUS_A Register..... | 3036 |
| 16-1218. QUEUE_94_STATUS_B Register..... | 3037 |
| 16-1219. QUEUE_94_STATUS_C Register..... | 3038 |
| 16-1220. QUEUE_95_STATUS_A Register..... | 3039 |
| 16-1221. QUEUE_95_STATUS_B Register..... | 3040 |
| 16-1222. QUEUE_95_STATUS_C Register..... | 3041 |
| 16-1223. QUEUE_96_STATUS_A Register..... | 3042 |
| 16-1224. QUEUE_96_STATUS_B Register..... | 3043 |
| 16-1225. QUEUE_96_STATUS_C Register..... | 3044 |
| 16-1226. QUEUE_97_STATUS_A Register..... | 3045 |
| 16-1227. QUEUE_97_STATUS_B Register..... | 3046 |
| 16-1228. QUEUE_97_STATUS_C Register..... | 3047 |

|  |      |
|--|------|
| 16-1229. QUEUE_98_STATUS_A Register.....   | 3048 |
| 16-1230. QUEUE_98_STATUS_B Register.....   | 3049 |
| 16-1231. QUEUE_98_STATUS_C Register.....   | 3050 |
| 16-1232. QUEUE_99_STATUS_A Register.....   | 3051 |
| 16-1233. QUEUE_99_STATUS_B Register.....   | 3052 |
| 16-1234. QUEUE_99_STATUS_C Register.....   | 3053 |
| 16-1235. QUEUE_100_STATUS_A Register ..... | 3054 |
| 16-1236. QUEUE_100_STATUS_B Register ..... | 3055 |
| 16-1237. QUEUE_100_STATUS_C Register ..... | 3056 |
| 16-1238. QUEUE_101_STATUS_A Register ..... | 3057 |
| 16-1239. QUEUE_101_STATUS_B Register ..... | 3058 |
| 16-1240. QUEUE_101_STATUS_C Register ..... | 3059 |
| 16-1241. QUEUE_102_STATUS_A Register ..... | 3060 |
| 16-1242. QUEUE_102_STATUS_B Register ..... | 3061 |
| 16-1243. QUEUE_102_STATUS_C Register ..... | 3062 |
| 16-1244. QUEUE_103_STATUS_A Register ..... | 3063 |
| 16-1245. QUEUE_103_STATUS_B Register ..... | 3064 |
| 16-1246. QUEUE_103_STATUS_C Register ..... | 3065 |
| 16-1247. QUEUE_104_STATUS_A Register ..... | 3066 |
| 16-1248. QUEUE_104_STATUS_B Register ..... | 3067 |
| 16-1249. QUEUE_104_STATUS_C Register ..... | 3068 |
| 16-1250. QUEUE_105_STATUS_A Register ..... | 3069 |
| 16-1251. QUEUE_105_STATUS_B Register ..... | 3070 |
| 16-1252. QUEUE_105_STATUS_C Register ..... | 3071 |
| 16-1253. QUEUE_106_STATUS_A Register ..... | 3072 |
| 16-1254. QUEUE_106_STATUS_B Register ..... | 3073 |
| 16-1255. QUEUE_106_STATUS_C Register ..... | 3074 |
| 16-1256. QUEUE_107_STATUS_A Register ..... | 3075 |
| 16-1257. QUEUE_107_STATUS_B Register ..... | 3076 |
| 16-1258. QUEUE_107_STATUS_C Register ..... | 3077 |
| 16-1259. QUEUE_108_STATUS_A Register ..... | 3078 |
| 16-1260. QUEUE_108_STATUS_B Register ..... | 3079 |
| 16-1261. QUEUE_108_STATUS_C Register ..... | 3080 |
| 16-1262. QUEUE_109_STATUS_A Register ..... | 3081 |
| 16-1263. QUEUE_109_STATUS_B Register ..... | 3082 |
| 16-1264. QUEUE_109_STATUS_C Register ..... | 3083 |
| 16-1265. QUEUE_110_STATUS_A Register ..... | 3084 |
| 16-1266. QUEUE_110_STATUS_B Register ..... | 3085 |
| 16-1267. QUEUE_110_STATUS_C Register ..... | 3086 |
| 16-1268. QUEUE_111_STATUS_A Register ..... | 3087 |
| 16-1269. QUEUE_111_STATUS_B Register ..... | 3088 |
| 16-1270. QUEUE_111_STATUS_C Register ..... | 3089 |
| 16-1271. QUEUE_112_STATUS_A Register ..... | 3090 |
| 16-1272. QUEUE_112_STATUS_B Register ..... | 3091 |
| 16-1273. QUEUE_112_STATUS_C Register ..... | 3092 |
| 16-1274. QUEUE_113_STATUS_A Register ..... | 3093 |
| 16-1275. QUEUE_113_STATUS_B Register ..... | 3094 |
| 16-1276. QUEUE_113_STATUS_C Register ..... | 3095 |
| 16-1277. QUEUE_114_STATUS_A Register ..... | 3096 |

|  |      |
|--|------|
| 16-1278. QUEUE_114_STATUS_B Register ..... | 3097 |
| 16-1279. QUEUE_114_STATUS_C Register ..... | 3098 |
| 16-1280. QUEUE_115_STATUS_A Register ..... | 3099 |
| 16-1281. QUEUE_115_STATUS_B Register ..... | 3100 |
| 16-1282. QUEUE_115_STATUS_C Register ..... | 3101 |
| 16-1283. QUEUE_116_STATUS_A Register ..... | 3102 |
| 16-1284. QUEUE_116_STATUS_B Register ..... | 3103 |
| 16-1285. QUEUE_116_STATUS_C Register ..... | 3104 |
| 16-1286. QUEUE_117_STATUS_A Register ..... | 3105 |
| 16-1287. QUEUE_117_STATUS_B Register ..... | 3106 |
| 16-1288. QUEUE_117_STATUS_C Register ..... | 3107 |
| 16-1289. QUEUE_118_STATUS_A Register ..... | 3108 |
| 16-1290. QUEUE_118_STATUS_B Register ..... | 3109 |
| 16-1291. QUEUE_118_STATUS_C Register ..... | 3110 |
| 16-1292. QUEUE_119_STATUS_A Register ..... | 3111 |
| 16-1293. QUEUE_119_STATUS_B Register ..... | 3112 |
| 16-1294. QUEUE_119_STATUS_C Register ..... | 3113 |
| 16-1295. QUEUE_120_STATUS_A Register ..... | 3114 |
| 16-1296. QUEUE_120_STATUS_B Register ..... | 3115 |
| 16-1297. QUEUE_120_STATUS_C Register ..... | 3116 |
| 16-1298. QUEUE_121_STATUS_A Register ..... | 3117 |
| 16-1299. QUEUE_121_STATUS_B Register ..... | 3118 |
| 16-1300. QUEUE_121_STATUS_C Register ..... | 3119 |
| 16-1301. QUEUE_122_STATUS_A Register ..... | 3120 |
| 16-1302. QUEUE_122_STATUS_B Register ..... | 3121 |
| 16-1303. QUEUE_122_STATUS_C Register ..... | 3122 |
| 16-1304. QUEUE_123_STATUS_A Register ..... | 3123 |
| 16-1305. QUEUE_123_STATUS_B Register ..... | 3124 |
| 16-1306. QUEUE_123_STATUS_C Register ..... | 3125 |
| 16-1307. QUEUE_124_STATUS_A Register ..... | 3126 |
| 16-1308. QUEUE_124_STATUS_B Register ..... | 3127 |
| 16-1309. QUEUE_124_STATUS_C Register ..... | 3128 |
| 16-1310. QUEUE_125_STATUS_A Register ..... | 3129 |
| 16-1311. QUEUE_125_STATUS_B Register ..... | 3130 |
| 16-1312. QUEUE_125_STATUS_C Register ..... | 3131 |
| 16-1313. QUEUE_126_STATUS_A Register ..... | 3132 |
| 16-1314. QUEUE_126_STATUS_B Register ..... | 3133 |
| 16-1315. QUEUE_126_STATUS_C Register ..... | 3134 |
| 16-1316. QUEUE_127_STATUS_A Register ..... | 3135 |
| 16-1317. QUEUE_127_STATUS_B Register ..... | 3136 |
| 16-1318. QUEUE_127_STATUS_C Register ..... | 3137 |
| 16-1319. QUEUE_128_STATUS_A Register ..... | 3138 |
| 16-1320. QUEUE_128_STATUS_B Register ..... | 3139 |
| 16-1321. QUEUE_128_STATUS_C Register ..... | 3140 |
| 16-1322. QUEUE_129_STATUS_A Register ..... | 3141 |
| 16-1323. QUEUE_129_STATUS_B Register ..... | 3142 |
| 16-1324. QUEUE_129_STATUS_C Register ..... | 3143 |
| 16-1325. QUEUE_130_STATUS_A Register ..... | 3144 |
| 16-1326. QUEUE_130_STATUS_B Register ..... | 3145 |

|  |      |
|--|------|
| 16-1327. QUEUE_130_STATUS_C Register ..... | 3146 |
| 16-1328. QUEUE_131_STATUS_A Register ..... | 3147 |
| 16-1329. QUEUE_131_STATUS_B Register ..... | 3148 |
| 16-1330. QUEUE_131_STATUS_C Register ..... | 3149 |
| 16-1331. QUEUE_132_STATUS_A Register ..... | 3150 |
| 16-1332. QUEUE_132_STATUS_B Register ..... | 3151 |
| 16-1333. QUEUE_132_STATUS_C Register ..... | 3152 |
| 16-1334. QUEUE_133_STATUS_A Register ..... | 3153 |
| 16-1335. QUEUE_133_STATUS_B Register ..... | 3154 |
| 16-1336. QUEUE_133_STATUS_C Register ..... | 3155 |
| 16-1337. QUEUE_134_STATUS_A Register ..... | 3156 |
| 16-1338. QUEUE_134_STATUS_B Register ..... | 3157 |
| 16-1339. QUEUE_134_STATUS_C Register ..... | 3158 |
| 16-1340. QUEUE_135_STATUS_A Register ..... | 3159 |
| 16-1341. QUEUE_135_STATUS_B Register ..... | 3160 |
| 16-1342. QUEUE_135_STATUS_C Register ..... | 3161 |
| 16-1343. QUEUE_136_STATUS_A Register ..... | 3162 |
| 16-1344. QUEUE_136_STATUS_B Register ..... | 3163 |
| 16-1345. QUEUE_136_STATUS_C Register ..... | 3164 |
| 16-1346. QUEUE_137_STATUS_A Register ..... | 3165 |
| 16-1347. QUEUE_137_STATUS_B Register ..... | 3166 |
| 16-1348. QUEUE_137_STATUS_C Register ..... | 3167 |
| 16-1349. QUEUE_138_STATUS_A Register ..... | 3168 |
| 16-1350. QUEUE_138_STATUS_B Register ..... | 3169 |
| 16-1351. QUEUE_138_STATUS_C Register ..... | 3170 |
| 16-1352. QUEUE_139_STATUS_A Register ..... | 3171 |
| 16-1353. QUEUE_139_STATUS_B Register ..... | 3172 |
| 16-1354. QUEUE_139_STATUS_C Register ..... | 3173 |
| 16-1355. QUEUE_140_STATUS_A Register ..... | 3174 |
| 16-1356. QUEUE_140_STATUS_B Register ..... | 3175 |
| 16-1357. QUEUE_140_STATUS_C Register ..... | 3176 |
| 16-1358. QUEUE_141_STATUS_A Register ..... | 3177 |
| 16-1359. QUEUE_141_STATUS_B Register ..... | 3178 |
| 16-1360. QUEUE_141_STATUS_C Register ..... | 3179 |
| 16-1361. QUEUE_142_STATUS_A Register ..... | 3180 |
| 16-1362. QUEUE_142_STATUS_B Register ..... | 3181 |
| 16-1363. QUEUE_142_STATUS_C Register ..... | 3182 |
| 16-1364. QUEUE_143_STATUS_A Register ..... | 3183 |
| 16-1365. QUEUE_143_STATUS_B Register ..... | 3184 |
| 16-1366. QUEUE_143_STATUS_C Register ..... | 3185 |
| 16-1367. QUEUE_144_STATUS_A Register ..... | 3186 |
| 16-1368. QUEUE_144_STATUS_B Register ..... | 3187 |
| 16-1369. QUEUE_144_STATUS_C Register ..... | 3188 |
| 16-1370. QUEUE_145_STATUS_A Register ..... | 3189 |
| 16-1371. QUEUE_145_STATUS_B Register ..... | 3190 |
| 16-1372. QUEUE_145_STATUS_C Register ..... | 3191 |
| 16-1373. QUEUE_146_STATUS_A Register ..... | 3192 |
| 16-1374. QUEUE_146_STATUS_B Register ..... | 3193 |
| 16-1375. QUEUE_146_STATUS_C Register ..... | 3194 |

|  |      |
|--|------|
| 16-1376. QUEUE_147_STATUS_A Register ..... | 3195 |
| 16-1377. QUEUE_147_STATUS_B Register ..... | 3196 |
| 16-1378. QUEUE_147_STATUS_C Register ..... | 3197 |
| 16-1379. QUEUE_148_STATUS_A Register ..... | 3198 |
| 16-1380. QUEUE_148_STATUS_B Register ..... | 3199 |
| 16-1381. QUEUE_148_STATUS_C Register ..... | 3200 |
| 16-1382. QUEUE_149_STATUS_A Register ..... | 3201 |
| 16-1383. QUEUE_149_STATUS_B Register ..... | 3202 |
| 16-1384. QUEUE_149_STATUS_C Register ..... | 3203 |
| 16-1385. QUEUE_150_STATUS_A Register ..... | 3204 |
| 16-1386. QUEUE_150_STATUS_B Register ..... | 3205 |
| 16-1387. QUEUE_150_STATUS_C Register ..... | 3206 |
| 16-1388. QUEUE_151_STATUS_A Register ..... | 3207 |
| 16-1389. QUEUE_151_STATUS_B Register ..... | 3208 |
| 16-1390. QUEUE_151_STATUS_C Register ..... | 3209 |
| 16-1391. QUEUE_152_STATUS_A Register ..... | 3210 |
| 16-1392. QUEUE_152_STATUS_B Register ..... | 3211 |
| 16-1393. QUEUE_152_STATUS_C Register ..... | 3212 |
| 16-1394. QUEUE_153_STATUS_A Register ..... | 3213 |
| 16-1395. QUEUE_153_STATUS_B Register ..... | 3214 |
| 16-1396. QUEUE_153_STATUS_C Register ..... | 3215 |
| 16-1397. QUEUE_154_STATUS_A Register ..... | 3216 |
| 16-1398. QUEUE_154_STATUS_B Register ..... | 3217 |
| 16-1399. QUEUE_154_STATUS_C Register ..... | 3218 |
| 16-1400. QUEUE_155_STATUS_A Register ..... | 3219 |
| 16-1401. QUEUE_155_STATUS_B Register ..... | 3220 |
| 16-1402. QUEUE_155_STATUS_C Register ..... | 3221 |
| 17-1. Mailbox Integration .....            | 3224 |
| 17-2. Mailbox Block Diagram .....          | 3226 |
| 17-3. REVISION Register .....              | 3235 |
| 17-4. SYSCONFIG Register .....             | 3236 |
| 17-5. MESSAGE_0 Register .....             | 3237 |
| 17-6. MESSAGE_1 Register .....             | 3238 |
| 17-7. MESSAGE_2 Register .....             | 3239 |
| 17-8. MESSAGE_3 Register .....             | 3240 |
| 17-9. MESSAGE_4 Register .....             | 3241 |
| 17-10. MESSAGE_5 Register .....            | 3242 |
| 17-11. MESSAGE_6 Register .....            | 3243 |
| 17-12. MESSAGE_7 Register .....            | 3244 |
| 17-13. FIFOSTATUS_0 Register .....         | 3245 |
| 17-14. FIFOSTATUS_1 Register .....         | 3246 |
| 17-15. FIFOSTATUS_2 Register .....         | 3247 |
| 17-16. FIFOSTATUS_3 Register .....         | 3248 |
| 17-17. FIFOSTATUS_4 Register .....         | 3249 |
| 17-18. FIFOSTATUS_5 Register .....         | 3250 |
| 17-19. FIFOSTATUS_6 Register .....         | 3251 |
| 17-20. FIFOSTATUS_7 Register .....         | 3252 |
| 17-21. MSGSTATUS_0 Register .....          | 3253 |
| 17-22. MSGSTATUS_1 Register .....          | 3254 |

|                                      |      |
|--------------------------------------|------|
| 17-23. MSGSTATUS_2 Register .....    | 3255 |
| 17-24. MSGSTATUS_3 Register .....    | 3256 |
| 17-25. MSGSTATUS_4 Register .....    | 3257 |
| 17-26. MSGSTATUS_5 Register .....    | 3258 |
| 17-27. MSGSTATUS_6 Register .....    | 3259 |
| 17-28. MSGSTATUS_7 Register .....    | 3260 |
| 17-29. IRQSTATUS_RAW_0 Register..... | 3261 |
| 17-30. IRQSTATUS_CLR_0 Register..... | 3263 |
| 17-31. IRQENABLE_SET_0 Register..... | 3265 |
| 17-32. IRQENABLE_CLR_0 Register..... | 3267 |
| 17-33. IRQSTATUS_RAW_1 Register..... | 3269 |
| 17-34. IRQSTATUS_CLR_1 Register..... | 3271 |
| 17-35. IRQENABLE_SET_1 Register..... | 3273 |
| 17-36. IRQENABLE_CLR_1 Register..... | 3275 |
| 17-37. IRQSTATUS_RAW_2 Register..... | 3277 |
| 17-38. IRQSTATUS_CLR_2 Register..... | 3279 |
| 17-39. IRQENABLE_SET_2 Register..... | 3281 |
| 17-40. IRQENABLE_CLR_2 Register..... | 3283 |
| 17-41. IRQSTATUS_RAW_3 Register..... | 3285 |
| 17-42. IRQSTATUS_CLR_3 Register..... | 3287 |
| 17-43. IRQENABLE_SET_3 Register..... | 3289 |
| 17-44. IRQENABLE_CLR_3 Register..... | 3291 |
| 17-45. REV Register.....             | 3296 |
| 17-46. SYSCONFIG Register .....      | 3297 |
| 17-47. SYSTATUS Register.....        | 3298 |
| 17-48. LOCK_REG_0 Register .....     | 3299 |
| 17-49. LOCK_REG_1 Register .....     | 3300 |
| 17-50. LOCK_REG_2 Register .....     | 3301 |
| 17-51. LOCK_REG_3 Register .....     | 3302 |
| 17-52. LOCK_REG_4 Register .....     | 3303 |
| 17-53. LOCK_REG_5 Register .....     | 3304 |
| 17-54. LOCK_REG_6 Register .....     | 3305 |
| 17-55. LOCK_REG_7 Register .....     | 3306 |
| 17-56. LOCK_REG_8 Register .....     | 3307 |
| 17-57. LOCK_REG_9 Register .....     | 3308 |
| 17-58. LOCK_REG_10 Register.....     | 3309 |
| 17-59. LOCK_REG_11 Register.....     | 3310 |
| 17-60. LOCK_REG_12 Register.....     | 3311 |
| 17-61. LOCK_REG_13 Register.....     | 3312 |
| 17-62. LOCK_REG_14 Register.....     | 3313 |
| 17-63. LOCK_REG_15 Register.....     | 3314 |
| 17-64. LOCK_REG_16 Register .....    | 3315 |
| 17-65. LOCK_REG_17 Register .....    | 3316 |
| 17-66. LOCK_REG_18 Register .....    | 3317 |
| 17-67. LOCK_REG_19 Register .....    | 3318 |
| 17-68. LOCK_REG_20 Register .....    | 3319 |
| 17-69. LOCK_REG_21 Register .....    | 3320 |
| 17-70. LOCK_REG_22 Register .....    | 3321 |
| 17-71. LOCK_REG_23 Register .....    | 3322 |

|  |      |
|--|------|
| 17-72. LOCK_REG_24 Register .....  | 3323 |
| 17-73. LOCK_REG_25 Register .....  | 3324 |
| 17-74. LOCK_REG_26 Register .....  | 3325 |
| 17-75. LOCK_REG_27 Register .....  | 3326 |
| 17-76. LOCK_REG_28 Register .....  | 3327 |
| 17-77. LOCK_REG_29 Register .....  | 3328 |
| 17-78. LOCK_REG_30 Register .....  | 3329 |
| 17-79. LOCK_REG_31 Register .....  | 3330 |
| 18-1. MMCHS Module SDIO Application .....                                      | 3333 |
| 18-2. MMCHS SD (4-bit) Card Application .....                                  | 3333 |
| 18-3. MMCHS Module MMC Application .....                                       | 3334 |
| 18-4. MMC/SD1/2 Connectivity to an MMC/SD Card .....                           | 3337 |
| 18-5. MMC/SD0 Connectivity to an MMC/SD Card .....                             | 3337 |
| 18-6. Sequential Read Operation (MMC Cards Only) .....                         | 3340 |
| 18-7. Sequential Write Operation (MMC Cards Only) .....                        | 3340 |
| 18-8. Multiple Block Read Operation (MMC Cards Only) .....                     | 3341 |
| 18-9. Multiple Block Write Operation (MMC Cards Only) .....                    | 3341 |
| 18-10. Command Token Format .....  | 3342 |
| 18-11. 48-Bit Response Packet (R1, R3, R4, R5, R6) .....                       | 3342 |
| 18-12. 136-Bit Response Packet (R2) .....                                      | 3342 |
| 18-13. Data Packet for Sequential Transfer (1-Bit) .....                       | 3343 |
| 18-14. Data Packet for Block Transfer (1-Bit) .....                            | 3343 |
| 18-15. Data Packet for Block Transfer (4-Bit) .....                            | 3343 |
| 18-16. Data Packet for Block Transfer (8-Bit) .....                            | 3344 |
| 18-17. DMA Receive Mode .....  | 3351 |
| 18-18. DMA Transmit Mode .....   | 3352 |
| 18-19. Buffer Management for a Write .....                                     | 3354 |
| 18-20. Buffer Management for a Read .....                                      | 3355 |
| 18-21. Busy Timeout for R1b, R5b Responses .....                               | 3358 |
| 18-22. Busy Timeout After Write CRC Status .....                               | 3358 |
| 18-23. Write CRC Status Timeout .....  | 3359 |
| 18-24. Read Data Timeout .....   | 3359 |
| 18-25. Boot Acknowledge Timeout When Using CMD0 .....                          | 3360 |
| 18-26. Boot Acknowledge Timeout When CMD Held Low .....                        | 3360 |
| 18-27. Auto CMD12 Timing During Write Transfer .....                           | 3362 |
| 18-28. Auto Command 12 Timings During Read Transfer .....                      | 3363 |
| 18-29. Output Driven on Falling Edge .....                                     | 3365 |
| 18-30. Output Driven on Rising Edge .....                                      | 3366 |
| 18-31. Boot Mode With CMD0 .....   | 3367 |
| 18-32. Boot Mode With CMD Line Tied to 0 .....                                 | 3368 |
| 18-33. MMC/SD/SDIO Controller Software Reset Flow .....                        | 3372 |
| 18-34. MMC/SD/SDIO Controller Bus Configuration Flow .....                     | 3373 |
| 18-35. MMC/SD/SDIO Controller Card Identification and Selection - Part 1 ..... | 3374 |
| 18-36. MMC/SD/SDIO Controller Card Identification and Selection - Part 2 ..... | 3375 |
| 18-37. SD_SYS CONFIG Register .....  | 3377 |
| 18-38. SD_SYS STATUS Register .....  | 3379 |
| 18-39. SD_CSRE Register .....  | 3380 |
| 18-40. SD_SYSTEST Register .....   | 3381 |
| 18-41. SD_CON Register .....   | 3385 |

|  |      |
|--|------|
| 18-42. SD_PWCNT Register .....   | 3389 |
| 18-43. SD_SDMASA Register .....  | 3390 |
| 18-44. SD_BLK Register .....   | 3391 |
| 18-45. SD_ARG Register .....   | 3392 |
| 18-46. SD_CMD Register .....   | 3393 |
| 18-47. SD_RSP10 Register .....   | 3398 |
| 18-48. SD_RSP32 Register .....   | 3399 |
| 18-49. SD_RSP54 Register .....   | 3400 |
| 18-50. SD_RSP76 Register .....   | 3401 |
| 18-51. SD_DATA Register .....  | 3402 |
| 18-52. SD_PSTATE Register .....  | 3403 |
| 18-53. SD_HCTL Register .....  | 3406 |
| 18-54. SD_SYSCTL Register .....  | 3409 |
| 18-55. SD_STAT Register .....  | 3411 |
| 18-56. SD_IE Register .....  | 3416 |
| 18-57. SD_ISE Register .....   | 3419 |
| 18-58. SD_AC12 Register .....  | 3422 |
| 18-59. SD_CAPA Register .....  | 3424 |
| 18-60. SD_CUR_CAPA Register .....  | 3426 |
| 18-61. SD_FE Register .....  | 3427 |
| 18-62. SD_ADMAES Register .....  | 3429 |
| 18-63. SD ADMASAL Register .....   | 3430 |
| 18-64. SD ADMASAH Register .....   | 3431 |
| 18-65. SD_REV Register .....   | 3432 |
| 19-1. UART/IrDA Module — UART Application .....  | 3436 |
| 19-2. UART/IrDA Module — IrDA/CIR Application .....  | 3436 |
| 19-3. UART/IrDA/CIR Functional Specification Block Diagram .....   | 3441 |
| 19-4. FIFO Management Registers .....  | 3446 |
| 19-5. RX FIFO Interrupt Request Generation .....   | 3448 |
| 19-6. TX FIFO Interrupt Request Generation .....   | 3449 |
| 19-7. Receive FIFO DMA Request Generation (32 Characters) .....  | 3450 |
| 19-8. Transmit FIFO DMA Request Generation (56 Spaces) .....   | 3451 |
| 19-9. Transmit FIFO DMA Request Generation (8 Spaces) .....  | 3452 |
| 19-10. Transmit FIFO DMA Request Generation (1 Space) .....  | 3452 |
| 19-11. Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming. (Threshold = 3; Spaces = 8) ..... | 3453 |
| 19-12. DMA Transmission .....  | 3453 |
| 19-13. DMA Reception .....   | 3454 |
| 19-14. UART Data Format .....  | 3461 |
| 19-15. Baud Rate Generation .....  | 3461 |
| 19-16. IrDA SIR Frame Format .....   | 3467 |
| 19-17. IrDA Encoding Mechanism .....   | 3468 |
| 19-18. IrDA Decoding Mechanism .....   | 3469 |
| 19-19. SIR Free Format Mode .....  | 3470 |
| 19-20. MIR Transmit Frame Format .....   | 3470 |
| 19-21. MIR BAUD Rate Adjustment Mechanism .....  | 3471 |
| 19-22. SIP Pulse .....   | 3471 |
| 19-23. FIR Transmit Frame Format .....   | 3471 |
| 19-24. Baud Rate Generator .....   | 3472 |

|   |      |
|---|------|
| 19-25. RC-5 Bit Encoding .....                            | 3476 |
| 19-26. SIRC Bit Encoding .....                            | 3477 |
| 19-27. RC-5 Standard Packet Format .....                  | 3477 |
| 19-28. SIRC Packet Format .....                           | 3477 |
| 19-29. SIRC Bit Transmission Example .....                | 3478 |
| 19-30. CIR Mode Block Components .....                    | 3478 |
| 19-31. CIR Pulse Modulation .....                         | 3480 |
| 19-32. CIR Modulation Duty Cycle .....                    | 3480 |
| 19-33. Variable Pulse Duration Definitions .....          | 3482 |
| 19-34. Receiver Holding Register (RHR) .....              | 3494 |
| 19-35. Transmit Holding Register (THR) .....              | 3494 |
| 19-36. UART Interrupt Enable Register (IER) .....         | 3495 |
| 19-37. IrDA Interrupt Enable Register (IER) .....         | 3496 |
| 19-38. CIR Interrupt Enable Register (IER) .....          | 3497 |
| 19-39. UART Interrupt Identification Register (IIR) ..... | 3498 |
| 19-40. IrDA Interrupt Identification Register (IIR) ..... | 3499 |
| 19-41. CIR Interrupt Identification Register (IIR) .....  | 3500 |
| 19-42. FIFO Control Register (FCR) .....                  | 3501 |
| 19-43. Line Control Register (LCR) .....                  | 3502 |
| 19-44. Modem Control Register (MCR) .....                 | 3503 |
| 19-45. UART Line Status Register (LSR) .....              | 3504 |
| 19-46. IrDA Line Status Register (LSR) .....              | 3505 |
| 19-47. CIR Line Status Register (LSR) .....               | 3506 |
| 19-48. Modem Status Register (MSR) .....                  | 3507 |
| 19-49. Transmission Control Register (TCR) .....          | 3508 |
| 19-50. Scratchpad Register (SPR) .....                    | 3508 |
| 19-51. Trigger Level Register (TLR) .....                 | 3509 |
| 19-52. Mode Definition Register 1 (MDR1) .....            | 3510 |
| 19-53. Mode Definition Register 2 (MDR2) .....            | 3511 |
| 19-54. Status FIFO Line Status Register (SFLSR) .....     | 3512 |
| 19-55. RESUME Register .....                              | 3512 |
| 19-56. Status FIFO Register Low (SFREGL) .....            | 3513 |
| 19-57. Status FIFO Register High (SFREGH) .....           | 3513 |
| 19-58. BOF Control Register (BLR) .....                   | 3514 |
| 19-59. Auxiliary Control Register (ACREG) .....           | 3515 |
| 19-60. Supplementary Control Register (SCR) .....         | 3516 |
| 19-61. Supplementary Status Register (SSR) .....          | 3517 |
| 19-62. BOF Length Register (EBLR) .....                   | 3518 |
| 19-63. Module Version Register (MVR) .....                | 3519 |
| 19-64. System Configuration Register (SYSC) .....         | 3520 |
| 19-65. System Status Register (SYSS) .....                | 3520 |
| 19-66. Wake-Up Enable Register (WER) .....                | 3521 |
| 19-67. Carrier Frequency Prescaler Register (CFPS) .....  | 3522 |
| 19-68. Divisor Latches Low Register (DLL) .....           | 3523 |
| 19-69. Divisor Latches High Register (DLH) .....          | 3523 |
| 19-70. Enhanced Feature Register (EFR) .....              | 3524 |
| 19-71. XON1/ADDR1 Register .....                          | 3525 |
| 19-72. XON2/ADDR2 Register .....                          | 3525 |
| 19-73. XOFF1 Register .....                               | 3526 |

|  |      |
|--|------|
| 19-74. XOFF2 Register .....  | 3526 |
| 19-75. Transmit Frame Length Low Register (TXFLL) .....              | 3527 |
| 19-76. Transmit Frame Length High Register (TXFLH) .....             | 3527 |
| 19-77. Received Frame Length Low Register (RXFLL) .....              | 3528 |
| 19-78. Received Frame Length High Register (RXFLH) .....             | 3528 |
| 19-79. UART Autobauding Status Register (UASR) .....                 | 3529 |
| 19-80. RXFIFO_LVL Register .....                                     | 3530 |
| 19-81. TXFIFO_LVL Register .....                                     | 3531 |
| 19-82. IER2 Register .....   | 3532 |
| 19-83. ISR2 Register .....   | 3533 |
| 19-84. FREQ_SEL Register .....                                       | 3534 |
| 19-85. Mode Definition Register 3 (MDR3) Register .....              | 3535 |
| 19-86. TX_DMA_THRESHOLD Register .....                               | 3536 |
| 20-1. Timer Block Diagram .....                                      | 3539 |
| 20-2. TCRR Timing Value .....  | 3542 |
| 20-3. Capture Wave Example for CAPT_MODE = 0 .....                   | 3543 |
| 20-4. Capture Wave Example for CAPT_MODE = 1 .....                   | 3544 |
| 20-5. Timing Diagram of Pulse-Width Modulation with SCPWM = 0 .....  | 3545 |
| 20-6. Timing Diagram of Pulse-Width Modulation with SCPWM = 1 .....  | 3546 |
| 20-7. Identification Register Register .....                         | 3552 |
| 20-8. Timer OCP Configuration Register Register .....                | 3553 |
| 20-9. Timer IRQSTATUS Raw Register Register .....                    | 3554 |
| 20-10. Timer IRQSTATUS Register Register .....                       | 3555 |
| 20-11. Timer IRQENABLE Set Register Register .....                   | 3556 |
| 20-12. Timer IRQENABLE Clear Register Register .....                 | 3557 |
| 20-13. Timer IRQ Wakeup Enable Register Register .....               | 3558 |
| 20-14. Timer Control Register Register .....                         | 3559 |
| 20-15. Timer Counter Register Register .....                         | 3561 |
| 20-16. Timer Load Register Register .....                            | 3562 |
| 20-17. Timer Trigger Register Register .....                         | 3563 |
| 20-18. Timer Write Posted Status Register Register .....             | 3564 |
| 20-19. Timer Match Register Register .....                           | 3565 |
| 20-20. Timer Capture Register Register .....                         | 3566 |
| 20-21. Timer Synchronous Interface Control Register Register .....   | 3567 |
| 20-22. Timer Capture Register Register .....                         | 3568 |
| 20-23. Block Diagram .....   | 3570 |
| 20-24. DMTimer 1 ms Integration .....                                | 3571 |
| 20-25. TCRR Timing Value .....                                       | 3573 |
| 20-26. 1ms Module Block Diagram .....                                | 3574 |
| 20-27. Capture Wave Example for CAPT_MODE 0 .....                    | 3576 |
| 20-28. Capture Wave Example for CAPT_MODE 1 .....                    | 3576 |
| 20-29. Timing Diagram of Pulse-Width Modulation, SCPWM Bit = 0 ..... | 3578 |
| 20-30. Timing Diagram of Pulse-Width Modulation, SCPWM Bit = 1 ..... | 3578 |
| 20-31. Wake-up Request Generation .....                              | 3580 |
| 20-32. TIDR Register .....   | 3583 |
| 20-33. TIOCP_CFG Register .....                                      | 3584 |
| 20-34. TISTAT Register .....   | 3585 |
| 20-35. TISR Register .....   | 3586 |
| 20-36. TIER Register .....   | 3587 |

|   |      |
|---|------|
| 20-37. TWER Register .....                          | 3588 |
| 20-38. TCLR Register .....                          | 3589 |
| 20-39. TCRR Register.....                           | 3591 |
| 20-40. TLDR Register .....                          | 3592 |
| 20-41. TTGR Register.....                           | 3593 |
| 20-42. TWPS Register .....                          | 3594 |
| 20-43. TMAR Register .....                          | 3596 |
| 20-44. TCAR1 Register .....                         | 3597 |
| 20-45. TSICR Register .....                         | 3598 |
| 20-46. TCAR2 Register .....                         | 3599 |
| 20-47. TPIR Register .....                          | 3600 |
| 20-48. TNIR Register.....                           | 3601 |
| 20-49. TCVR Register.....                           | 3602 |
| 20-50. TOCR Register .....                          | 3603 |
| 20-51. TOWR Register .....                          | 3604 |
| 20-52. RTC Block Diagram .....                      | 3607 |
| 20-53. RTC Functional Block Diagram.....            | 3607 |
| 20-54. Kick Register State Machine Diagram .....    | 3610 |
| 20-55. Flow Control for Updating RTC Registers..... | 3612 |
| 20-56. Compensation Illustration .....              | 3613 |
| 20-57. SECONDS_REG Register .....                   | 3617 |
| 20-58. MINUTES_REG Register.....                    | 3618 |
| 20-59. HOURS_REG Register .....                     | 3619 |
| 20-60. DAYS_REG Register .....                      | 3620 |
| 20-61. MONTHS_REG Register .....                    | 3621 |
| 20-62. YEARS_REG Register.....                      | 3622 |
| 20-63. WEEKS_REG Register .....                     | 3623 |
| 20-64. ALARM_SECONDS_REG Register.....              | 3624 |
| 20-65. ALARM_MINUTES_REG Register .....             | 3625 |
| 20-66. ALARM_HOURS_REG Register .....               | 3626 |
| 20-67. ALARM_DAYS_REG Register.....                 | 3627 |
| 20-68. ALARM_MONTHS_REG Register .....              | 3628 |
| 20-69. ALARM_YEARS_REG Register.....                | 3629 |
| 20-70. RTC_CTRL_REG Register.....                   | 3630 |
| 20-71. RTC_STATUS_REG Register .....                | 3632 |
| 20-72. RTC_INTERRUPTS_REG Register .....            | 3633 |
| 20-73. RTC_COMP_LSB_REG Register .....              | 3634 |
| 20-74. RTC_COMP_MSB_REG Register .....              | 3635 |
| 20-75. RTC_OSC_REG Register .....                   | 3636 |
| 20-76. RTC_SCRATCH0_REG Register .....              | 3637 |
| 20-77. RTC_SCRATCH1_REG Register .....              | 3638 |
| 20-78. RTC_SCRATCH2_REG Register .....              | 3639 |
| 20-79. KICK0R Register .....                        | 3640 |
| 20-80. KICK1R Register .....                        | 3641 |
| 20-81. RTC_REVISION Register .....                  | 3642 |
| 20-82. RTC_SYSCONFIG Register .....                 | 3643 |
| 20-83. RTC_IRQWAKEEN Register.....                  | 3644 |
| 20-84. ALARM2_SECONDS_REG Register .....            | 3645 |
| 20-85. ALARM2_MINUTES_REG Register .....            | 3646 |

|   |      |
|---|------|
| 20-86. ALARM2_HOURS_REG Register.....                             | 3647 |
| 20-87. ALARM2_DAYS_REG Register .....                             | 3648 |
| 20-88. ALARM2_MONTHS_REG Register.....                            | 3649 |
| 20-89. ALARM2_YEARS_REG Register .....                            | 3650 |
| 20-90. RTC_PMIC Register .....                                    | 3651 |
| 20-91. RTC_DEBOUNCE Register.....                                 | 3652 |
| 20-92. 32-Bit Watchdog Timer Functional Block Diagram.....        | 3655 |
| 20-93. Watchdog Timers General Functional View .....              | 3656 |
| 20-94. WDT_WIDR Register .....                                    | 3664 |
| 20-95. WDT_WDSC Register .....                                    | 3665 |
| 20-96. WDT_WDST Register.....                                     | 3666 |
| 20-97. WDT_WISR Register .....                                    | 3667 |
| 20-98. WDT_WIER Register .....                                    | 3668 |
| 20-99. WDT_WCLR Register.....                                     | 3669 |
| 20-100. WDT_WCRR Register.....                                    | 3670 |
| 20-101. WDT_WLDR Register .....                                   | 3671 |
| 20-102. WDT_WTGR Register.....                                    | 3672 |
| 20-103. WDT_WWPS Register .....                                   | 3673 |
| 20-104. WDT_WDLY Register .....                                   | 3674 |
| 20-105. WDT_WSPR Register .....                                   | 3675 |
| 20-106. WDT_WIRQSTATRAW Register.....                             | 3676 |
| 20-107. WDT_WIRQSTAT Register .....                               | 3677 |
| 20-108. WDT_WIRQENSET Register.....                               | 3678 |
| 20-109. WDT_WIRQENCLR Register.....                               | 3679 |
| 21-1. I2C0 Integration and Bus Application .....                  | 3682 |
| 21-2. I2C(1–2) Integration and Bus Application .....              | 3682 |
| 21-3. I2C Functional Block Diagram .....                          | 3684 |
| 21-4. Multiple I2C Modules Connected.....                         | 3685 |
| 21-5. Bit Transfer on the I2C Bus .....                           | 3686 |
| 21-6. Start and Stop Condition Events .....                       | 3687 |
| 21-7. I2C Data Transfer .....                                     | 3687 |
| 21-8. I2C Data Transfer Formats.....                              | 3688 |
| 21-9. Arbitration Procedure Between Two Master Transmitters ..... | 3689 |
| 21-10. Synchronization of Two I2C Clock Generators .....          | 3690 |
| 21-11. Receive FIFO Interrupt Request Generation .....            | 3692 |
| 21-12. Transmit FIFO Interrupt Request Generation.....            | 3692 |
| 21-13. Receive FIFO DMA Request Generation .....                  | 3693 |
| 21-14. Transmit FIFO DMA Request Generation (High Threshold)..... | 3694 |
| 21-15. Transmit FIFO DMA Request Generation (Low Threshold) ..... | 3694 |
| 21-16. I2C_REVNB_LO Register .....                                | 3699 |
| 21-17. I2C_REVNB_HI Register .....                                | 3700 |
| 21-18. I2C_SYSC Register .....                                    | 3701 |
| 21-19. I2C_IRQSTATUS_RAW Register .....                           | 3703 |
| 21-20. I2C_IRQSTATUS Register .....                               | 3709 |
| 21-21. I2C_IRQENABLE_SET Register .....                           | 3711 |
| 21-22. I2C_IRQENABLE_CLR Register .....                           | 3713 |
| 21-23. I2C_WE Register .....                                      | 3715 |
| 21-24. I2C_DMARXENABLE_SET Register .....                         | 3718 |
| 21-25. I2C_DMATXENABLE_SET Register.....                          | 3719 |

|  |      |
|--|------|
| 21-26. I2C_DMARXENABLE_CLR Register .....                            | 3720 |
| 21-27. I2C_DMATXENABLE_CLR Register .....                            | 3721 |
| 21-28. I2C_DMARXWAKE_EN Register .....                               | 3722 |
| 21-29. I2C_DMATXWAKE_EN Register .....                               | 3724 |
| 21-30. I2C_SYSS Register .....                                       | 3726 |
| 21-31. I2C_BUFS Register .....                                       | 3727 |
| 21-32. I2C_CNT Register .....  | 3729 |
| 21-33. I2C_DATA Register .....                                       | 3730 |
| 21-34. I2C_CON Register .....  | 3731 |
| 21-35. I2C_OA Register .....   | 3734 |
| 21-36. I2C_SA Register .....   | 3735 |
| 21-37. I2C_PSC Register .....  | 3736 |
| 21-38. I2C_SCLL Register .....                                       | 3737 |
| 21-39. I2C_SCLH Register .....                                       | 3738 |
| 21-40. I2C_SYSTEST Register .....                                    | 3739 |
| 21-41. I2C_BUFSSTAT Register .....                                   | 3743 |
| 21-42. I2C_OA1 Register .....  | 3744 |
| 21-43. I2C_OA2 Register .....  | 3745 |
| 21-44. I2C_OA3 Register .....  | 3746 |
| 21-45. I2C_ACTOA Register .....                                      | 3747 |
| 21-46. I2C_SBLOCK Register .....                                     | 3748 |
| 22-1. McASP0–1 Integration .....                                     | 3753 |
| 22-2. McASP Block Diagram .....                                      | 3756 |
| 22-3. McASP to Parallel 2-Channel DACs .....                         | 3757 |
| 22-4. McASP to 6-Channel DAC and 2-Channel DAC .....                 | 3757 |
| 22-5. McASP to Digital Amplifier .....                               | 3758 |
| 22-6. McASP as Digital Audio Encoder .....                           | 3758 |
| 22-7. McASP as 16 Channel Digital Processor .....                    | 3758 |
| 22-8. TDM Format–6 Channel TDM Example .....                         | 3759 |
| 22-9. TDM Format Bit Delays from Frame Sync .....                    | 3760 |
| 22-10. Inter-Integrated Sound (I2S) Format .....                     | 3760 |
| 22-11. Biphase-Mark Code (BMC) .....                                 | 3761 |
| 22-12. S/PDIF Subframe Format .....                                  | 3762 |
| 22-13. S/PDIF Frame Format .....                                     | 3763 |
| 22-14. Definition of Bit, Word, and Slot .....                       | 3764 |
| 22-15. Bit Order and Word Alignment Within a Slot Examples .....     | 3764 |
| 22-16. Definition of Frame and Frame Sync Width .....                | 3765 |
| 22-17. Transmit Clock Generator Block Diagram .....                  | 3766 |
| 22-18. Receive Clock Generator Block Diagram .....                   | 3767 |
| 22-19. Frame Sync Generator Block Diagram .....                      | 3768 |
| 22-20. Burst Frame Sync Mode .....                                   | 3770 |
| 22-21. Transmit DMA Event (AXEVT) Generation in TDM Time Slots ..... | 3772 |
| 22-22. Individual Serializer and Connections Within McASP .....      | 3777 |
| 22-23. Receive Format Unit .....                                     | 3778 |
| 22-24. Transmit Format Unit .....                                    | 3778 |
| 22-25. McASP I/O Pin Control Block Diagram .....                     | 3780 |
| 22-26. Processor Service Time Upon Transmit DMA Event (AXEVT) .....  | 3782 |
| 22-27. Processor Service Time Upon Receive DMA Event (AREVT) .....   | 3783 |
| 22-28. McASP Audio FIFO (AFIFO) Block Diagram .....                  | 3785 |

|   |      |
|---|------|
| 22-29. Data Flow Through Transmit Format Unit, Illustrated .....        | 3788 |
| 22-30. Data Flow Through Receive Format Unit, Illustrated .....         | 3790 |
| 22-31. Transmit Clock Failure Detection Circuit Block Diagram.....      | 3794 |
| 22-32. Receive Clock Failure Detection Circuit Block Diagram.....       | 3796 |
| 22-33. Serializers in Loopback Mode .....                               | 3797 |
| 22-34. Interrupt Multiplexing .....                                     | 3803 |
| 22-35. Audio Mute (AMUTE) Block Diagram.....                            | 3804 |
| 22-36. DMA Events in an Audio Example—Two Events (Scenario 1) .....     | 3806 |
| 22-37. DMA Events in an Audio Example—Four Events (Scenario 2).....     | 3806 |
| 22-38. DMA Events in an Audio Example .....                             | 3807 |
| 22-39. Revision Identification Register (REV).....                      | 3810 |
| 22-40. Power Idle SYSCONFIG Register (PWRIDLESYSCONFIG).....            | 3811 |
| 22-41. Pin Function Register (PFUNC) .....                              | 3812 |
| 22-42. Pin Direction Register (PDIR).....                               | 3814 |
| 22-43. Pin Data Output Register (PDOUT).....                            | 3816 |
| 22-44. Pin Data Input Register (PDIN).....                              | 3818 |
| 22-45. Pin Data Set Register (PDSET) .....                              | 3820 |
| 22-46. Pin Data Clear Register (PDCLR).....                             | 3822 |
| 22-47. Global Control Register (GBLCTL).....                            | 3824 |
| 22-48. Audio Mute Control Register (AMUTE) .....                        | 3826 |
| 22-49. Digital Loopback Control Register (DLBCTL).....                  | 3828 |
| 22-50. Digital Mode Control Register (DITCTL) .....                     | 3829 |
| 22-51. Receiver Global Control Register (RGBLCTL) .....                 | 3830 |
| 22-52. Receive Format Unit Bit Mask Register (RMASK) .....              | 3831 |
| 22-53. Receive Bit Stream Format Register (RFMT) .....                  | 3832 |
| 22-54. Receive Frame Sync Control Register (AFSRCTL).....               | 3834 |
| 22-55. Receive Clock Control Register (ACLKRCTL).....                   | 3835 |
| 22-56. Receive High-Frequency Clock Control Register (AHCLKRCTL).....   | 3836 |
| 22-57. Receive TDM Time Slot Register (RTDM) .....                      | 3837 |
| 22-58. Receiver Interrupt Control Register (RINTCTL) .....              | 3838 |
| 22-59. Receiver Status Register (RSTAT).....                            | 3839 |
| 22-60. Current Receive TDM Time Slot Registers (RSLOT).....             | 3840 |
| 22-61. Receive Clock Check Control Register (RCLKCHK) .....             | 3841 |
| 22-62. Receiver DMA Event Control Register (REVTCTL).....               | 3842 |
| 22-63. Transmitter Global Control Register (XGLCTL) .....               | 3843 |
| 22-64. Transmit Format Unit Bit Mask Register (XMASK) .....             | 3844 |
| 22-65. Transmit Bit Stream Format Register (XFMT) .....                 | 3845 |
| 22-66. Transmit Frame Sync Control Register (AFSXCTL) .....             | 3847 |
| 22-67. Transmit Clock Control Register (ACLKXCTL) .....                 | 3848 |
| 22-68. Transmit High-Frequency Clock Control Register (AHCLKXCTL) ..... | 3849 |
| 22-69. Transmit TDM Time Slot Register (XTDM) .....                     | 3850 |
| 22-70. Transmitter Interrupt Control Register (XINTCTL) .....           | 3851 |
| 22-71. Transmitter Status Register (XSTAT).....                         | 3852 |
| 22-72. Current Transmit TDM Time Slot Register (XSLOT) .....            | 3853 |
| 22-73. Transmit Clock Check Control Register (XCLKCHK).....             | 3854 |
| 22-74. Transmitter DMA Event Control Register (XEVCTL).....             | 3855 |
| 22-75. Serializer Control Registers (SRCTL $n$ ) .....                  | 3856 |
| 22-76. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5).....       | 3857 |
| 22-77. DIT Right Channel Status Registers (DITCSR0-DITCSR5).....        | 3857 |

|   |      |
|---|------|
| 22-78. DIT Left Channel User Data Registers (DITUUDRA0-DITUUDRA5).....  | 3857 |
| 22-79. DIT Right Channel User Data Registers (DITUUDRB0-DITUUDRB5)..... | 3857 |
| 22-80. Transmit Buffer Registers (XBUFn).....                           | 3858 |
| 22-81. Receive Buffer Registers (RBUFn).....                            | 3858 |
| 22-82. Write FIFO Control Register (WFIFOCTL) .....                     | 3859 |
| 22-83. Write FIFO Status Register (WFIFOSTS) .....                      | 3860 |
| 22-84. Read FIFO Control Register (RFIFOCTL) .....                      | 3861 |
| 22-85. Read FIFO Status Register (RFIFOSTS) .....                       | 3862 |
| 23-1. DCAN Integration.....   | 3865 |
| 23-2. DCAN Block Diagram .....  | 3867 |
| 23-3. CAN Module General Initialization Flow.....                       | 3869 |
| 23-4. CAN Bit-Timing Configuration .....                                | 3870 |
| 23-5. CAN Core in Silent Mode.....                                      | 3872 |
| 23-6. CAN Core in Loopback Mode.....                                    | 3873 |
| 23-7. CAN Core in External Loopback Mode .....                          | 3874 |
| 23-8. CAN Core in Loop Back Combined With Silent Mode.....              | 3875 |
| 23-9. CAN Interrupt Topology 1 .....                                    | 3877 |
| 23-10. CAN Interrupt Topology 2 .....                                   | 3877 |
| 23-11. Local Power-Down Mode Flow Diagram .....                         | 3879 |
| 23-12. CPU Handling of a FIFO Buffer (Interrupt Driven) .....           | 3888 |
| 23-13. Bit Timing .....   | 3889 |
| 23-14. The Propagation Time Segment.....                                | 3890 |
| 23-15. Synchronization on Late and Early Edges .....                    | 3892 |
| 23-16. Filtering of Short Dominant Spikes .....                         | 3893 |
| 23-17. Structure of the CAN Core's CAN Protocol Controller.....         | 3894 |
| 23-18. Data Transfer Between IF1/IF2 Registers and Message RAM.....     | 3898 |
| 23-19. CAN Control Register (DCAN CTL).....                             | 3906 |
| 23-20. Error and Status Register (DCAN ES).....                         | 3908 |
| 23-21. Error Counter Register (DCAN ERRC) .....                         | 3910 |
| 23-22. Bit Timing Register (DCAN BTR).....                              | 3911 |
| 23-23. Interrupt Register (DCAN INT).....                               | 3912 |
| 23-24. Test Register (DCAN TEST) .....                                  | 3913 |
| 23-25. Parity Error Code Register (DCAN PERR).....                      | 3914 |
| 23-26. Auto-Bus-On Time Register (DCAN ABOTR) .....                     | 3915 |
| 23-27. Transmission Request X Register (DCAN TXRQ X) .....              | 3916 |
| 23-28. Transmission Request X Register (DCAN TXRQ X) .....              | 3918 |
| 23-29. Interrupt Pending X Register (DCAN INTPND X) .....               | 3920 |
| 23-30. Message Valid X Register (DCAN MSGVAL X) .....                   | 3922 |
| 23-31. IF1 Command Registers (DCAN IF1CMD).....                         | 3925 |
| 23-32. IF2 Command Registers (DCAN IF2CMD).....                         | 3925 |
| 23-33. IF1 Mask Register (DCAN IF1MSK) .....                            | 3928 |
| 23-34. IF2 Mask Register (DCAN IF2MSK) .....                            | 3928 |
| 23-35. IF1 Arbitration Register (DCAN IF1ARB).....                      | 3929 |
| 23-36. IF2 Arbitration Register (DCAN IF2ARB).....                      | 3929 |
| 23-37. IF1 Message Control Register (DCAN IF1MCTL).....                 | 3931 |
| 23-38. IF2 Message Control Register (DCAN IF2MCTL).....                 | 3931 |
| 23-39. IF1 Data A Register (DCAN IF1DATA) .....                         | 3933 |
| 23-40. IF1 Data B Register (DCAN IF1DATA) .....                         | 3933 |
| 23-41. IF2 Data A Register (DCAN IF2DATA) .....                         | 3933 |

|  |      |
|--|------|
| 23-42. IF2 Data B Register (DCAN IF2DATA) .....  | 3933 |
| 23-43. IF3 Observation Register (DCAN IF3OBS) .....  | 3934 |
| 23-44. IF3 Mask Register (DCAN IF3MSK) .....   | 3936 |
| 23-45. IF3 Arbitration Register (DCAN IF3ARB).....   | 3937 |
| 23-46. IF3 Message Control Register (DCAN IF3MCTL).....  | 3938 |
| 23-47. IF3 Data A Register (DCAN IF3DATA) .....  | 3940 |
| 23-48. IF3 Data A Register (DCAN IF3DATB) .....  | 3940 |
| 23-49. CAN TX I/O Control Register (DCAN TIOC).....  | 3942 |
| 23-50. CAN RX IO control register (DCAN RIOC).....   | 3944 |
| 24-1. SPI Master Application.....  | 3948 |
| 24-2. SPI Slave Application .....  | 3948 |
| 24-3. SPI Full-Duplex Transmission .....   | 3951 |
| 24-4. SPI Half-Duplex Transmission (Receive-only Slave) .....                                  | 3952 |
| 24-5. SPI Half-Duplex Transmission (Transmit-Only Slave).....                                  | 3952 |
| 24-6. Phase and Polarity Combinations .....  | 3954 |
| 24-7. Full Duplex Single Transfer Format with PHA = 0 .....                                    | 3955 |
| 24-8. Full Duplex Single Transfer Format With PHA = 1.....                                     | 3956 |
| 24-9. Continuous Transfers With SPIEN Maintained Active (Single-Data-Pin Interface Mode) ..... | 3961 |
| 24-10. Continuous Transfers With SPIEN Maintained Active (Dual-Data-Pin Interface Mode) .....  | 3961 |
| 24-11. Extended SPI Transfer With Start Bit PHA = 1 .....                                      | 3963 |
| 24-12. Chip-Select SPIEN Timing Controls .....   | 3964 |
| 24-13. Transmit/Receive Mode With No FIFO Used.....  | 3968 |
| 24-14. Transmit/Receive Mode With Only Receive FIFO Enabled .....                              | 3968 |
| 24-15. Transmit/Receive Mode With Only Transmit FIFO Used .....                                | 3969 |
| 24-16. Transmit/Receive Mode With Both FIFO Direction Used .....                               | 3969 |
| 24-17. Transmit-Only Mode With FIFO Used .....   | 3970 |
| 24-18. Receive-Only Mode With FIFO Used .....  | 3970 |
| 24-19. Buffer Almost Full Level (AFL).....   | 3971 |
| 24-20. Buffer Almost Empty Level (AEL) .....   | 3972 |
| 24-21. Master Single Channel Initial Delay.....  | 3973 |
| 24-22. 3-Pin Mode System Overview .....  | 3974 |
| 24-23. Example of SPI Slave with One Master and Multiple Slave Devices on Channel 0.....       | 3976 |
| 24-24. SPI Half-Duplex Transmission (Receive-Only Slave).....                                  | 3978 |
| 24-25. SPI Half-Duplex Transmission (Transmit-Only Slave).....                                 | 3979 |
| 24-26. McSPI Revision Register (MCSPI_REVISION) .....  | 3987 |
| 24-27. McSPI System Configuration Register (MCSPI_SYSCONFIG) .....                             | 3988 |
| 24-28. McSPI System Status Register (MCSPI_SYSSTATUS) .....                                    | 3989 |
| 24-29. McSPI Interrupt Status Register (MCSPI_IRQSTATUS).....                                  | 3990 |
| 24-30. McSPI Interrupt Enable Register (MCSPI_IRQENABLE) .....                                 | 3993 |
| 24-31. McSPI System Register (MCSPI_SYST).....   | 3995 |
| 24-32. McSPI Module Control Register (MCSPI_MODULCTRL).....                                    | 3997 |
| 24-33. McSPI Channel ( <i>i</i> ) Configuration Register (MCSPI_CH( <i>i</i> )CONF) .....      | 3999 |
| 24-34. McSPI Channel ( <i>i</i> ) Status Register (MCSPI_CH( <i>i</i> )STAT).....              | 4003 |
| 24-35. McSPI Channel ( <i>i</i> ) Control Register (MCSPI_CH( <i>i</i> )CTRL) .....            | 4004 |
| 24-36. McSPI Channel ( <i>i</i> ) Transmit Register (MCSPI_TX( <i>i</i> ) ) .....              | 4005 |
| 24-37. McSPI Channel ( <i>i</i> ) Receive Register (MCSPI_RX( <i>i</i> ) ) .....               | 4005 |
| 24-38. McSPI Transfer Levels Register (MCSPI_XFERLEVEL) .....                                  | 4006 |
| 24-39. McSPI DMA Address Aligned FIFO Transmitter Register (MCSPI_DAFTX) .....                 | 4007 |
| 24-40. McSPI DMA Address Aligned FIFO Receiver Register (MCSPI_DAFRX).....                     | 4008 |

|        |  |      |
|--------|--|------|
| 25-1.  | GPIO0 Module Integration .....                               | 4011 |
| 25-2.  | GPIO[1–3] Module Integration .....                           | 4011 |
| 25-3.  | Interrupt Request Generation .....                           | 4016 |
| 25-4.  | Write @ GPIO_CLEARDATAOUT Register Example .....             | 4018 |
| 25-5.  | Write @ GPIO_SETIRQENABLEx Register Example .....            | 4019 |
| 25-6.  | General-Purpose Interface Used as a Keyboard Interface ..... | 4020 |
| 25-7.  | GPIO_REVISION Register .....                                 | 4022 |
| 25-8.  | GPIO_SYSCONFIG Register .....                                | 4023 |
| 25-9.  | GPIO_IRQSTATUS_RAW_0 Register.....                           | 4024 |
| 25-10. | GPIO_IRQSTATUS_RAW_1 Register.....                           | 4025 |
| 25-11. | GPIO_IRQSTATUS_0 Register .....                              | 4026 |
| 25-12. | GPIO_IRQSTATUS_1 Register .....                              | 4027 |
| 25-13. | GPIO_IRQSTATUS_SET_0 Register.....                           | 4028 |
| 25-14. | GPIO_IRQSTATUS_SET_1 Register.....                           | 4029 |
| 25-15. | GPIO_IRQSTATUS_CLR_0 Register.....                           | 4030 |
| 25-16. | GPIO_IRQSTATUS_CLR_1 Register.....                           | 4031 |
| 25-17. | GPIO_IRQWAKEN_0 Register .....                               | 4032 |
| 25-18. | GPIO_IRQWAKEN_1 Register .....                               | 4033 |
| 25-19. | GPIO_SYSSTATUS Register.....                                 | 4034 |
| 25-20. | GPIO_CTRL Register .....                                     | 4035 |
| 25-21. | GPIO_OE Register .....                                       | 4036 |
| 25-22. | GPIO_DATAIN Register .....                                   | 4037 |
| 25-23. | GPIO_DATAOUT Register .....                                  | 4038 |
| 25-24. | GPIO_LEVELDETECT0 Register .....                             | 4039 |
| 25-25. | GPIO_LEVELDETECT1 Register .....                             | 4040 |
| 25-26. | GPIO_RISINGDETECT Register .....                             | 4041 |
| 25-27. | GPIO_FALLINGDETECT Register .....                            | 4042 |
| 25-28. | GPIO_DEBOUNCENABLE Register.....                             | 4043 |
| 25-29. | GPIO_DEBOUNCINGTIME Register .....                           | 4044 |
| 25-30. | GPIO_CLEARDATAOUT Register .....                             | 4045 |
| 25-31. | GPIO_SETDATAOUT Register.....                                | 4046 |
| 26-1.  | Public ROM Code Architecture .....                           | 4048 |
| 26-2.  | Public ROM Code Boot Procedure .....                         | 4049 |
| 26-3.  | ROM Memory Map .....   | 4050 |
| 26-4.  | Public RAM Memory Map .....                                  | 4052 |
| 26-5.  | ROM Code Startup Sequence .....                              | 4054 |
| 26-6.  | ROM Code Booting Procedure .....                             | 4055 |
| 26-7.  | Fast External Boot.....                                      | 4065 |
| 26-8.  | Memory Booting .....   | 4066 |
| 26-9.  | GPMC XIP Timings .....                                       | 4068 |
| 26-10. | Image Shadowing on GP Device .....                           | 4070 |
| 26-11. | GPMC NAND Timings .....                                      | 4071 |
| 26-12. | NAND Device Detection .....                                  | 4075 |
| 26-13. | NAND Invalid Blocks Detection.....                           | 4076 |
| 26-14. | NAND Read Sector Procedure .....                             | 4077 |
| 26-15. | ECC Data Mapping for 2 KB Page and 8b BCH Encoding .....     | 4078 |
| 26-16. | ECC Data Mapping for 4 KB Page and 16b BCH Encoding.....     | 4079 |
| 26-17. | MMC/SD Booting .....   | 4081 |
| 26-18. | MMC/SD Detection Procedure.....                              | 4082 |

---

|   |      |
|---|------|
| 26-19. MMC/SD Booting, Get Booting File ..... | 4085 |
| 26-20. MBR Detection Procedure.....           | 4086 |
| 26-21. MBR, Get Partition .....               | 4087 |
| 26-22. FAT Detection Procedure .....          | 4090 |
| 26-23. Peripheral Booting Procedure .....     | 4094 |
| 26-24. USB Initialization Procedure .....     | 4098 |
| 26-25. Image Transfer for USB Boot.....       | 4099 |
| 26-26. Image Formats on GP Devices .....      | 4099 |
| 26-27. Wakeup Booting by ROM .....            | 4102 |

## List of Tables

|       |   |     |
|-------|---|-----|
| 1-1.  | Device Features .....                                       | 147 |
| 1-2.  | Device_ID (Address 0x44E10600) Bit Field Descriptions ..... | 148 |
| 1-3.  | DEV_FEATURE (Address 0x44E10604) Register Values .....      | 149 |
| 2-1.  | L3 Memory Map .....   | 150 |
| 2-2.  | L4_WKUP Peripheral Memory Map .....                         | 152 |
| 2-3.  | L4_PER Peripheral Memory Map .....                          | 153 |
| 2-4.  | L4 Fast Peripheral Memory Map .....                         | 157 |
| 2-5.  | M3 Processor Memory Map .....                               | 159 |
| 3-1.  | MPU Subsystem Clock Frequencies .....                       | 165 |
| 3-2.  | Reset Scheme of the MPU Subsystem .....                     | 166 |
| 3-3.  | ARM Core Supported Features .....                           | 168 |
| 3-4.  | Overview of the MPU Subsystem Power Domain .....            | 169 |
| 3-5.  | MPU Power States.....                                       | 170 |
| 3-6.  | MPU Subsystem Operation Power Modes .....                   | 171 |
| 5-1.  | SGX530 Connectivity Attributes.....                         | 179 |
| 5-2.  | SGX530 Clock Signals .....                                  | 179 |
| 6-1.  | ARM Cortex-A8 Interrupts .....                              | 196 |
| 6-2.  | ARM Cortex-M3 Wakeup Processor Interrupts .....             | 200 |
| 6-3.  | Timer and eCAP Event Capture .....                          | 202 |
| 6-4.  | INTC REGISTERS.....   | 203 |
| 6-5.  | INTC_REVISION Register Field Descriptions.....              | 205 |
| 6-6.  | INTC_SYSCONFIG Register Field Descriptions.....             | 206 |
| 6-7.  | INTC_SYSSTATUS Register Field Descriptions .....            | 207 |
| 6-8.  | INTC_SIR_IRQ Register Field Descriptions.....               | 208 |
| 6-9.  | INTC_SIR_FIQ Register Field Descriptions .....              | 209 |
| 6-10. | INTC_CONTROL Register Field Descriptions .....              | 210 |
| 6-11. | INTC_PROTECTION Register Field Descriptions .....           | 211 |
| 6-12. | INTC_IDLE Register Field Descriptions.....                  | 212 |
| 6-13. | INTC_IRQ_PRIORITY Register Field Descriptions .....         | 213 |
| 6-14. | INTC_FIQ_PRIORITY Register Field Descriptions.....          | 214 |
| 6-15. | INTC_THRESHOLD Register Field Descriptions .....            | 215 |
| 6-16. | INTC_ITR0 Register Field Descriptions.....                  | 216 |
| 6-17. | INTC_MIR0 Register Field Descriptions .....                 | 217 |
| 6-18. | INTC_MIR_CLEAR0 Register Field Descriptions.....            | 218 |
| 6-19. | INTC_MIR_SET0 Register Field Descriptions .....             | 219 |
| 6-20. | INTC_ISR_SET0 Register Field Descriptions.....              | 220 |
| 6-21. | INTC_ISR_CLEAR0 Register Field Descriptions .....           | 221 |
| 6-22. | INTC_PENDING_IRQ0 Register Field Descriptions .....         | 222 |
| 6-23. | INTC_PENDING_FIQ0 Register Field Descriptions.....          | 223 |
| 6-24. | INTC_ITR1 Register Field Descriptions.....                  | 224 |
| 6-25. | INTC_MIR1 Register Field Descriptions .....                 | 225 |
| 6-26. | INTC_MIR_CLEAR1 Register Field Descriptions.....            | 226 |
| 6-27. | INTC_MIR_SET1 Register Field Descriptions .....             | 227 |
| 6-28. | INTC_ISR_SET1 Register Field Descriptions.....              | 228 |
| 6-29. | INTC_ISR_CLEAR1 Register Field Descriptions .....           | 229 |
| 6-30. | INTC_PENDING_IRQ1 Register Field Descriptions .....         | 230 |
| 6-31. | INTC_PENDING_FIQ1 Register Field Descriptions .....         | 231 |

|       |  |     |
|-------|--|-----|
| 6-32. | INTC_ITR2 Register Field Descriptions.....                                 | 232 |
| 6-33. | INTC_MIR2 Register Field Descriptions .....                                | 233 |
| 6-34. | INTC_MIR_CLEAR2 Register Field Descriptions.....                           | 234 |
| 6-35. | INTC_MIR_SET2 Register Field Descriptions .....                            | 235 |
| 6-36. | INTC_ISR_SET2 Register Field Descriptions.....                             | 236 |
| 6-37. | INTC_ISR_CLEAR2 Register Field Descriptions .....                          | 237 |
| 6-38. | INTC_PENDING_IRQ2 Register Field Descriptions .....                        | 238 |
| 6-39. | INTC_PENDING_FIQ2 Register Field Descriptions .....                        | 239 |
| 6-40. | INTC_ITR3 Register Field Descriptions.....                                 | 240 |
| 6-41. | INTC_MIR3 Register Field Descriptions .....                                | 241 |
| 6-42. | INTC_MIR_CLEAR3 Register Field Descriptions.....                           | 242 |
| 6-43. | INTC_MIR_SET3 Register Field Descriptions .....                            | 243 |
| 6-44. | INTC_ISR_SET3 Register Field Descriptions .....                            | 244 |
| 6-45. | INTC_ISR_CLEAR3 Register Field Descriptions .....                          | 245 |
| 6-46. | INTC_PENDING_IRQ3 Register Field Descriptions .....                        | 246 |
| 6-47. | INTC_PENDING_FIQ3 Register Field Descriptions .....                        | 247 |
| 6-48. | INTC_ILR0 to INTC_ILR127 Register Field Descriptions .....                 | 248 |
| 7-1.  | Unsupported GPMC Features.....   | 252 |
| 7-2.  | GPMC Connectivity Attributes.....  | 253 |
| 7-3.  | GPMC Clock Signals .....   | 253 |
| 7-4.  | GPMC Signal List .....   | 254 |
| 7-5.  | GPMC Pin Multiplexing Options .....  | 255 |
| 7-6.  | GPMC Clocks .....  | 260 |
| 7-7.  | GPMC_CONFIG1_i Configuration .....   | 260 |
| 7-8.  | GPMC Local Power Management Features.....                                  | 260 |
| 7-9.  | GPMC Interrupt Events .....  | 261 |
| 7-10. | Idle Cycle Insertion Configuration.....                                    | 272 |
| 7-11. | Chip-Select Configuration for NAND Interfacing.....                        | 301 |
| 7-12. | ECC Enable Settings .....  | 310 |
| 7-13. | Flattened BCH Codeword Mapping (512 Bytes + 104 Bits) .....                | 315 |
| 7-14. | Aligned Message Byte Mapping in 8-bit NAND .....                           | 315 |
| 7-15. | Aligned Message Byte Mapping in 16-bit NAND .....                          | 316 |
| 7-16. | Aligned Nibble Mapping of Message in 8-bit NAND.....                       | 316 |
| 7-17. | Misaligned Nibble Mapping of Message in 8-bit NAND.....                    | 316 |
| 7-18. | Aligned Nibble Mapping of Message in 16-bit NAND .....                     | 316 |
| 7-19. | Misaligned Nibble Mapping of Message in 16-bit NAND (1 Unused Nibble)..... | 317 |
| 7-20. | Misaligned Nibble Mapping of Message in 16-bit NAND (2 Unused Nibble)..... | 317 |
| 7-21. | Misaligned Nibble Mapping of Message in 16-bit NAND (3 Unused Nibble)..... | 317 |
| 7-22. | Prefetch Mode Configuration .....  | 328 |
| 7-23. | Write-Posting Mode Configuration .....                                     | 330 |
| 7-24. | GPMC Configuration in NOR Mode .....                                       | 336 |
| 7-25. | GPMC Configuration in NAND Mode.....                                       | 336 |
| 7-26. | Reset GPMC.....  | 336 |
| 7-27. | NOR Memory Type .....  | 337 |
| 7-28. | NOR Chip-Select Configuration .....  | 337 |
| 7-29. | NOR Timings Configuration .....  | 337 |
| 7-30. | WAIT Pin Configuration .....   | 337 |
| 7-31. | Enable Chip-Select .....   | 338 |
| 7-32. | NAND Memory Type .....   | 338 |

|       |  |     |
|-------|--|-----|
| 7-33. | NAND Chip-Select Configuration .....                                 | 338 |
| 7-34. | Asynchronous Read and Write Operations.....                          | 338 |
| 7-35. | ECC Engine.....  | 338 |
| 7-36. | Prefetch and Write-Posting Engine .....                              | 340 |
| 7-37. | WAIT Pin Configuration .....   | 340 |
| 7-38. | Enable Chip-Select .....   | 340 |
| 7-39. | Mode Parameters Check List Table .....                               | 341 |
| 7-40. | Access Type Parameters Check List Table .....                        | 341 |
| 7-41. | Timing Parameters.....   | 343 |
| 7-42. | NAND Formulas Description Table .....                                | 345 |
| 7-43. | Synchronous NOR Formulas Description Table .....                     | 346 |
| 7-44. | Asynchronous NOR Formulas Description Table.....                     | 352 |
| 7-45. | GPMC Signals.....  | 354 |
| 7-46. | Useful Timing Parameters on the Memory Side .....                    | 356 |
| 7-47. | Calculating GPMC Timing Parameters.....                              | 357 |
| 7-48. | AC Characteristics for Asynchronous Read Access .....                | 358 |
| 7-49. | GPMC Timing Parameters for Asynchronous Read Access .....            | 359 |
| 7-50. | AC Characteristics for Asynchronous Single Write (Memory Side) ..... | 360 |
| 7-51. | GPMC Timing Parameters for Asynchronous Single Write .....           | 361 |
| 7-52. | NAND Interface Bus Operations Summary.....                           | 362 |
| 7-53. | NOR Interface Bus Operations Summary .....                           | 362 |
| 7-54. | GPMC Registers .....   | 365 |
| 7-55. | GPMC_REVISION Field Descriptions.....                                | 366 |
| 7-56. | GPMC_SYSCONFIG Field Descriptions.....                               | 366 |
| 7-57. | GPMC_SYSSTATUS Field Descriptions.....                               | 367 |
| 7-58. | GPMC_IRQSTATUS Field Descriptions .....                              | 368 |
| 7-59. | GPMC_IRQENABLE Field Descriptions .....                              | 369 |
| 7-60. | GPMC_TIMEOUT_CONTROL Field Descriptions .....                        | 370 |
| 7-61. | GPMC_ERR_ADDRESS Field Descriptions .....                            | 370 |
| 7-62. | GPMC_ERR_TYPE Field Descriptions .....                               | 371 |
| 7-63. | GPMC_CONFIG Field Descriptions .....                                 | 372 |
| 7-64. | GPMC_STATUS Field Descriptions .....                                 | 373 |
| 7-65. | GPMC_CONFIG1_i Field Descriptions .....                              | 374 |
| 7-66. | GPMC_CONFIG2_i Field Descriptions .....                              | 376 |
| 7-67. | GPMC_CONFIG3_i Field Descriptions .....                              | 377 |
| 7-68. | GPMC_CONFIG4_i Field Descriptions .....                              | 379 |
| 7-69. | GPMC_CONFIG5_i Field Descriptions .....                              | 381 |
| 7-70. | GPMC_CONFIG6_i Field Descriptions .....                              | 382 |
| 7-71. | GPMC_CONFIG7_i Field Descriptions .....                              | 383 |
| 7-72. | GPMC_NAND_COMMAND_i Field Descriptions .....                         | 384 |
| 7-73. | GPMC_NAND_ADDRESS_i Field Descriptions .....                         | 384 |
| 7-74. | GPMC_NAND_DATA_i Field Descriptions .....                            | 384 |
| 7-75. | GPMC_PREFETCH_CONFIG1 Field Descriptions.....                        | 385 |
| 7-76. | GPMC_PREFETCH_CONFIG2 Field Descriptions.....                        | 387 |
| 7-77. | GPMC_PREFETCH_CONTROL Field Descriptions .....                       | 387 |
| 7-78. | GPMC_PREFETCH_STATUS Field Descriptions .....                        | 388 |
| 7-79. | GPMC_ECC_CONFIG Field Descriptions .....                             | 389 |
| 7-80. | GPMC_ECC_CONTROL Field Descriptions .....                            | 390 |
| 7-81. | GPMC_ECC_SIZE_CONFIG Field Descriptions .....                        | 391 |

|        |  |     |
|--------|--|-----|
| 7-82.  | GPMC_ECCj_RESULT Field Descriptions .....  | 393 |
| 7-83.  | GPMC_BCH_RESULT0_i Field Descriptions .....  | 394 |
| 7-84.  | GPMC_BCH_RESULT1_i Field Descriptions .....  | 394 |
| 7-85.  | GPMC_BCH_RESULT2_i Field Descriptions .....  | 394 |
| 7-86.  | GPMC_BCH_RESULT3_i Field Descriptions .....  | 395 |
| 7-87.  | GPMC_BCH_SWDATA Field Descriptions .....   | 395 |
| 7-88.  | GPMC_BCH_RESULT4_i Field Descriptions .....  | 395 |
| 7-89.  | GPMC_BCH_RESULT5_i Field Descriptions .....  | 396 |
| 7-90.  | GPMC_BCH_RESULT6_i Field Descriptions .....  | 396 |
| 7-91.  | OCMC RAM Connectivity Attributes.....  | 398 |
| 7-92.  | OCMC RAM Clock Signals .....   | 398 |
| 7-93.  | Unsupported EMIF Features .....  | 400 |
| 7-94.  | EMIF Connectivity Attributes .....   | 401 |
| 7-95.  | EMIF Clock Signals .....   | 401 |
| 7-96.  | EMIF Pin List .....  | 401 |
| 7-97.  | DDR2/3/mDDR Memory Controller Signal Descriptions .....                                    | 403 |
| 7-98.  | Digital Filter Configuration .....   | 407 |
| 7-99.  | IBANK, RSIZE and PAGESIZE Fields Information .....   | 408 |
| 7-100. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=0 and REG_EBANK_POS=0.....    | 409 |
| 7-101. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=1 and REG_EBANK_POS=0.....    | 410 |
| 7-102. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=2 and REG_EBANK_POS=0.....    | 410 |
| 7-103. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=3 and REG_EBANK_POS=0.....    | 411 |
| 7-104. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=0 and REG_EBANK_POS=1.....    | 411 |
| 7-105. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=1 and REG_EBANK_POS = 1 ..... | 411 |
| 7-106. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=2 and REG_EBANK_POS = 1 ..... | 412 |
| 7-107. | OCP Address to DDR2/3/mDDR Address Mapping for REG_IBANK_POS=3 and REG_EBANK_POS=1.....    | 412 |
| 7-108. | Refresh Modes .....  | 415 |
| 7-109. | EMIF4D REGISTERS .....   | 421 |
| 7-110. | EMIF_MOD_ID_REV Register Field Descriptions .....  | 423 |
| 7-111. | STATUS Register Field Descriptions.....  | 424 |
| 7-112. | SDRAM_CONFIG Register Field Descriptions.....  | 425 |
| 7-113. | SDRAM_CONFIG_2 Register Field Descriptions.....  | 427 |
| 7-114. | SDRAM_REF_CTRL Register Field Descriptions .....   | 428 |
| 7-115. | SDRAM_REF_CTRL_SHDW Register Field Descriptions .....                                      | 429 |
| 7-116. | SDRAM_TIM_1 Register Field Descriptions .....  | 430 |
| 7-117. | SDRAM_TIM_1_SHDW Register Field Descriptions.....  | 431 |
| 7-118. | SDRAM_TIM_2 Register Field Descriptions .....  | 432 |
| 7-119. | SDRAM_TIM_2_SHDW Register Field Descriptions.....  | 433 |
| 7-120. | SDRAM_TIM_3 Register Field Descriptions .....  | 434 |
| 7-121. | SDRAM_TIM_3_SHDW Register Field Descriptions.....  | 435 |
| 7-122. | PWR_MGMT_CTRL Register Field Descriptions.....   | 436 |
| 7-123. | PWR_MGMT_CTRL_SHDW Register Field Descriptions .....                                       | 438 |
| 7-124. | Interface Configuration Register Field Descriptions .....                                  | 439 |

|  |     |
|--|-----|
| 7-125. Interface Configuration Value 1 Register Field Descriptions .....   | 440 |
| 7-126. Interface Configuration Value 2 Register Field Descriptions .....   | 441 |
| 7-127. PERF_CNT_1 Register Field Descriptions .....  | 442 |
| 7-128. PERF_CNT_2 Register Field Descriptions .....  | 443 |
| 7-129. PERF_CNT_CFG Register Field Descriptions .....  | 444 |
| 7-130. PERF_CNT_SEL Register Field Descriptions .....  | 445 |
| 7-131. PERF_CNT_TIM Register Field Descriptions .....  | 446 |
| 7-132. READ_IDLE_CTRL Register Field Descriptions .....  | 447 |
| 7-133. READ_IDLE_CTRL_SHDW Register Field Descriptions .....   | 448 |
| 7-134. IRQSTATUS_RAW_SYS Register Field Descriptions .....   | 449 |
| 7-135. IRQSTATUS_SYS Register Field Descriptions .....   | 450 |
| 7-136. IRQENABLE_SET_SYS Register Field Descriptions .....   | 451 |
| 7-137. IRQENABLE_CLR_SYS Register Field Descriptions .....   | 452 |
| 7-138. ZQ_CONFIG Register Field Descriptions.....  | 453 |
| 7-139. Read-Write Leveling Ramp Window Register Field Descriptions .....   | 454 |
| 7-140. Read-Write Leveling Ramp Control Register Field Descriptions .....  | 455 |
| 7-141. Read-Write Leveling Control Register Field Descriptions .....   | 456 |
| 7-142. DDR_PHY_CTRL_1 Register Field Descriptions .....  | 457 |
| 7-143. DDR_PHY_CTRL_1_SHDW Register Field Descriptions .....   | 459 |
| 7-144. Priority to Class of Service Mapping Register Field Descriptions .....  | 461 |
| 7-145. Connection ID to Class of Service 1 Mapping Register Field Descriptions .....   | 462 |
| 7-146. Connection ID to Class of Service 2 Mapping Register Field Descriptions .....   | 463 |
| 7-147. Read Write Execution Threshold Register Field Descriptions .....  | 465 |
| 7-148. Memory-Mapped Registers for DDR2/3/mDDR PHY .....   | 466 |
| 7-149. DDR PHY Command 0/1/2 Address/Command Slave Ratio Register<br>(CMD0/1/2_REG_PHY_CTRL_SLAVE_RATIO_0) Field Descriptions .....                  | 468 |
| 7-150. DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register(<br>CMD0/1/2_REG_PHY_DLL_LOCK_DIFF_0) Field Descriptions .....             | 468 |
| 7-151. DDR PHY Command 0/1/2 Invert Clockout Selection Register(<br>CMD0/1/2_REG_PHY_INVERT_CLKOUT_0) Field Descriptions .....                       | 469 |
| 7-152. DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register<br>(DATA0/1_REG_PHY_RD_DQS_SLAVE_RATIO_0) Field Descriptions .....                       | 469 |
| 7-153. DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register<br>(DATA0/1_REG_PHY_WR_DQS_SLAVE_RATIO_0) .....   | 470 |
| 7-154. DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register(<br>DATA0/1_REG_PHY_WR_DQS_SLAVE_RATIO_0) Field Descriptions.....                       | 470 |
| 7-155. DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (<br>DATA0/1_REG_PHY_WRLVL_INIT_RATIO_0) Field Descriptions.....                    | 470 |
| 7-156. DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register<br>(DATA0/1_REG_PHY_WRLVL_INIT_MODE_0) .....                           | 471 |
| 7-157. DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register<br>(DATA0_REG_PHY_GATELVL_INIT_RATIO_0) Field Descriptions .....                   | 471 |
| 7-158. DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register<br>(DATA0/1_REG_PHY_GATELVL_INIT_MODE_0) Field Descriptions ..... | 472 |
| 7-159. DDR PHY Data Macro 0/1 DQS Gate Slave Ratio Register<br>(DATA0/1_REG_PHY_FIFO_WE_SLAVE_RATIO_0) Field Descriptions.....                       | 472 |
| 7-160. DDR PHY Data Macro 0/1 Write Data Slave Ratio Register<br>(DATA0/1_REG_PHY_WR_DATA_SLAVE_RATIO_0) Field Descriptions .....                    | 473 |
| 7-161. DDR PHY Data Macro 0/1 Delay Selection Register (DATA0/1_REG_PHY_USE_RANK0_DELAYS)<br>Field Descriptions .....                                | 474 |
| 7-162. ELM Connectivity Attributes .....   | 476 |
| 7-163. ELM Clock Signals.....  | 476 |

|   |     |
|---|-----|
| 7-164. Local Power Management Features.....   | 477 |
| 7-165. Events .....   | 477 |
| 7-166. ELM_LOCATION_STATUS_i Value Decoding Table .....                                   | 479 |
| 7-167. ELM Processing Initialization .....  | 480 |
| 7-168. ELM Processing Completion for Continuous Mode .....                                | 480 |
| 7-169. ELM Processing Completion for Page Mode.....                                       | 481 |
| 7-170. Use Case: Continuous Mode .....  | 481 |
| 7-171. 16-bit NAND Sector Buffer Address Map .....  | 483 |
| 7-172. Use Case: Page Mode .....  | 483 |
| 7-173. ELM Registers.....   | 486 |
| 7-174. ELM Revision Register (ELM_REVISION) Field Descriptions .....                      | 487 |
| 7-175. ELM System Configuration Register (ELM_SYSCONFIG) Field Descriptions.....          | 487 |
| 7-176. ELM System Status Register (ELM_SYSSTATUS) Field Descriptions .....                | 488 |
| 7-177. ELM Interrupt Status Register (ELM_IRQSTATUS) Field Descriptions .....             | 489 |
| 7-178. ELM Interrupt Enable Register (ELM_IRQENABLE) Field Descriptions.....              | 491 |
| 7-179. ELM Location Configuration Register (ELM_LOCATION_CONFIG) Field Descriptions ..... | 492 |
| 7-180. ELM Page Definition Register (ELM_PAGE_CTRL) Field Descriptions .....              | 493 |
| 7-181. ELM_SYNDROME_FRAGMENT_0_i Register Field Descriptions .....                        | 494 |
| 7-182. ELM_SYNDROME_FRAGMENT_1_i Register Field Descriptions .....                        | 494 |
| 7-183. ELM_SYNDROME_FRAGMENT_2_i Register Field Descriptions .....                        | 494 |
| 7-184. ELM_SYNDROME_FRAGMENT_3_i Register Field Descriptions .....                        | 495 |
| 7-185. ELM_SYNDROME_FRAGMENT_4_i Register Field Descriptions .....                        | 495 |
| 7-186. ELM_SYNDROME_FRAGMENT_5_i Register Field Descriptions .....                        | 495 |
| 7-187. ELM_SYNDROME_FRAGMENT_6_i Register Field Descriptions .....                        | 496 |
| 7-188. ELM_LOCATION_STATUS_i Register Field Descriptions .....                            | 496 |
| 7-189. ELM_ERROR_LOCATION_0-15_i Registers Field Descriptions .....                       | 497 |
| 8-1. Master Module Standby-Mode Settings.....   | 500 |
| 8-2. Master Module Standby Status .....   | 501 |
| 8-3. Module Idle Mode Settings .....  | 501 |
| 8-4. Idle States for a Slave Module .....   | 502 |
| 8-5. Slave Module Mode Settings in PRCM .....   | 502 |
| 8-6. Module Clock Enabling Condition.....   | 503 |
| 8-7. Clock Domain Functional Clock States .....   | 504 |
| 8-8. Clock Domain States .....  | 505 |
| 8-9. Clock Transition Mode Settings .....   | 505 |
| 8-10. States of a Memory Area in a Power Domain.....                                      | 506 |
| 8-11. States of a Logic Area in a Power Domain .....                                      | 506 |
| 8-12. Power Domain Control and Status Registers .....                                     | 506 |
| 8-13. Typical Power Modes.....  | 507 |
| 8-14. M3 Interrupts 1–3 .....   | 511 |
| 8-15. Output Clocks in Locked Condition .....   | 516 |
| 8-16. Output Clocks Before Lock and During Relock Modes .....                             | 517 |
| 8-17. Output Clocks in Locked Condition.....  | 519 |
| 8-18. Output Clocks Before Lock and During Relock Modes .....                             | 519 |
| 8-19. PLL and Clock Frequencies.....  | 522 |
| 8-20. Core PLL Typical Frequencies (MHz) .....  | 522 |
| 8-21. Bus Interface Clocks.....   | 523 |
| 8-22. Per PLL Typical Frequencies (MHz) .....   | 524 |
| 8-23. Reset Sources.....  | 538 |

|   |     |
|---|-----|
| 8-24. Core Logic Voltage and Power Domains .....                | 541 |
| 8-25. Power Domain State Table .....                            | 541 |
| 8-26. Power Domain of Various Modules.....                      | 541 |
| 8-27. CM_PER REGISTERS.....                                     | 544 |
| 8-28. CM_PER_L4LS_CLKSTCTRL Register Field Descriptions .....   | 546 |
| 8-29. CM_PER_L3S_CLKSTCTRL Register Field Descriptions.....     | 548 |
| 8-30. CM_PER_L4FW_CLKSTCTRL Register Field Descriptions .....   | 549 |
| 8-31. CM_PER_L3_CLKSTCTRL Register Field Descriptions .....     | 550 |
| 8-32. CM_PER_CPGMAC0_CLKCTRL Register Field Descriptions .....  | 551 |
| 8-33. CM_PER_LCDC_CLKCTRL Register Field Descriptions .....     | 552 |
| 8-34. CM_PER_USB0_CLKCTRL Register Field Descriptions.....      | 553 |
| 8-35. CM_PER_TPTC0_CLKCTRL Register Field Descriptions .....    | 554 |
| 8-36. CM_PER_EMIF_CLKCTRL Register Field Descriptions .....     | 555 |
| 8-37. CM_PER_OCMCRAM_CLKCTRL Register Field Descriptions.....   | 556 |
| 8-38. CM_PER_GPMC_CLKCTRL Register Field Descriptions.....      | 557 |
| 8-39. CM_PER_MCASP0_CLKCTRL Register Field Descriptions.....    | 558 |
| 8-40. CM_PER_UART5_CLKCTRL Register Field Descriptions .....    | 559 |
| 8-41. CM_PER_MMC0_CLKCTRL Register Field Descriptions .....     | 560 |
| 8-42. CM_PER_ELM_CLKCTRL Register Field Descriptions .....      | 561 |
| 8-43. CM_PER_I2C2_CLKCTRL Register Field Descriptions .....     | 562 |
| 8-44. CM_PER_I2C1_CLKCTRL Register Field Descriptions .....     | 563 |
| 8-45. CM_PER_SPI0_CLKCTRL Register Field Descriptions .....     | 564 |
| 8-46. CM_PER_SPI1_CLKCTRL Register Field Descriptions .....     | 565 |
| 8-47. CM_PER_L4LS_CLKCTRL Register Field Descriptions .....     | 566 |
| 8-48. CM_PER_L4FW_CLKCTRL Register Field Descriptions.....      | 567 |
| 8-49. CM_PER_MCASP1_CLKCTRL Register Field Descriptions .....   | 568 |
| 8-50. CM_PER_UART1_CLKCTRL Register Field Descriptions .....    | 569 |
| 8-51. CM_PER_UART2_CLKCTRL Register Field Descriptions .....    | 570 |
| 8-52. CM_PER_UART3_CLKCTRL Register Field Descriptions .....    | 571 |
| 8-53. CM_PER_UART4_CLKCTRL Register Field Descriptions .....    | 572 |
| 8-54. CM_PER_TIMER7_CLKCTRL Register Field Descriptions .....   | 573 |
| 8-55. CM_PER_TIMER2_CLKCTRL Register Field Descriptions .....   | 574 |
| 8-56. CM_PER_TIMER3_CLKCTRL Register Field Descriptions .....   | 575 |
| 8-57. CM_PER_TIMER4_CLKCTRL Register Field Descriptions .....   | 576 |
| 8-58. CM_PER_GPIO1_CLKCTRL Register Field Descriptions.....     | 577 |
| 8-59. CM_PER_GPIO2_CLKCTRL Register Field Descriptions.....     | 578 |
| 8-60. CM_PER_GPIO3_CLKCTRL Register Field Descriptions.....     | 579 |
| 8-61. CM_PER_TPCC_CLKCTRL Register Field Descriptions .....     | 580 |
| 8-62. CM_PER_DCAN0_CLKCTRL Register Field Descriptions.....     | 581 |
| 8-63. CM_PER_DCAN1_CLKCTRL Register Field Descriptions.....     | 582 |
| 8-64. CM_PER_EPWMSS1_CLKCTRL Register Field Descriptions .....  | 583 |
| 8-65. CM_PER_EMIF_FW_CLKCTRL Register Field Descriptions.....   | 584 |
| 8-66. CM_PER_EPWMSS0_CLKCTRL Register Field Descriptions .....  | 585 |
| 8-67. CM_PER_EPWMSS2_CLKCTRL Register Field Descriptions .....  | 586 |
| 8-68. CM_PER_L3_INSTR_CLKCTRL Register Field Descriptions ..... | 587 |
| 8-69. CM_PER_L3_CLKCTRL Register Field Descriptions.....        | 588 |
| 8-70. CM_PER_IEEE5000_CLKCTRL Register Field Descriptions.....  | 589 |
| 8-71. CM_PER_PRU_ICSS_CLKCTRL Register Field Descriptions ..... | 590 |
| 8-72. CM_PER_TIMER5_CLKCTRL Register Field Descriptions .....   | 591 |

|        |   |     |
|--------|---|-----|
| 8-73.  | CM_PER_TIMER6_CLKCTRL Register Field Descriptions .....         | 592 |
| 8-74.  | CM_PER_MMC1_CLKCTRL Register Field Descriptions .....           | 593 |
| 8-75.  | CM_PER_MMC2_CLKCTRL Register Field Descriptions .....           | 594 |
| 8-76.  | CM_PER_TPTC1_CLKCTRL Register Field Descriptions .....          | 595 |
| 8-77.  | CM_PER_TPTC2_CLKCTRL Register Field Descriptions .....          | 596 |
| 8-78.  | CM_PER_SPINLOCK_CLKCTRL Register Field Descriptions .....       | 597 |
| 8-79.  | CM_PER_MAILBOX0_CLKCTRL Register Field Descriptions .....       | 598 |
| 8-80.  | CM_PER_L4HS_CLKSTCTRL Register Field Descriptions .....         | 599 |
| 8-81.  | CM_PER_L4HS_CLKCTRL Register Field Descriptions .....           | 600 |
| 8-82.  | CM_PER_OCPWP_L3_CLKSTCTRL Register Field Descriptions .....     | 601 |
| 8-83.  | CM_PER_OCPWP_CLKCTRL Register Field Descriptions .....          | 602 |
| 8-84.  | CM_PER_PRU_ICSS_CLKSTCTRL Register Field Descriptions .....     | 603 |
| 8-85.  | CM_PER_CPSW_CLKSTCTRL Register Field Descriptions .....         | 604 |
| 8-86.  | CM_PER_LCDC_CLKSTCTRL Register Field Descriptions .....         | 605 |
| 8-87.  | CM_PER_CLKDIV32K_CLKCTRL Register Field Descriptions .....      | 606 |
| 8-88.  | CM_PER_CLK_24MHZ_CLKSTCTRL Register Field Descriptions .....    | 607 |
| 8-89.  | CM_WKUP REGISTERS .....   | 607 |
| 8-90.  | CM_WKUP_CLKSTCTRL Register Field Descriptions .....             | 611 |
| 8-91.  | CM_WKUP_CONTROL_CLKCTRL Register Field Descriptions .....       | 613 |
| 8-92.  | CM_WKUP_GPIO0_CLKCTRL Register Field Descriptions .....         | 614 |
| 8-93.  | CM_WKUP_L4WKUP_CLKCTRL Register Field Descriptions .....        | 615 |
| 8-94.  | CM_WKUP_TIMER0_CLKCTRL Register Field Descriptions .....        | 616 |
| 8-95.  | CM_WKUP_DEBUGSS_CLKCTRL Register Field Descriptions .....       | 617 |
| 8-96.  | CM_L3_AON_CLKSTCTRL Register Field Descriptions .....           | 618 |
| 8-97.  | CM_AUTOIDLE_DPLL_MPUMPU Register Field Descriptions .....       | 619 |
| 8-98.  | CM_IDLEST_DPLL_MPUMPU Register Field Descriptions .....         | 620 |
| 8-99.  | CM_SSC_DELTAMSTEP_DPLL_MPUMPU Register Field Descriptions ..... | 621 |
| 8-100. | CM_SSC_MODFREQDIV_DPLL_MPUMPU Register Field Descriptions ..... | 622 |
| 8-101. | CM_CLKSEL_DPLL_MPUMPU Register Field Descriptions .....         | 623 |
| 8-102. | CM_AUTOIDLE_DPLL_DDR Register Field Descriptions .....          | 624 |
| 8-103. | CM_IDLEST_DPLL_DDR Register Field Descriptions .....            | 625 |
| 8-104. | CM_SSC_DELTAMSTEP_DPLL_DDR Register Field Descriptions .....    | 626 |
| 8-105. | CM_SSC_MODFREQDIV_DPLL_DDR Register Field Descriptions .....    | 627 |
| 8-106. | CM_CLKSEL_DPLL_DDR Register Field Descriptions .....            | 628 |
| 8-107. | CM_AUTOIDLE_DPLL_DISP Register Field Descriptions .....         | 629 |
| 8-108. | CM_IDLEST_DPLL_DISP Register Field Descriptions .....           | 630 |
| 8-109. | CM_SSC_DELTAMSTEP_DPLL_DISP Register Field Descriptions .....   | 631 |
| 8-110. | CM_SSC_MODFREQDIV_DPLL_DISP Register Field Descriptions .....   | 632 |
| 8-111. | CM_CLKSEL_DPLL_DISP Register Field Descriptions .....           | 633 |
| 8-112. | CM_AUTOIDLE_DPLL_CORE Register Field Descriptions .....         | 634 |
| 8-113. | CM_IDLEST_DPLL_CORE Register Field Descriptions .....           | 635 |
| 8-114. | CM_SSC_DELTAMSTEP_DPLL_CORE Register Field Descriptions .....   | 636 |
| 8-115. | CM_SSC_MODFREQDIV_DPLL_CORE Register Field Descriptions .....   | 637 |
| 8-116. | CM_CLKSEL_DPLL_CORE Register Field Descriptions .....           | 638 |
| 8-117. | CM_AUTOIDLE_DPLL_PER Register Field Descriptions .....          | 639 |
| 8-118. | CM_IDLEST_DPLL_PER Register Field Descriptions .....            | 640 |
| 8-119. | CM_SSC_DELTAMSTEP_DPLL_PER Register Field Descriptions .....    | 641 |
| 8-120. | CM_SSC_MODFREQDIV_DPLL_PER Register Field Descriptions .....    | 642 |
| 8-121. | CM_CLKDCOLDO_DPLL_PER Register Field Descriptions .....         | 643 |

|   |     |
|---|-----|
| 8-122. CM_DIV_M4_DPLL_CORE Register Field Descriptions .....          | 644 |
| 8-123. CM_DIV_M5_DPLL_CORE Register Field Descriptions .....          | 645 |
| 8-124. CM_CLKMODE_DPLL_MPUMPU Register Field Descriptions.....        | 646 |
| 8-125. CM_CLKMODE_DPLL_PER Register Field Descriptions .....          | 648 |
| 8-126. CM_CLKMODE_DPLL_CORE Register Field Descriptions .....         | 649 |
| 8-127. CM_CLKMODE_DPLL_DDR Register Field Descriptions.....           | 651 |
| 8-128. CM_CLKMODE_DPLL_DISP Register Field Descriptions .....         | 653 |
| 8-129. CM_CLKSEL_DPLL_PERIPH Register Field Descriptions .....        | 655 |
| 8-130. CM_DIV_M2_DPLL_DDR Register Field Descriptions .....           | 656 |
| 8-131. CM_DIV_M2_DPLL_DISP Register Field Descriptions .....          | 657 |
| 8-132. CM_DIV_M2_DPLL_MPUMPU Register Field Descriptions .....        | 658 |
| 8-133. CM_DIV_M2_DPLL_PER Register Field Descriptions.....            | 659 |
| 8-134. CM_WKUP_WKUP_M3_CLKCTRL Register Field Descriptions .....      | 660 |
| 8-135. CM_WKUP_UART0_CLKCTRL Register Field Descriptions .....        | 661 |
| 8-136. CM_WKUP_I2C0_CLKCTRL Register Field Descriptions.....          | 662 |
| 8-137. CM_WKUP_ADC_TSC_CLKCTRL Register Field Descriptions.....       | 663 |
| 8-138. CM_WKUP_SMARTREFLEX0_CLKCTRL Register Field Descriptions ..... | 664 |
| 8-139. CM_WKUP_TIMER1_CLKCTRL Register Field Descriptions .....       | 665 |
| 8-140. CM_WKUP_SMARTREFLEX1_CLKCTRL Register Field Descriptions ..... | 666 |
| 8-141. CM_L4_WKUP_AON_CLKSTCTRL Register Field Descriptions .....     | 667 |
| 8-142. CM_WKUP_WDT1_CLKCTRL Register Field Descriptions.....          | 668 |
| 8-143. CM_DIV_M6_DPLL_CORE Register Field Descriptions .....          | 669 |
| 8-144. CM_DPLL REGISTERS.....   | 670 |
| 8-145. CLKSEL_TIMER7_CLK Register Field Descriptions .....            | 671 |
| 8-146. CLKSEL_TIMER2_CLK Register Field Descriptions .....            | 672 |
| 8-147. CLKSEL_TIMER3_CLK Register Field Descriptions .....            | 673 |
| 8-148. CLKSEL_TIMER4_CLK Register Field Descriptions .....            | 674 |
| 8-149. CM_MAC_CLKSEL Register Field Descriptions.....                 | 675 |
| 8-150. CLKSEL_TIMER5_CLK Register Field Descriptions .....            | 676 |
| 8-151. CLKSEL_TIMER6_CLK Register Field Descriptions .....            | 677 |
| 8-152. CM_CPTS_RFT_CLKSEL Register Field Descriptions .....           | 678 |
| 8-153. CLKSEL_TIMER1MS_CLK Register Field Descriptions .....          | 679 |
| 8-154. CLKSEL_GFX_FCLK Register Field Descriptions.....               | 680 |
| 8-155. CLKSEL_PRU_ICSS_OCP_CLK Register Field Descriptions .....      | 681 |
| 8-156. CLKSEL_LCDC_PIXEL_CLK Register Field Descriptions .....        | 682 |
| 8-157. CLKSEL_WDT1_CLK Register Field Descriptions .....              | 683 |
| 8-158. CLKSEL_GPIO0_DBCLK Register Field Descriptions.....            | 684 |
| 8-159. CM_MPUMPU REGISTERS .....                                      | 684 |
| 8-160. CM_MPUMPU_CLKSTCTRL Register Field Descriptions .....          | 685 |
| 8-161. CM_MPUMPU_CLKCTRL Register Field Descriptions .....            | 686 |
| 8-162. CM_DEVICE REGISTERS .....                                      | 686 |
| 8-163. CM_CLKOUT_CTRL Register Field Descriptions .....               | 688 |
| 8-164. CM_RTC REGISTERS .....   | 689 |
| 8-165. CM_RTC_RTC_CLKCTRL Register Field Descriptions .....           | 690 |
| 8-166. CM_RTC_CLKSTCTRL Register Field Descriptions .....             | 691 |
| 8-167. CM_GFX REGISTERS .....   | 692 |
| 8-168. CM_GFX_L3_CLKSTCTRL Register Field Descriptions .....          | 693 |
| 8-169. CM_GFX_GFX_CLKCTRL Register Field Descriptions .....           | 694 |
| 8-170. CM_GFX_L4LS_GFX_CLKSTCTRL Register Field Descriptions .....    | 695 |

|        |   |     |
|--------|---|-----|
| 8-171. | CM_GFX_MMUCFG_CLKCTRL Register Field Descriptions.....      | 696 |
| 8-172. | CM_GFX_MMUDATA_CLKCTRL Register Field Descriptions .....    | 697 |
| 8-173. | CM_CEFUSE REGISTERS .....                                   | 697 |
| 8-174. | CM_CEFUSE_CLKSTCTRL Register Field Descriptions .....       | 699 |
| 8-175. | CM_CEFUSE_CEFUSE_CLKCTRL Register Field Descriptions .....  | 700 |
| 8-176. | PRM_IRQ REGISTERS.....                                      | 701 |
| 8-177. | REVISION_PRM Register Field Descriptions .....              | 702 |
| 8-178. | PRM_IRQSTATUS_MPU Register Field Descriptions .....         | 703 |
| 8-179. | PRM_IRQENABLE_MPU Register Field Descriptions .....         | 704 |
| 8-180. | PRM_IRQSTATUS_M3 Register Field Descriptions .....          | 705 |
| 8-181. | PRM_IRQENABLE_M3 Register Field Descriptions .....          | 706 |
| 8-182. | PRM_PER REGISTERS .....                                     | 707 |
| 8-183. | RM_PER_RSTCTRL Register Field Descriptions .....            | 708 |
| 8-184. | PM_PER_PWRSTST Register Field Descriptions.....             | 709 |
| 8-185. | PM_PER_PWRSTCTRL Register Field Descriptions .....          | 710 |
| 8-186. | PRM_WKUP REGISTERS .....                                    | 711 |
| 8-187. | RM_WKUP_RSTCTRL Register Field Descriptions.....            | 712 |
| 8-188. | PM_WKUP_PWRSTCTRL Register Field Descriptions .....         | 713 |
| 8-189. | PM_WKUP_PWRSTST Register Field Descriptions .....           | 714 |
| 8-190. | RM_WKUP_RSTST Register Field Descriptions .....             | 715 |
| 8-191. | PRM_MPREGISTERS.....  | 715 |
| 8-192. | PM_MPREGISTERS_Register Field Descriptions .....            | 717 |
| 8-193. | PM_MPREGISTERS_Register Field Descriptions .....            | 719 |
| 8-194. | RM_MPREGISTERS_Register Field Descriptions .....            | 720 |
| 8-195. | PRM_DEVICE REGISTERS.....                                   | 720 |
| 8-196. | PRM_RSTCTRL Register Field Descriptions .....               | 722 |
| 8-197. | PRM_RSTTIME Register Field Descriptions .....               | 723 |
| 8-198. | PRM_RSTST Register Field Descriptions .....                 | 724 |
| 8-199. | PRM_SRAM_COUNT Register Field Descriptions .....            | 725 |
| 8-200. | PRM_LDO_SRAM_CORE_SETUP Register Field Descriptions .....   | 726 |
| 8-201. | PRM_LDO_SRAM_CORE_CTRL Register Field Descriptions .....    | 728 |
| 8-202. | PRM_LDO_SRAM_MPREGISTERS_Register Field Descriptions .....  | 729 |
| 8-203. | PRM_LDO_SRAM_MPREGISTERS_Register Field Descriptions .....  | 731 |
| 8-204. | PRM_RTC REGISTERS .....                                     | 731 |
| 8-205. | PM_RTC_PWRSTCTRL Register Field Descriptions .....          | 733 |
| 8-206. | PM_RTC_PWRSTST Register Field Descriptions.....             | 734 |
| 8-207. | PRM_GFX REGISTERS .....                                     | 734 |
| 8-208. | PM_GFX_PWRSTCTRL Register Field Descriptions .....          | 736 |
| 8-209. | RM_GFX_RSTCTRL Register Field Descriptions .....            | 737 |
| 8-210. | PM_GFX_PWRSTST Register Field Descriptions.....             | 738 |
| 8-211. | RM_GFX_RSTST Register Field Descriptions.....               | 739 |
| 8-212. | PRM_CEFUSE REGISTERS.....                                   | 739 |
| 8-213. | PM_CEFUSE_PWRSTCTRL Register Field Descriptions.....        | 740 |
| 8-214. | PM_CEFUSE_PWRSTST Register Field Descriptions .....         | 741 |
| 9-1.   | Pad Control Register Field Descriptions .....               | 743 |
| 9-2.   | Mode Selection .....  | 744 |
| 9-3.   | Pull Selection .....  | 744 |
| 9-4.   | Interconnect Priority Values .....                          | 746 |
| 9-5.   | Available Sources for Timer[5–7] and eCAP[0–2] Events ..... | 749 |

|       |   |     |
|-------|---|-----|
| 9-6.  | Selection Mux Values .....                                  | 751 |
| 9-7.  | DDR Slew Rate Control Settings .....                        | 752 |
| 9-8.  | DDR Impedance Control Settings .....                        | 752 |
| 9-9.  | DDR PHY to IO Pin Mapping.....                              | 752 |
| 9-10. | CONTROL_MODULE REGISTERS .....                              | 753 |
| 9-11. | control_revision Register Field Descriptions .....          | 759 |
| 9-12. | device_id Register Field Descriptions .....                 | 760 |
| 9-13. | control_hwinfo Register Field Descriptions .....            | 761 |
| 9-14. | control_sysconfig Register Field Descriptions .....         | 762 |
| 9-15. | control_status Register Field Descriptions.....             | 763 |
| 9-16. | control_emif_sdram_config Register Field Descriptions ..... | 764 |
| 9-17. | core_sldo_ctrl Register Field Descriptions.....             | 766 |
| 9-18. | mpu_sldo_ctrl Register Field Descriptions.....              | 767 |
| 9-19. | clk32kdivratio_ctrl Register Field Descriptions .....       | 768 |
| 9-20. | bandgap_ctrl Register Field Descriptions.....               | 769 |
| 9-21. | bandgap_trim Register Field Descriptions .....              | 770 |
| 9-22. | pll_clkinpulow_ctrl Register Field Descriptions.....        | 771 |
| 9-23. | mosc_ctrl Register Field Descriptions .....                 | 772 |
| 9-24. | rcosc_ctrl Register Field Descriptions .....                | 773 |
| 9-25. | deepsleep_ctrl Register Field Descriptions .....            | 774 |
| 9-26. | dev_feature Register Field Descriptions .....               | 775 |
| 9-27. | init_priority_0 Register Field Descriptions .....           | 776 |
| 9-28. | init_priority_1 Register Field Descriptions .....           | 777 |
| 9-29. | mmu_cfg Register Field Descriptions .....                   | 778 |
| 9-30. | tptc_cfg Register Field Descriptions .....                  | 779 |
| 9-31. | usb_ctrl0 Register Field Descriptions .....                 | 780 |
| 9-32. | usb_sts0 Register Field Descriptions .....                  | 782 |
| 9-33. | usb_ctrl1 Register Field Descriptions .....                 | 783 |
| 9-34. | usb_sts1 Register Field Descriptions .....                  | 785 |
| 9-35. | mac_id0_lo Register Field Descriptions.....                 | 786 |
| 9-36. | mac_id0_hi Register Field Descriptions.....                 | 787 |
| 9-37. | mac_id1_lo Register Field Descriptions.....                 | 788 |
| 9-38. | mac_id1_hi Register Field Descriptions.....                 | 789 |
| 9-39. | dcan_raminit Register Field Descriptions .....              | 790 |
| 9-40. | usb_wkup_ctrl Register Field Descriptions .....             | 791 |
| 9-41. | gmii_sel Register Field Descriptions .....                  | 792 |
| 9-42. | pwmss_ctrl Register Field Descriptions .....                | 793 |
| 9-43. | mreqprio_0 Register Field Descriptions.....                 | 794 |
| 9-44. | mreqprio_1 Register Field Descriptions.....                 | 795 |
| 9-45. | hw_event_sel_grp1 Register Field Descriptions .....         | 796 |
| 9-46. | hw_event_sel_grp2 Register Field Descriptions .....         | 797 |
| 9-47. | hw_event_sel_grp3 Register Field Descriptions .....         | 798 |
| 9-48. | hw_event_sel_grp4 Register Field Descriptions .....         | 799 |
| 9-49. | smrt_ctrl Register Field Descriptions .....                 | 800 |
| 9-50. | mpuss_hw_debug_sel Register Field Descriptions .....        | 801 |
| 9-51. | mpuss_hw_dbg_info Register Field Descriptions.....          | 802 |
| 9-52. | vdd_mpu_opp_050 Register Field Descriptions .....           | 803 |
| 9-53. | vdd_mpu_opp_100 Register Field Descriptions .....           | 804 |
| 9-54. | vdd_mpu_opp_120 Register Field Descriptions .....           | 805 |

|        |   |     |
|--------|---|-----|
| 9-55.  | vdd_mpu_opp_turbo Register Field Descriptions.....    | 806 |
| 9-56.  | vdd_core_opp_050 Register Field Descriptions .....    | 807 |
| 9-57.  | vdd_core_opp_100 Register Field Descriptions .....    | 808 |
| 9-58.  | bb_scale Register Field Descriptions .....            | 809 |
| 9-59.  | usb_vid_pid Register Field Descriptions .....         | 810 |
| 9-60.  | conf_<module>_<pin> Register Field Descriptions ..... | 811 |
| 9-61.  | cqdetect_status Register Field Descriptions .....     | 812 |
| 9-62.  | ddr_io_ctrl Register Field Descriptions.....          | 813 |
| 9-63.  | vtp_ctrl Register Field Descriptions.....             | 814 |
| 9-64.  | vref_ctrl Register Field Descriptions.....            | 815 |
| 9-65.  | tpcc_evt_mux_0_3 Register Field Descriptions.....     | 816 |
| 9-66.  | tpcc_evt_mux_4_7 Register Field Descriptions.....     | 817 |
| 9-67.  | tpcc_evt_mux_8_11 Register Field Descriptions .....   | 818 |
| 9-68.  | tpcc_evt_mux_12_15 Register Field Descriptions.....   | 819 |
| 9-69.  | tpcc_evt_mux_16_19 Register Field Descriptions.....   | 820 |
| 9-70.  | tpcc_evt_mux_20_23 Register Field Descriptions .....  | 821 |
| 9-71.  | tpcc_evt_mux_24_27 Register Field Descriptions .....  | 822 |
| 9-72.  | tpcc_evt_mux_28_31 Register Field Descriptions.....   | 823 |
| 9-73.  | tpcc_evt_mux_32_35 Register Field Descriptions.....   | 824 |
| 9-74.  | tpcc_evt_mux_36_39 Register Field Descriptions.....   | 825 |
| 9-75.  | tpcc_evt_mux_40_43 Register Field Descriptions.....   | 826 |
| 9-76.  | tpcc_evt_mux_44_47 Register Field Descriptions.....   | 827 |
| 9-77.  | tpcc_evt_mux_48_51 Register Field Descriptions .....  | 828 |
| 9-78.  | tpcc_evt_mux_52_55 Register Field Descriptions.....   | 829 |
| 9-79.  | tpcc_evt_mux_56_59 Register Field Descriptions .....  | 830 |
| 9-80.  | tpcc_evt_mux_60_63 Register Field Descriptions .....  | 831 |
| 9-81.  | timer_evt_capt Register Field Descriptions.....       | 832 |
| 9-82.  | ecap_evt_capt Register Field Descriptions .....       | 833 |
| 9-83.  | adc_evt_capt Register Field Descriptions .....        | 834 |
| 9-84.  | reset_iso Register Field Descriptions .....           | 835 |
| 9-85.  | ddr_cke_ctrl Register Field Descriptions.....         | 836 |
| 9-86.  | sma2 Register Field Descriptions .....                | 837 |
| 9-87.  | m3_txev_eoi Register Field Descriptions .....         | 838 |
| 9-88.  | ipc_msg_reg0 Register Field Descriptions .....        | 839 |
| 9-89.  | ipc_msg_reg1 Register Field Descriptions .....        | 840 |
| 9-90.  | ipc_msg_reg2 Register Field Descriptions .....        | 841 |
| 9-91.  | ipc_msg_reg3 Register Field Descriptions .....        | 842 |
| 9-92.  | ipc_msg_reg4 Register Field Descriptions .....        | 843 |
| 9-93.  | ipc_msg_reg5 Register Field Descriptions .....        | 844 |
| 9-94.  | ipc_msg_reg6 Register Field Descriptions .....        | 845 |
| 9-95.  | ipc_msg_reg7 Register Field Descriptions .....        | 846 |
| 9-96.  | ddr_cmd0_ioctl Register Field Descriptions .....      | 847 |
| 9-97.  | ddr_cmd1_ioctl Register Field Descriptions .....      | 849 |
| 9-98.  | ddr_cmd2_ioctl Register Field Descriptions .....      | 851 |
| 9-99.  | ddr_data0_ioctl Register Field Descriptions .....     | 853 |
| 9-100. | ddr_data1_ioctl Register Field Descriptions .....     | 855 |
| 10-1.  | L3 Master — Slave Connectivity .....                  | 860 |
| 10-2.  | MConnID Assignment .....                              | 860 |
| 11-1.  | TPCC Connectivity Attributes .....                    | 867 |

|        |  |     |
|--------|--|-----|
| 11-2.  | TPCC Clock Signals .....   | 867 |
| 11-3.  | TPTC Connectivity Attributes.....  | 868 |
| 11-4.  | TPTC Clock Signals .....   | 868 |
| 11-5.  | EDMA3 Parameter RAM Contents .....   | 876 |
| 11-6.  | EDMA3 Channel Parameter Description.....   | 878 |
| 11-7.  | Channel Options Parameters (OPT) Field Descriptions .....                                | 879 |
| 11-8.  | Dummy and Null Transfer Request.....   | 883 |
| 11-9.  | Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set) .....                   | 884 |
| 11-10. | Expected Number of Transfers for Non-Null Transfer .....                                 | 890 |
| 11-11. | Shadow Region Registers .....  | 894 |
| 11-12. | Chain Event Triggers .....   | 896 |
| 11-13. | EDMA3 Transfer Completion Interrupts.....  | 896 |
| 11-14. | EDMA3 Error Interrupts .....   | 896 |
| 11-15. | Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping.....                           | 897 |
| 11-16. | Number of Interrupts.....  | 897 |
| 11-17. | Allowed Accesses.....  | 902 |
| 11-18. | MPPA Registers to Region Assignment .....  | 902 |
| 11-19. | Example Access Denied .....  | 903 |
| 11-20. | Example Access Allowed .....   | 904 |
| 11-21. | Read/Write Command Optimization Rules .....  | 908 |
| 11-22. | EDMA3 Transfer Controller Configurations.....  | 910 |
| 11-23. | Direct Mapped .....  | 930 |
| 11-24. | Crossbar Mapped .....  | 931 |
| 11-25. | EDMACC Registers.....  | 933 |
| 11-26. | Peripheral ID Register (PID) Field Descriptions .....                                    | 936 |
| 11-27. | EDMA3CC Configuration Register (CCCFG) Field Descriptions .....                          | 937 |
| 11-28. | EDMA3CC System Configuration Register (SYSCONFIG) Field Descriptions .....               | 939 |
| 11-29. | DMA Channel Map <i>n</i> Registers (DCHMAP <i>n</i> ) Field Descriptions .....           | 940 |
| 11-30. | QDMA Channel Map <i>n</i> Registers (QCHMAP <i>n</i> ) Field Descriptions .....          | 941 |
| 11-31. | DMA Channel Queue <i>n</i> Number Registers (DMAQNUM <i>n</i> ) Field Descriptions ..... | 942 |
| 11-32. | Bits in DMAQNUM <i>n</i> .....   | 942 |
| 11-33. | QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions .....                   | 943 |
| 11-34. | Queue Priority Register (QUEPRI) Field Descriptions .....                                | 944 |
| 11-35. | Event Missed Register (EMR) Field Descriptions .....                                     | 945 |
| 11-36. | Event Missed Register High (EMRH) Field Descriptions .....                               | 945 |
| 11-37. | Event Missed Clear Register (EMCR) Field Descriptions .....                              | 946 |
| 11-38. | Event Missed Clear Register High (EMCRH) Field Descriptions .....                        | 946 |
| 11-39. | QDMA Event Missed Register (QEMR) Field Descriptions .....                               | 947 |
| 11-40. | QDMA Event Missed Clear Register (QEMCR) Field Descriptions .....                        | 948 |
| 11-41. | EDMA3CC Error Register (CCERR) Field Descriptions .....                                  | 949 |
| 11-42. | EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions .....                         | 949 |
| 11-43. | Error Evaluation Register (EEVAL) Field Descriptions.....                                | 951 |
| 11-44. | DMA Region Access Enable Registers for Region M (DRAEm/DRAEHm) Field Descriptions .....  | 952 |
| 11-45. | QDMA Region Access Enable for Region M (QRAEm) Field Descriptions .....                  | 953 |
| 11-46. | Event Queue Entry Registers (QxEy) Field Descriptions .....                              | 954 |
| 11-47. | Queue Status Register <i>n</i> (QSTAT <i>n</i> ) Field Descriptions .....                | 955 |
| 11-48. | Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions .....                  | 956 |
| 11-49. | EDMA3CC Status Register (CCSTAT) Field Descriptions .....                                | 957 |
| 11-50. | Memory Protection Fault Address Register (MPFAR) Field Descriptions .....                | 959 |

|   |     |
|---|-----|
| 11-51. Memory Protection Fault Status Register (MPFSR) Field Descriptions .....   | 960 |
| 11-52. Memory Protection Fault Command Register (MPFCR) Field Descriptions .....  | 961 |
| 11-53. Memory Protection Page Attribute Register (MPPAn) Field Descriptions ..... | 962 |
| 11-54. Event Register (ER) Field Descriptions .....                               | 964 |
| 11-55. Event Register High (ERH) Field Descriptions.....                          | 964 |
| 11-56. Event Clear Register (ECR) Field Descriptions .....                        | 965 |
| 11-57. Event Clear Register High (ECRH) Field Descriptions.....                   | 965 |
| 11-58. Event Set Register (ESR) Field Descriptions .....                          | 966 |
| 11-59. Event Set Register High (ESRH) Field Descriptions .....                    | 967 |
| 11-60. Chained Event Register (CER) Field Descriptions .....                      | 968 |
| 11-61. Chained Event Register High (CERH) Field Descriptions .....                | 969 |
| 11-62. Event Enable Register (EER) Field Descriptions .....                       | 970 |
| 11-63. Event Enable Register High (EERH) Field Descriptions.....                  | 970 |
| 11-64. Event Enable Clear Register (EECR) Field Descriptions .....                | 971 |
| 11-65. Event Enable Clear Register High (EECRH) Field Descriptions.....           | 971 |
| 11-66. Event Enable Set Register (EESR) Field Descriptions .....                  | 972 |
| 11-67. Event Enable Set Register High (EESRH) Field Descriptions .....            | 972 |
| 11-68. Secondary Event Register (SER) Field Descriptions.....                     | 973 |
| 11-69. Secondary Event Register High (SERH) Field Descriptions .....              | 973 |
| 11-70. Secondary Event Clear Register (SECR) Field Descriptions .....             | 974 |
| 11-71. Secondary Event Clear Register High (SECRH) Field Descriptions .....       | 974 |
| 11-72. Interrupt Enable Register (IER) Field Descriptions .....                   | 975 |
| 11-73. Interrupt Enable Register High (IERH) Field Descriptions.....              | 975 |
| 11-74. Interrupt Enable Clear Register (IECR) Field Descriptions .....            | 976 |
| 11-75. Interrupt Enable Clear Register High (IECRH) Field Descriptions .....      | 976 |
| 11-76. Interrupt Enable Set Register (IESR) Field Descriptions .....              | 977 |
| 11-77. Interrupt Enable Set Register High (IESRH) Field Descriptions .....        | 977 |
| 11-78. Interrupt Pending Register (IPR) Field Descriptions .....                  | 978 |
| 11-79. Interrupt Pending Register High (IPRH) Field Descriptions .....            | 978 |
| 11-80. Interrupt Clear Register (ICR) Field Descriptions .....                    | 979 |
| 11-81. Interrupt Clear Register High (ICRH) Field Descriptions .....              | 979 |
| 11-82. Interrupt Evaluate Register (IEVAL) Field Descriptions.....                | 980 |
| 11-83. QDMA Event Register (QER) Field Descriptions .....                         | 981 |
| 11-84. QDMA Event Enable Register (QEER) Field Descriptions .....                 | 982 |
| 11-85. QDMA Event Enable Clear Register (QEECR) Field Descriptions .....          | 983 |
| 11-86. QDMA Event Enable Set Register (QEESR) Field Descriptions .....            | 984 |
| 11-87. QDMA Secondary Event Register (QSER) Field Descriptions.....               | 985 |
| 11-88. QDMA Secondary Event Clear Register (QSECER) Field Descriptions .....      | 986 |
| 11-89. EDMA3TC Registers .....  | 987 |
| 11-90. Peripheral ID Register (PID) Field Descriptions .....                      | 988 |
| 11-91. EDMA3TC Configuration Register (TCCFG) Field Descriptions.....             | 989 |
| 11-92. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions .....          | 990 |
| 11-93. Error Register (ERRSTAT) Field Descriptions .....                          | 992 |
| 11-94. Error Enable Register (ERREN) Field Descriptions .....                     | 993 |
| 11-95. Error Clear Register (ERRCLR) Field Descriptions .....                     | 994 |
| 11-96. Error Details Register (ERRDET) Field Descriptions.....                    | 995 |
| 11-97. Error Interrupt Command Register (ERRCMD) Field Descriptions .....         | 996 |
| 11-98. Read Rate Register (RDRATE) Field Descriptions.....                        | 997 |
| 11-99. Source Active Options Register (SAOPT) Field Descriptions.....             | 998 |

|   |      |
|---|------|
| 11-100. Source Active Source Address Register (SASRC) Field Descriptions .....                              | 1000 |
| 11-101. Source Active Count Register (SACNT) Field Descriptions .....                                       | 1000 |
| 11-102. Source Active Destination Address Register (SADST) Field Descriptions .....                         | 1001 |
| 11-103. Source Active Source B-Dimension Index Register (SABIDX) Field Descriptions .....                   | 1001 |
| 11-104. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions .....                  | 1002 |
| 11-105. Source Active Count Reload Register (SACNTRLD) Field Descriptions .....                             | 1003 |
| 11-106. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions .....              | 1003 |
| 11-107. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions .....         | 1004 |
| 11-108. Destination FIFO Options Register (DFOPT $n$ ) Field Descriptions .....                             | 1005 |
| 11-109. Destination FIFO Source Address Register (DFSRC $n$ ) Field Descriptions .....                      | 1007 |
| 11-110. Destination FIFO Count Register (DFCNT $n$ ) Field Descriptions .....                               | 1007 |
| 11-111. Destination FIFO Destination Address Register (DFDST $n$ ) Field Descriptions .....                 | 1008 |
| 11-112. Destination FIFO B-Index Register (DFBIDX $n$ ) Field Descriptions .....                            | 1008 |
| 11-113. Destination FIFO Memory Protection Proxy Register (DFMPPRXY $n$ ) Field Descriptions .....          | 1009 |
| 11-114. Destination FIFO Count Reload Register (DFCNTRLD $n$ ) Field Descriptions .....                     | 1010 |
| 11-115. Destination FIFO Source Address B-Reference Register (DFSRCBREF $n$ ) Field Descriptions .....      | 1010 |
| 11-116. Destination FIFO Destination Address B-Reference Register (DFDSTBREF $n$ ) Field Descriptions ..... | 1011 |
| 11-117. Debug List.....   | 1011 |
| 12-1. TSC_ADC Connectivity Attributes.....  | 1017 |
| 12-2. TSC_ADC Clock Signals .....   | 1018 |
| 12-3. TSC_ADC Pin List.....   | 1018 |
| 12-4. TSC_ADC_SS REGISTERS .....  | 1025 |
| 12-5. REVISION Register Field Descriptions .....  | 1027 |
| 12-6. SYSCONFIG Register Field Descriptions .....   | 1028 |
| 12-7. IRQSTATUS_RAW Register Field Descriptions.....  | 1029 |
| 12-8. IRQSTATUS Register Field Descriptions .....   | 1031 |
| 12-9. IRQENABLE_SET Register Field Descriptions.....  | 1033 |
| 12-10. IRQENABLE_CLR Register Field Descriptions .....  | 1035 |
| 12-11. IRQWAKEUP Register Field Descriptions .....  | 1037 |
| 12-12. DMAENABLE_SET Register Field Descriptions .....  | 1038 |
| 12-13. DMAENABLE_CLR Register Field Descriptions .....  | 1039 |
| 12-14. CTRL Register Field Descriptions .....   | 1040 |
| 12-15. ADCSTAT Register Field Descriptions .....  | 1041 |
| 12-16. ADCRANGE Register Field Descriptions.....  | 1042 |
| 12-17. ADC_CLKDIV Register Field Descriptions .....   | 1043 |
| 12-18. ADC_MISC Register Field Descriptions .....   | 1044 |
| 12-19. STEPENABLE Register Field Descriptions .....   | 1045 |
| 12-20. IDLECONFIG Register Field Descriptions .....   | 1046 |
| 12-21. TS_CHARGE_STEPCONFIG Register Field Descriptions .....   | 1047 |
| 12-22. TS_CHARGE_DELAY Register Field Descriptions .....  | 1048 |
| 12-23. STEPCONFIG1 Register Field Descriptions .....  | 1049 |
| 12-24. STEPDELAY1 Register Field Descriptions .....   | 1050 |
| 12-25. STEPCONFIG2 Register Field Descriptions .....  | 1051 |
| 12-26. STEPDELAY2 Register Field Descriptions .....   | 1052 |
| 12-27. STEPCONFIG3 Register Field Descriptions .....  | 1053 |
| 12-28. STEPDELAY3 Register Field Descriptions .....   | 1054 |
| 12-29. STEPCONFIG4 Register Field Descriptions .....  | 1055 |
| 12-30. STEPDELAY4 Register Field Descriptions .....   | 1056 |
| 12-31. STEPCONFIG5 Register Field Descriptions .....  | 1057 |

---

|  |      |
|--|------|
| 12-32. STEPDELAY5 Register Field Descriptions .....              | 1058 |
| 12-33. STEPCONFIG6 Register Field Descriptions .....             | 1059 |
| 12-34. STEPDELAY6 Register Field Descriptions .....              | 1060 |
| 12-35. STEPCONFIG7 Register Field Descriptions .....             | 1061 |
| 12-36. STEPDELAY7 Register Field Descriptions .....              | 1062 |
| 12-37. STEPCONFIG8 Register Field Descriptions .....             | 1063 |
| 12-38. STEPDELAY8 Register Field Descriptions .....              | 1064 |
| 12-39. STEPCONFIG9 Register Field Descriptions .....             | 1065 |
| 12-40. STEPDELAY9 Register Field Descriptions .....              | 1066 |
| 12-41. STEPCONFIG10 Register Field Descriptions .....            | 1067 |
| 12-42. STEPDELAY10 Register Field Descriptions .....             | 1068 |
| 12-43. STEPCONFIG11 Register Field Descriptions .....            | 1069 |
| 12-44. STEPDELAY11 Register Field Descriptions .....             | 1070 |
| 12-45. STEPCONFIG12 Register Field Descriptions .....            | 1071 |
| 12-46. STEPDELAY12 Register Field Descriptions .....             | 1072 |
| 12-47. STEPCONFIG13 Register Field Descriptions .....            | 1073 |
| 12-48. STEPDELAY13 Register Field Descriptions .....             | 1074 |
| 12-49. STEPCONFIG14 Register Field Descriptions .....            | 1075 |
| 12-50. STEPDELAY14 Register Field Descriptions .....             | 1076 |
| 12-51. STEPCONFIG15 Register Field Descriptions .....            | 1077 |
| 12-52. STEPDELAY15 Register Field Descriptions .....             | 1078 |
| 12-53. STEPCONFIG16 Register Field Descriptions .....            | 1079 |
| 12-54. STEPDELAY16 Register Field Descriptions .....             | 1080 |
| 12-55. FIFO0COUNT Register Field Descriptions.....               | 1081 |
| 12-56. FIFO0THRESHOLD Register Field Descriptions.....           | 1082 |
| 12-57. DMA0REQ Register Field Descriptions .....                 | 1083 |
| 12-58. FIFO1COUNT Register Field Descriptions.....               | 1084 |
| 12-59. FIFO1THRESHOLD Register Field Descriptions.....           | 1085 |
| 12-60. DMA1REQ Register Field Descriptions .....                 | 1086 |
| 12-61. FIFO0DATA Register Field Descriptions .....               | 1087 |
| 12-62. FIFO1DATA Register Field Descriptions .....               | 1088 |
| 13-1. LCD Controller Connectivity Attributes .....               | 1092 |
| 13-2. LCD Controller Clock Signals .....                         | 1093 |
| 13-3. LCD Controller Pin List .....                              | 1093 |
| 13-4. LCD External I/O Signals .....                             | 1096 |
| 13-5. Register Configuration for DMA Engine Programming .....    | 1097 |
| 13-6. LIDD I/O Name Map .....                                    | 1099 |
| 13-7. Operation Modes Supported by Raster Controller .....       | 1100 |
| 13-8. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer .....   | 1102 |
| 13-9. Frame Buffer Size According to BPP .....                   | 1103 |
| 13-10. Color/Grayscale Intensities and Modulation Rates .....    | 1107 |
| 13-11. Number of Colors/Shades of Gray Available on Screen ..... | 1107 |
| 13-12. Highlander 0.8 Interrupt Module Control Registers .....   | 1111 |
| 13-13. LCD REGISTERS .....                                       | 1120 |
| 13-14. PID Register Field Descriptions .....                     | 1122 |
| 13-15. CTRL Register Field Descriptions.....                     | 1123 |
| 13-16. LIDD_CTRL Register Field Descriptions .....               | 1124 |
| 13-17. LIDD_CS0_CONF Register Field Descriptions.....            | 1125 |
| 13-18. LIDD_CS0_ADDR Register Field Descriptions.....            | 1126 |

|   |      |
|---|------|
| 13-19. LIDD_CS0_DATA Register Field Descriptions .....                  | 1127 |
| 13-20. LIDD_CS1_CONF Register Field Descriptions.....                   | 1128 |
| 13-21. LIDD_CS1_ADDR Register Field Descriptions.....                   | 1129 |
| 13-22. LIDD_CS1_DATA Register Field Descriptions .....                  | 1130 |
| 13-23. RASTER_CTRL Register Field Descriptions .....                    | 1131 |
| 13-24. RASTER_TIMING_0 Register Field Descriptions .....                | 1133 |
| 13-25. RASTER_TIMING_1 Register Field Descriptions .....                | 1134 |
| 13-26. RASTER_TIMING_2 Register Field Descriptions .....                | 1135 |
| 13-27. RASTER_SUBPANEL Register Field Descriptions .....                | 1137 |
| 13-28. RASTER_SUBPANEL2 Register Field Descriptions .....               | 1138 |
| 13-29. LCDDMA_CTRL Register Field Descriptions .....                    | 1139 |
| 13-30. LCDDMA_FB0_BASE Register Field Descriptions .....                | 1140 |
| 13-31. LCDDMA_FB0_CEILING Register Field Descriptions.....              | 1141 |
| 13-32. LCDDMA_FB1_BASE Register Field Descriptions .....                | 1142 |
| 13-33. LCDDMA_FB1_CEILING Register Field Descriptions.....              | 1143 |
| 13-34. SYSCONFIG Register Field Descriptions .....                      | 1144 |
| 13-35. IRQSTATUS_RAW Register Field Descriptions.....                   | 1145 |
| 13-36. IRQSTATUS Register Field Descriptions .....                      | 1147 |
| 13-37. IRQENABLE_SET Register Field Descriptions.....                   | 1149 |
| 13-38. IRQENABLE_CLEAR Register Field Descriptions .....                | 1151 |
| 13-39. CLKC_ENABLE Register Field Descriptions .....                    | 1153 |
| 13-40. CLKC_RESET Register Field Descriptions .....                     | 1154 |
| 14-1. Unsupported CPGMAC Features .....                                 | 1157 |
| 14-2. Ethernet Switch Connectivity Attributes .....                     | 1159 |
| 14-3. Ethernet Switch Clock Signals .....                               | 1160 |
| 14-4. Ethernet Switch Pin List .....                                    | 1161 |
| 14-5. GMII Interface Signal Descriptions in GIG (1000Mbps) Mode .....   | 1163 |
| 14-6. GMII Interface Signal Descriptions in MII (100/10Mbps) Mode ..... | 1164 |
| 14-7. RMII Interface Signal Descriptions .....                          | 1165 |
| 14-8. RGMII Interface Signal Descriptions .....                         | 1167 |
| 14-9. VLAN Header Encapsulation Word Field Descriptions .....           | 1186 |
| 14-10. Learned Address Control Bits.....                                | 1187 |
| 14-11. Free (Unused) Address Table Entry Bit Values .....               | 1187 |
| 14-12. Multicast Address Table Entry Bit Values .....                   | 1188 |
| 14-13. VLAN/Multicast Address Table Entry Bit Values .....              | 1188 |
| 14-14. Unicast Address Table Entry Bit Values.....                      | 1189 |
| 14-15. OUI Unicast Address Table Entry Bit Values .....                 | 1190 |
| 14-16. Unicast Address Table Entry Bit Values.....                      | 1191 |
| 14-17. VLAN Table Entry .....   | 1192 |
| 14-18. Operations of Emulation Control Input and Register Bits .....    | 1202 |
| 14-19. Rx Statistics Summary.....                                       | 1211 |
| 14-20. Tx Statistics Summary .....                                      | 1212 |
| 14-21. Values of messageType field .....                                | 1225 |
| 14-22. MDIO Read Frame Format.....                                      | 1226 |
| 14-23. MDIO Write Frame Format.....                                     | 1226 |
| 14-24. CPSW_ALE REGISTERS.....  | 1233 |
| 14-25. IDVER Register Field Descriptions .....                          | 1234 |
| 14-26. CONTROL Register Field Descriptions.....                         | 1235 |
| 14-27. PRESCALE Register Field Descriptions .....                       | 1237 |

---

|   |      |
|---|------|
| 14-28. UNKNOWN_VLAN Register Field Descriptions .....       | 1238 |
| 14-29. TBLCTL Register Field Descriptions .....             | 1239 |
| 14-30. TBLW2 Register Field Descriptions .....              | 1240 |
| 14-31. TBLW1 Register Field Descriptions .....              | 1241 |
| 14-32. TBLW0 Register Field Descriptions .....              | 1242 |
| 14-33. PORTCTL0 Register Field Descriptions .....           | 1243 |
| 14-34. PORTCTL1 Register Field Descriptions .....           | 1244 |
| 14-35. PORTCTL2 Register Field Descriptions .....           | 1245 |
| 14-36. PORTCTL3 Register Field Descriptions .....           | 1246 |
| 14-37. PORTCTL4 Register Field Descriptions .....           | 1247 |
| 14-38. PORTCTL5 Register Field Descriptions .....           | 1248 |
| 14-39. CPSW_CPDMA REGISTERS .....                           | 1249 |
| 14-40. TX_IDVER Register Field Descriptions .....           | 1251 |
| 14-41. TX_CONTROL Register Field Descriptions .....         | 1252 |
| 14-42. TX_TEARDOWN Register Field Descriptions .....        | 1253 |
| 14-43. RX_IDVER Register Field Descriptions .....           | 1254 |
| 14-44. RX_CONTROL Register Field Descriptions .....         | 1255 |
| 14-45. RX_TEARDOWN Register Field Descriptions .....        | 1256 |
| 14-46. CPDMA_SOFT_RESET Register Field Descriptions .....   | 1257 |
| 14-47. DMACONTROL Register Field Descriptions .....         | 1258 |
| 14-48. DMASTATUS Register Field Descriptions .....          | 1260 |
| 14-49. RX_BUFFER_OFFSET Register Field Descriptions .....   | 1262 |
| 14-50. EMCONTROL Register Field Descriptions .....          | 1263 |
| 14-51. TX_PRI0_RATE Register Field Descriptions .....       | 1264 |
| 14-52. TX_PRI1_RATE Register Field Descriptions .....       | 1265 |
| 14-53. TX_PRI2_RATE Register Field Descriptions .....       | 1266 |
| 14-54. TX_PRI3_RATE Register Field Descriptions .....       | 1267 |
| 14-55. TX_PRI4_RATE Register Field Descriptions .....       | 1268 |
| 14-56. TX_PRI5_RATE Register Field Descriptions .....       | 1269 |
| 14-57. TX_PRI6_RATE Register Field Descriptions .....       | 1270 |
| 14-58. TX_PRI7_RATE Register Field Descriptions .....       | 1271 |
| 14-59. TX_INTSTAT_RAW Register Field Descriptions .....     | 1272 |
| 14-60. TX_INTSTAT_MASKED Register Field Descriptions .....  | 1273 |
| 14-61. TX_INTMASK_SET Register Field Descriptions .....     | 1274 |
| 14-62. TX_INTMASK_CLEAR Register Field Descriptions .....   | 1275 |
| 14-63. CPDMA_IN_VECTOR Register Field Descriptions .....    | 1276 |
| 14-64. CPDMA_EOI_VECTOR Register Field Descriptions .....   | 1277 |
| 14-65. RX_INTSTAT_RAW Register Field Descriptions .....     | 1278 |
| 14-66. RX_INTSTAT_MASKED Register Field Descriptions .....  | 1279 |
| 14-67. RX_INTMASK_SET Register Field Descriptions .....     | 1280 |
| 14-68. RX_INTMASK_CLEAR Register Field Descriptions .....   | 1281 |
| 14-69. DMA_INTSTAT_RAW Register Field Descriptions .....    | 1282 |
| 14-70. DMA_INTSTAT_MASKED Register Field Descriptions ..... | 1283 |
| 14-71. DMA_INTMASK_SET Register Field Descriptions .....    | 1284 |
| 14-72. DMA_INTMASK_CLEAR Register Field Descriptions .....  | 1285 |
| 14-73. RX0_PENDTHRESH Register Field Descriptions .....     | 1286 |
| 14-74. RX1_PENDTHRESH Register Field Descriptions .....     | 1287 |
| 14-75. RX2_PENDTHRESH Register Field Descriptions .....     | 1288 |
| 14-76. RX3_PENDTHRESH Register Field Descriptions .....     | 1289 |

|  |      |
|--|------|
| 14-77. RX4_PENDTHRESH Register Field Descriptions .....      | 1290 |
| 14-78. RX5_PENDTHRESH Register Field Descriptions .....      | 1291 |
| 14-79. RX6_PENDTHRESH Register Field Descriptions .....      | 1292 |
| 14-80. RX7_PENDTHRESH Register Field Descriptions .....      | 1293 |
| 14-81. RX0_FREEBUFFER Register Field Descriptions .....      | 1294 |
| 14-82. RX1_FREEBUFFER Register Field Descriptions .....      | 1295 |
| 14-83. RX2_FREEBUFFER Register Field Descriptions .....      | 1296 |
| 14-84. RX3_FREEBUFFER Register Field Descriptions .....      | 1297 |
| 14-85. RX4_FREEBUFFER Register Field Descriptions .....      | 1298 |
| 14-86. RX5_FREEBUFFER Register Field Descriptions .....      | 1299 |
| 14-87. RX6_FREEBUFFER Register Field Descriptions .....      | 1300 |
| 14-88. RX7_FREEBUFFER Register Field Descriptions .....      | 1301 |
| 14-89. CPSW_CPTS REGISTERS .....                             | 1301 |
| 14-90. CPTS_IDVER Register Field Descriptions .....          | 1303 |
| 14-91. CPTS_CONTROL Register Field Descriptions .....        | 1304 |
| 14-92. CPTS_TS_PUSH Register Field Descriptions .....        | 1305 |
| 14-93. CPTS_TS_LOAD_VAL Register Field Descriptions .....    | 1306 |
| 14-94. CPTS_TS_LOAD_EN Register Field Descriptions .....     | 1307 |
| 14-95. CPTS_INTSTAT_RAW Register Field Descriptions .....    | 1308 |
| 14-96. CPTS_INTSTAT_MASKED Register Field Descriptions ..... | 1309 |
| 14-97. CPTS_INT_ENABLE Register Field Descriptions .....     | 1310 |
| 14-98. CPTS_EVENT_POP Register Field Descriptions .....      | 1311 |
| 14-99. CPTS_EVENT_LOW Register Field Descriptions .....      | 1312 |
| 14-100. CPTS_EVENT_HIGH Register Field Descriptions .....    | 1313 |
| 14-101. CPSW_STATS REGISTERS .....                           | 1314 |
| 14-102. CPDMA_STATERAM REGISTERS .....                       | 1314 |
| 14-103. TX0_HDP Register Field Descriptions .....            | 1317 |
| 14-104. TX1_HDP Register Field Descriptions .....            | 1318 |
| 14-105. TX2_HDP Register Field Descriptions .....            | 1319 |
| 14-106. TX3_HDP Register Field Descriptions .....            | 1320 |
| 14-107. TX4_HDP Register Field Descriptions .....            | 1321 |
| 14-108. TX5_HDP Register Field Descriptions .....            | 1322 |
| 14-109. TX6_HDP Register Field Descriptions .....            | 1323 |
| 14-110. TX7_HDP Register Field Descriptions .....            | 1324 |
| 14-111. RX0_HDP Register Field Descriptions .....            | 1325 |
| 14-112. RX1_HDP Register Field Descriptions .....            | 1326 |
| 14-113. RX2_HDP Register Field Descriptions .....            | 1327 |
| 14-114. RX3_HDP Register Field Descriptions .....            | 1328 |
| 14-115. RX4_HDP Register Field Descriptions .....            | 1329 |
| 14-116. RX5_HDP Register Field Descriptions .....            | 1330 |
| 14-117. RX6_HDP Register Field Descriptions .....            | 1331 |
| 14-118. RX7_HDP Register Field Descriptions .....            | 1332 |
| 14-119. TX0_CP Register Field Descriptions .....             | 1333 |
| 14-120. TX1_CP Register Field Descriptions .....             | 1334 |
| 14-121. TX2_CP Register Field Descriptions .....             | 1335 |
| 14-122. TX3_CP Register Field Descriptions .....             | 1336 |
| 14-123. TX4_CP Register Field Descriptions .....             | 1337 |
| 14-124. TX5_CP Register Field Descriptions .....             | 1338 |
| 14-125. TX6_CP Register Field Descriptions .....             | 1339 |

|         |   |      |
|---------|---|------|
| 14-126. | TX7_CP Register Field Descriptions .....              | 1340 |
| 14-127. | RX0_CP Register Field Descriptions .....              | 1341 |
| 14-128. | RX1_CP Register Field Descriptions .....              | 1342 |
| 14-129. | RX2_CP Register Field Descriptions .....              | 1343 |
| 14-130. | RX3_CP Register Field Descriptions .....              | 1344 |
| 14-131. | RX4_CP Register Field Descriptions .....              | 1345 |
| 14-132. | RX5_CP Register Field Descriptions .....              | 1346 |
| 14-133. | RX6_CP Register Field Descriptions .....              | 1347 |
| 14-134. | RX7_CP Register Field Descriptions .....              | 1348 |
| 14-135. | CPSW_PORT REGISTERS .....                             | 1348 |
| 14-136. | P0_CONTROL Register Field Descriptions .....          | 1350 |
| 14-137. | P0_MAX_BLKS Register Field Descriptions .....         | 1351 |
| 14-138. | P0_BLK_CNT Register Field Descriptions .....          | 1352 |
| 14-139. | P0_TX_IN_CTL Register Field Descriptions .....        | 1353 |
| 14-140. | P0_PORT_VLAN Register Field Descriptions.....         | 1354 |
| 14-141. | P0_TX_PRI_MAP Register Field Descriptions.....        | 1355 |
| 14-142. | P0_CPDMA_TX_PRI_MAP Register Field Descriptions.....  | 1356 |
| 14-143. | P0_CPDMA_RX_CH_MAP Register Field Descriptions .....  | 1357 |
| 14-144. | P0_RX_DSCP_PRI_MAP0 Register Field Descriptions ..... | 1358 |
| 14-145. | P0_RX_DSCP_PRI_MAP1 Register Field Descriptions ..... | 1359 |
| 14-146. | P0_RX_DSCP_PRI_MAP2 Register Field Descriptions ..... | 1360 |
| 14-147. | P0_RX_DSCP_PRI_MAP3 Register Field Descriptions ..... | 1361 |
| 14-148. | P0_RX_DSCP_PRI_MAP4 Register Field Descriptions ..... | 1362 |
| 14-149. | P0_RX_DSCP_PRI_MAP5 Register Field Descriptions ..... | 1363 |
| 14-150. | P0_RX_DSCP_PRI_MAP6 Register Field Descriptions ..... | 1364 |
| 14-151. | P0_RX_DSCP_PRI_MAP7 Register Field Descriptions ..... | 1365 |
| 14-152. | P1_CONTROL Register Field Descriptions .....          | 1366 |
| 14-153. | P1_MAX_BLKS Register Field Descriptions .....         | 1368 |
| 14-154. | P1_BLK_CNT Register Field Descriptions .....          | 1369 |
| 14-155. | P1_TX_IN_CTL Register Field Descriptions .....        | 1370 |
| 14-156. | P1_PORT_VLAN Register Field Descriptions.....         | 1371 |
| 14-157. | P1_TX_PRI_MAP Register Field Descriptions.....        | 1372 |
| 14-158. | P1_TS_SEQ_MTYPE Register Field Descriptions .....     | 1373 |
| 14-159. | P1_SA_LO Register Field Descriptions .....            | 1374 |
| 14-160. | P1_SA_HI Register Field Descriptions.....             | 1375 |
| 14-161. | P1_SEND_PERCENT Register Field Descriptions.....      | 1376 |
| 14-162. | P1_RX_DSCP_PRI_MAP0 Register Field Descriptions ..... | 1377 |
| 14-163. | P1_RX_DSCP_PRI_MAP1 Register Field Descriptions ..... | 1378 |
| 14-164. | P1_RX_DSCP_PRI_MAP2 Register Field Descriptions ..... | 1379 |
| 14-165. | P1_RX_DSCP_PRI_MAP3 Register Field Descriptions ..... | 1380 |
| 14-166. | P1_RX_DSCP_PRI_MAP4 Register Field Descriptions ..... | 1381 |
| 14-167. | P1_RX_DSCP_PRI_MAP5 Register Field Descriptions ..... | 1382 |
| 14-168. | P1_RX_DSCP_PRI_MAP6 Register Field Descriptions ..... | 1383 |
| 14-169. | P1_RX_DSCP_PRI_MAP7 Register Field Descriptions ..... | 1384 |
| 14-170. | P2_CONTROL Register Field Descriptions .....          | 1385 |
| 14-171. | P2_MAX_BLKS Register Field Descriptions .....         | 1387 |
| 14-172. | P2_BLK_CNT Register Field Descriptions .....          | 1388 |
| 14-173. | P2_TX_IN_CTL Register Field Descriptions .....        | 1389 |
| 14-174. | P2_PORT_VLAN Register Field Descriptions.....         | 1390 |

|   |      |
|---|------|
| 14-175. P2_TX_PRI_MAP Register Field Descriptions .....       | 1391 |
| 14-176. P2_TS_SEQ_MTYPE Register Field Descriptions .....     | 1392 |
| 14-177. P2_SA_LO Register Field Descriptions .....            | 1393 |
| 14-178. P2_SA_HI Register Field Descriptions .....            | 1394 |
| 14-179. P2_SEND_PERCENT Register Field Descriptions .....     | 1395 |
| 14-180. P2_RX_DSCP_PRI_MAP0 Register Field Descriptions ..... | 1396 |
| 14-181. P2_RX_DSCP_PRI_MAP1 Register Field Descriptions ..... | 1397 |
| 14-182. P2_RX_DSCP_PRI_MAP2 Register Field Descriptions ..... | 1398 |
| 14-183. P2_RX_DSCP_PRI_MAP3 Register Field Descriptions ..... | 1399 |
| 14-184. P2_RX_DSCP_PRI_MAP4 Register Field Descriptions ..... | 1400 |
| 14-185. P2_RX_DSCP_PRI_MAP5 Register Field Descriptions ..... | 1401 |
| 14-186. P2_RX_DSCP_PRI_MAP6 Register Field Descriptions ..... | 1402 |
| 14-187. P2_RX_DSCP_PRI_MAP7 Register Field Descriptions ..... | 1403 |
| 14-188. CPSW_SL REGISTERS .....                               | 1404 |
| 14-189. IDVER Register Field Descriptions .....               | 1405 |
| 14-190. MACCONTROL Register Field Descriptions .....          | 1406 |
| 14-191. MACSTATUS Register Field Descriptions .....           | 1409 |
| 14-192. SOFT_RESET Register Field Descriptions .....          | 1410 |
| 14-193. RX_MAXLEN Register Field Descriptions .....           | 1411 |
| 14-194. BOFFTEST Register Field Descriptions .....            | 1412 |
| 14-195. RX_PAUSE Register Field Descriptions .....            | 1413 |
| 14-196. TX_PAUSE Register Field Descriptions .....            | 1414 |
| 14-197. EMCONTROL Register Field Descriptions .....           | 1415 |
| 14-198. RX_PRI_MAP Register Field Descriptions .....          | 1416 |
| 14-199. TX_GAP Register Field Descriptions .....              | 1417 |
| 14-200. CPSW_SS REGISTERS .....                               | 1417 |
| 14-201. ID_VER Register Field Descriptions .....              | 1418 |
| 14-202. CONTROL Register Field Descriptions .....             | 1419 |
| 14-203. SOFT_RESET Register Field Descriptions .....          | 1420 |
| 14-204. STAT_PORT_EN Register Field Descriptions .....        | 1421 |
| 14-205. PTYPE Register Field Descriptions .....               | 1422 |
| 14-206. SOFT_IDLE Register Field Descriptions .....           | 1423 |
| 14-207. THRU_RATE Register Field Descriptions .....           | 1424 |
| 14-208. GAP_THRESH Register Field Descriptions .....          | 1425 |
| 14-209. TX_START_WDS Register Field Descriptions .....        | 1426 |
| 14-210. FLOW_CONTROL Register Field Descriptions .....        | 1427 |
| 14-211. VLAN_LTYPE Register Field Descriptions .....          | 1428 |
| 14-212. TS_LTYPE Register Field Descriptions .....            | 1429 |
| 14-213. DLR_LTYPE Register Field Descriptions .....           | 1430 |
| 14-214. CPSW_WR REGISTERS .....                               | 1430 |
| 14-215. IDVER Register Field Descriptions .....               | 1432 |
| 14-216. SOFT_RESET Register Field Descriptions .....          | 1433 |
| 14-217. CONTROL Register Field Descriptions .....             | 1434 |
| 14-218. INT_CONTROL Register Field Descriptions .....         | 1435 |
| 14-219. C0_RX_THRESH_EN Register Field Descriptions .....     | 1436 |
| 14-220. C0_RX_EN Register Field Descriptions .....            | 1437 |
| 14-221. C0_TX_EN Register Field Descriptions .....            | 1438 |
| 14-222. C0_MISC_EN Register Field Descriptions .....          | 1439 |
| 14-223. C1_RX_THRESH_EN Register Field Descriptions .....     | 1440 |

|   |      |
|---|------|
| 14-224. C1_RX_EN Register Field Descriptions .....  | 1441 |
| 14-225. C1_TX_EN Register Field Descriptions.....   | 1442 |
| 14-226. C1_MISC_EN Register Field Descriptions .....  | 1443 |
| 14-227. C2_RX_THRESH_EN Register Field Descriptions .....   | 1444 |
| 14-228. C2_RX_EN Register Field Descriptions .....  | 1445 |
| 14-229. C2_TX_EN Register Field Descriptions.....   | 1446 |
| 14-230. C2_MISC_EN Register Field Descriptions .....  | 1447 |
| 14-231. C0_RX_THRESH_STAT Register Field Descriptions.....  | 1448 |
| 14-232. C0_RX_STAT Register Field Descriptions .....  | 1449 |
| 14-233. C0_TX_STAT Register Field Descriptions.....   | 1450 |
| 14-234. C0_MISC_STAT Register Field Descriptions .....  | 1451 |
| 14-235. C1_RX_THRESH_STAT Register Field Descriptions.....  | 1452 |
| 14-236. C1_RX_STAT Register Field Descriptions .....  | 1453 |
| 14-237. C1_TX_STAT Register Field Descriptions.....   | 1454 |
| 14-238. C1_MISC_STAT Register Field Descriptions .....  | 1455 |
| 14-239. C2_RX_THRESH_STAT Register Field Descriptions.....  | 1456 |
| 14-240. C2_RX_STAT Register Field Descriptions .....  | 1457 |
| 14-241. C2_TX_STAT Register Field Descriptions.....   | 1458 |
| 14-242. C2_MISC_STAT Register Field Descriptions .....  | 1459 |
| 14-243. C0_RX_IMAX Register Field Descriptions.....   | 1460 |
| 14-244. C0_TX_IMAX Register Field Descriptions .....  | 1461 |
| 14-245. C1_RX_IMAX Register Field Descriptions.....   | 1462 |
| 14-246. C1_TX_IMAX Register Field Descriptions .....  | 1463 |
| 14-247. C2_RX_IMAX Register Field Descriptions.....   | 1464 |
| 14-248. C2_TX_IMAX Register Field Descriptions .....  | 1465 |
| 14-249. RGMII_CTL Register Field Descriptions .....   | 1466 |
| 14-250. Management Data Input/Output (MDIO) Registers .....   | 1466 |
| 14-251. MDIO Version Register (MDIOVER) Field Descriptions .....  | 1467 |
| 14-252. MDIO Control Register (MDIOCONTROL) Field Descriptions .....  | 1468 |
| 14-253. PHY Acknowledge Status Register (MDIOALIVE) Field Descriptions .....                                      | 1469 |
| 14-254. PHY Link Status Register (MDIOLINK) Field Descriptions .....  | 1469 |
| 14-255. MDIO Link Status Change Interrupt Register (MDIOLINKINTRAW) Field Descriptions .....                      | 1470 |
| 14-256. MDIO Link Status Change Interrupt Register (Masked Value) (MDIOLINKINTMASKED) Field Descriptions .....    | 1470 |
| 14-257. MDIO User Command Complete Interrupt Register (Raw Value) (MDIOUSERINTRAW) Field Descriptions .....       | 1471 |
| 14-258. MDIO User Command Complete Interrupt Register (Masked Value) (MDIOUSERINTMASKED) Field Descriptions ..... | 1471 |
| 14-259. MDIO User Command Complete Interrupt Mask Set Register (MDIOUSERINTMASKSET) Field Descriptions .....      | 1472 |
| 14-260. MDIO User Command Complete Interrupt Mask Clear Register (MDIOUSERINTMASKCLR) Field Descriptions .....    | 1472 |
| 14-261. MDIO User Access Register 0 (MDIOUSERACCESS0) Field Descriptions .....                                    | 1473 |
| 14-262. MDIO User PHY Select Register 0 (MDIOUSERPHYSEL0) Field Descriptions.....                                 | 1474 |
| 14-263. MDIO User Access Register 1 (MDIOUSERACCESS1) Field Descriptions .....                                    | 1475 |
| 14-264. MDIO User PHY Select Register 1 (MDIOUSERPHYSEL1) Field Descriptions.....                                 | 1476 |
| 15-1. Unsupported Features .....  | 1479 |
| 15-2. PWMSS Connectivity Attributes .....   | 1480 |
| 15-3. PWMSS Clock Signals.....  | 1481 |
| 15-4. PWMSS Pin List .....  | 1481 |

|  |      |
|--|------|
| 15-5. PWMSS REGISTERS .....  | 1481 |
| 15-6. IDVER Register Field Descriptions.....                             | 1482 |
| 15-7. SYS CONFIG Register Field Descriptions .....                       | 1483 |
| 15-8. CLKCONFIG Register Field Descriptions .....                        | 1484 |
| 15-9. CLKSTATUS Register Field Descriptions .....                        | 1485 |
| 15-10. Submodule Configuration Parameters.....                           | 1490 |
| 15-11. Time-Base Submodule Registers .....                               | 1495 |
| 15-12. Key Time-Base Signals .....                                       | 1496 |
| 15-13. Counter-Compare Submodule Registers .....                         | 1504 |
| 15-14. Counter-Compare Submodule Key Signals.....                        | 1504 |
| 15-15. Action-Qualifier Submodule Registers .....                        | 1508 |
| 15-16. Action-Qualifier Submodule Possible Input Events .....            | 1509 |
| 15-17. Action-Qualifier Event Priority for Up-Down-Count Mode .....      | 1511 |
| 15-18. Action-Qualifier Event Priority for Up-Count Mode.....            | 1511 |
| 15-19. Action-Qualifier Event Priority for Down-Count Mode .....         | 1511 |
| 15-20. Behavior if CMPA/CMPB is Greater than the Period .....            | 1512 |
| 15-21. EPWMx Initialization for .....                                    | 1515 |
| 15-22. EPWMx Run Time Changes for .....                                  | 1515 |
| 15-23. EPWMx Initialization for .....                                    | 1517 |
| 15-24. EPWMx Run Time Changes for .....                                  | 1517 |
| 15-25. EPWMx Initialization for .....                                    | 1519 |
| 15-26. EPWMx Run Time Changes for .....                                  | 1519 |
| 15-27. EPWMx Initialization for .....                                    | 1521 |
| 15-28. EPWMx Run Time Changes for .....                                  | 1521 |
| 15-29. EPWMx Initialization for .....                                    | 1523 |
| 15-30. EPWMx Run Time Changes for .....                                  | 1523 |
| 15-31. EPWMx Initialization for .....                                    | 1525 |
| 15-32. EPWMx Run Time Changes for .....                                  | 1525 |
| 15-33. Dead-Band Generator Submodule Registers.....                      | 1526 |
| 15-34. Classical Dead-Band Operating Modes .....                         | 1528 |
| 15-35. PWM-Chopper Submodule Registers .....                             | 1530 |
| 15-36. Trip-Zone Submodule Registers.....                                | 1535 |
| 15-37. Possible Actions On a Trip Event .....                            | 1536 |
| 15-38. Event-Trigger Submodule Registers .....                           | 1538 |
| 15-39. Resolution for PWM and HRPWM .....                                | 1543 |
| 15-40. HRPWM Submodule Registers .....                                   | 1544 |
| 15-41. Relationship Between MEP Steps, PWM Frequency and Resolution..... | 1545 |
| 15-42. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right).....       | 1546 |
| 15-43. EPWM1 Initialization for .....                                    | 1553 |
| 15-44. EPWM2 Initialization for .....                                    | 1553 |
| 15-45. EPWM3 Initialization for .....                                    | 1553 |
| 15-46. EPWM1 Initialization for .....                                    | 1556 |
| 15-47. EPWM2 Initialization for .....                                    | 1556 |
| 15-48. EPWM1 Initialization for .....                                    | 1559 |
| 15-49. EPWM2 Initialization for .....                                    | 1559 |
| 15-50. EPWM1 Initialization for .....                                    | 1562 |
| 15-51. EPWM2 Initialization for .....                                    | 1562 |
| 15-52. EPWM3 Initialization for .....                                    | 1563 |
| 15-53. EPWM1 Initialization for .....                                    | 1568 |

|   |      |
|---|------|
| 15-54. EPWM2 Initialization for .....   | 1568 |
| 15-55. EPWM3 Initialization for .....   | 1569 |
| 15-56. EPWM1 Initialization for .....   | 1572 |
| 15-57. EPWM2 Initialization for .....   | 1572 |
| 15-58. ePWM Module Control and Status Registers Grouped by Submodule .....                    | 1573 |
| 15-59. Time-Base Submodule Registers .....  | 1574 |
| 15-60. Time-Base Control Register (TBCTL) Field Descriptions .....                            | 1574 |
| 15-61. Time-Base Status Register (TBSTS) Field Descriptions .....                             | 1576 |
| 15-62. Time-Base Phase Register (TBPHS) Field Descriptions .....                              | 1577 |
| 15-63. Time-Base Counter Register (TBCNT) Field Descriptions .....                            | 1577 |
| 15-64. Time-Base Period Register (TBPRD) Field Descriptions .....                             | 1578 |
| 15-65. Counter-Compare Submodule Registers .....  | 1578 |
| 15-66. Counter-Compare Control Register (CMPCTL) Field Descriptions .....                     | 1579 |
| 15-67. Counter-Compare A Register (CMPA) Field Descriptions .....                             | 1580 |
| 15-68. Counter-Compare B Register (CMPB) Field Descriptions .....                             | 1581 |
| 15-69. Action-Qualifier Submodule Registers .....   | 1581 |
| 15-70. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions .....           | 1582 |
| 15-71. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions .....           | 1583 |
| 15-72. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions .....             | 1584 |
| 15-73. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions ..... | 1585 |
| 15-74. Dead-Band Generator Submodule Registers .....  | 1585 |
| 15-75. Dead-Band Generator Control Register (DBCTL) Field Descriptions .....                  | 1586 |
| 15-76. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions .....        | 1587 |
| 15-77. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions .....       | 1587 |
| 15-78. Trip-Zone Submodule Registers .....  | 1588 |
| 15-79. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions .....                   | 1588 |
| 15-80. Trip-Zone Control Register (TZCTL) Field Descriptions .....                            | 1589 |
| 15-81. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions .....                  | 1589 |
| 15-82. Trip-Zone Flag Register (TZFLG) Field Descriptions .....                               | 1590 |
| 15-83. Trip-Zone Clear Register (TZCLR) Field Descriptions .....                              | 1591 |
| 15-84. Trip-Zone Force Register (TZFRC) Field Descriptions .....                              | 1591 |
| 15-85. Event-Trigger Submodule Registers .....  | 1592 |
| 15-86. Event-Trigger Selection Register (ETSEL) Field Descriptions .....                      | 1592 |
| 15-87. Event-Trigger Prescale Register (ETPS) Field Descriptions .....                        | 1593 |
| 15-88. Event-Trigger Flag Register (ETFLG) Field Descriptions .....                           | 1594 |
| 15-89. Event-Trigger Clear Register (ETCLR) Field Descriptions .....                          | 1594 |
| 15-90. Event-Trigger Force Register (ETFRC) Field Descriptions .....                          | 1595 |
| 15-91. PWM-Chopper Control Register (PCCTL) Bit Descriptions .....                            | 1596 |
| 15-92. High-Resolution PWM Submodule Registers .....  | 1596 |
| 15-93. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions .....            | 1597 |
| 15-94. Counter-Compare A High-Resolution Register (CMPAHHR) Field Descriptions .....          | 1597 |
| 15-95. HRPWM Control Register (HRCTL) Field Descriptions .....                                | 1598 |
| 15-96. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger .....              | 1612 |
| 15-97. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger .....  | 1614 |
| 15-98. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger .....                 | 1616 |
| 15-99. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers .....    | 1618 |
| 15-100. ECAP Initialization for APWM Mode .....   | 1620 |
| 15-101. ECAP1 Initialization for Multichannel PWM Generation with Synchronization .....       | 1622 |
| 15-102. ECAP2 Initialization for Multichannel PWM Generation with Synchronization .....       | 1622 |

|   |      |
|---|------|
| 15-103. ECAP3 Initialization for Multichannel PWM Generation with Synchronization .....   | 1622 |
| 15-104. ECAP4 Initialization for Multichannel PWM Generation with Synchronization .....   | 1622 |
| 15-105. ECAP1 Initialization for Multichannel PWM Generation with Phase Control .....     | 1625 |
| 15-106. ECAP2 Initialization for Multichannel PWM Generation with Phase Control .....     | 1625 |
| 15-107. ECAP3 Initialization for Multichannel PWM Generation with Phase Control .....     | 1625 |
| 15-108. Control and Status Register Set .....   | 1626 |
| 15-109. Time-Stamp Counter Register (TSCTR) Field Descriptions .....                      | 1626 |
| 15-110. Counter Phase Control Register (CTRPHS) Field Descriptions .....                  | 1627 |
| 15-111. Capture 1 Register (CAP1) Field Descriptions .....                                | 1627 |
| 15-112. Capture 2 Register (CAP2) Field Descriptions .....                                | 1628 |
| 15-113. Capture 3 Register (CAP3) Field Descriptions .....                                | 1628 |
| 15-114. Capture 4 Register (CAP4) Field Descriptions .....                                | 1629 |
| 15-115. ECAP Control Register 1 (ECCTL1) Field Descriptions .....                         | 1629 |
| 15-116. ECAP Control Register 2 (ECCTL2) Field Descriptions .....                         | 1631 |
| 15-117. ECAP Interrupt Enable Register (ECEINT) Field Descriptions .....                  | 1633 |
| 15-118. ECAP Interrupt Flag Register (ECFLG) Field Descriptions .....                     | 1634 |
| 15-119. ECAP Interrupt Clear Register (ECCLR) Field Descriptions .....                    | 1635 |
| 15-120. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions .....                  | 1636 |
| 15-121. Revision ID Register (REVID) Field Descriptions .....                             | 1637 |
| 15-122. Quadrature Decoder Truth Table .....  | 1644 |
| 15-123. eQEP Registers .....  | 1659 |
| 15-124. eQEP Position Counter Register (QPOSCNT) Field Descriptions .....                 | 1660 |
| 15-125. eQEP Position Counter Initialization Register (QPOSINIT) Field Descriptions ..... | 1660 |
| 15-126. eQEP Maximum Position Count Register (QPOSMAX) Field Descriptions .....           | 1660 |
| 15-127. eQEP Position-Compare Register (QPOSCMP) Field Descriptions .....                 | 1661 |
| 15-128. eQEP Index Position Latch Register (QPOSILAT) Field Descriptions .....            | 1661 |
| 15-129. eQEP Strobe Position Latch Register (QPOSSLAT) Field Descriptions .....           | 1661 |
| 15-130. eQEP Position Counter Latch Register (QPOSLAT) Field Descriptions .....           | 1662 |
| 15-131. eQEP Unit Timer Register (QUTMR) Field Descriptions .....                         | 1662 |
| 15-132. eQEP Unit Period Register (QUPRD) Field Descriptions .....                        | 1662 |
| 15-133. eQEP Watchdog Timer Register (QWDTMR) Field Descriptions .....                    | 1663 |
| 15-134. eQEP Watchdog Period Register (QWDPRD) Field Description .....                    | 1663 |
| 15-135. eQEP Decoder Control Register (QDECCTL) Field Descriptions .....                  | 1664 |
| 15-136. eQEP Control Register (QEPCTL) Field Descriptions .....                           | 1665 |
| 15-137. eQEP Capture Control Register (QCAPCTL) Field Descriptions .....                  | 1667 |
| 15-138. eQEP Position-Compare Control Register (QPOSCTL) Field Descriptions .....         | 1668 |
| 15-139. eQEP Interrupt Enable Register (QEINT) Field Descriptions .....                   | 1669 |
| 15-140. eQEP Interrupt Flag Register (QFLG) Field Descriptions .....                      | 1670 |
| 15-141. eQEP Interrupt Clear Register (QCLR) Field Descriptions .....                     | 1671 |
| 15-142. eQEP Interrupt Force Register (QFRC) Field Descriptions .....                     | 1673 |
| 15-143. eQEP Status Register (QEPSTS) Field Descriptions .....                            | 1674 |
| 15-144. eQEP Capture Time Register (QCTMR) Field Descriptions .....                       | 1675 |
| 15-145. eQEP Capture Period Register (QCPRD) Field Descriptions .....                     | 1675 |
| 15-146. eQEP Capture Timer Latch Register (QCTMRLAT) Field Descriptions .....             | 1675 |
| 15-147. eQEP Capture Period Latch Register (QCPRDLAT) Field Descriptions .....            | 1676 |
| 15-148. eQEP Revision ID Register (REVID) Field Descriptions .....                        | 1676 |
| 16-1. USB Connectivity Attributes .....   | 1681 |
| 16-2. USB Clock Signals .....   | 1682 |
| 16-3. USB Pin List .....  | 1682 |

|        |  |      |
|--------|--|------|
| 16-4.  | PERI_TXCSR Register Bit Configuration for Bulk IN Transactions .....         | 1698 |
| 16-5.  | PERI_RXCSR Register Bit Configuration for Bulk OUT Transactions .....        | 1700 |
| 16-6.  | PERI_TXCSR Register Bit Configuration for Isochronous IN Transactions .....  | 1702 |
| 16-7.  | PERI_RXCSR Register Bit Configuration for Isochronous OUT Transactions ..... | 1703 |
| 16-8.  | Isochronous OUT Error Handling: Peripheral Mode .....                        | 1704 |
| 16-9.  | Packet Descriptor Word 0 (PD0) Bit Field Descriptions .....                  | 1723 |
| 16-10. | Packet Descriptor Word 1 (PD1) Bit Field Descriptions .....                  | 1724 |
| 16-11. | Packet Descriptor Word 2 (PD2) Bit Field Descriptions .....                  | 1724 |
| 16-12. | Packet Descriptor Word 3 (PD3) Bit Field Descriptions .....                  | 1724 |
| 16-13. | Packet Descriptor Word 4 (PD4) Bit Field Descriptions .....                  | 1725 |
| 16-14. | Packet Descriptor Word 5 (PD5) Bit Field Descriptions .....                  | 1725 |
| 16-15. | Packet Descriptor Word 6 (PD6) Bit Field Descriptions .....                  | 1725 |
| 16-16. | Packet Descriptor Word 7 (PD7) Bit Field Descriptions .....                  | 1725 |
| 16-17. | Buffer Descriptor Word 0 (BD0) Bit Field Descriptions .....                  | 1726 |
| 16-18. | Buffer Descriptor Word 1 (BD1) Bit Field Descriptions .....                  | 1726 |
| 16-19. | Buffer Descriptor Word 2 (BD2) Bit Field Descriptions .....                  | 1726 |
| 16-20. | Buffer Descriptor Word 3 (BD3) Bit Field Descriptions .....                  | 1726 |
| 16-21. | Buffer Descriptor Word 4 (BD4) Bit Field Descriptions .....                  | 1727 |
| 16-22. | Buffer Descriptor Word 5 (BD5) Bit Field Descriptions .....                  | 1727 |
| 16-23. | Buffer Descriptor Word 6 (BD6) Bit Field Descriptions .....                  | 1727 |
| 16-24. | Buffer Descriptor Word 7 (BD7) Bit Field Descriptions .....                  | 1727 |
| 16-25. | Teardown Descriptor Word 0 Bit Field Descriptions .....                      | 1728 |
| 16-26. | Teardown Descriptor Words 1 to 7 Bit Field Descriptions .....                | 1728 |
| 16-27. | Queue-Endpoint Assignments .....   | 1729 |
| 16-28. | 53 Bytes Test Packet Content .....   | 1745 |
| 16-29. | USBSS REGISTERS .....  | 1747 |
| 16-30. | REVREG Register Field Descriptions .....                                     | 1749 |
| 16-31. | SYSCONFIG Register Field Descriptions .....                                  | 1750 |
| 16-32. | IRQSTATRAW Register Field Descriptions .....                                 | 1751 |
| 16-33. | IRQSTAT Register Field Descriptions .....                                    | 1752 |
| 16-34. | IRQENABLER Register Field Descriptions .....                                 | 1753 |
| 16-35. | IRQCLEARR Register Field Descriptions .....                                  | 1754 |
| 16-36. | IRQDMATHOLDTX00 Register Field Descriptions .....                            | 1755 |
| 16-37. | IRQDMATHOLDTX01 Register Field Descriptions .....                            | 1756 |
| 16-38. | IRQDMATHOLDTX02 Register Field Descriptions .....                            | 1757 |
| 16-39. | IRQDMATHOLDTX03 Register Field Descriptions .....                            | 1758 |
| 16-40. | IRQDMATHOLDRX00 Register Field Descriptions .....                            | 1759 |
| 16-41. | IRQDMATHOLDRX01 Register Field Descriptions .....                            | 1760 |
| 16-42. | IRQDMATHOLDRX02 Register Field Descriptions .....                            | 1761 |
| 16-43. | IRQDMATHOLDRX03 Register Field Descriptions .....                            | 1762 |
| 16-44. | IRQDMATHOLDTX10 Register Field Descriptions .....                            | 1763 |
| 16-45. | IRQDMATHOLDTX11 Register Field Descriptions .....                            | 1764 |
| 16-46. | IRQDMATHOLDTX12 Register Field Descriptions .....                            | 1765 |
| 16-47. | IRQDMATHOLDTX13 Register Field Descriptions .....                            | 1766 |
| 16-48. | IRQDMATHOLDRX10 Register Field Descriptions .....                            | 1767 |
| 16-49. | IRQDMATHOLDRX11 Register Field Descriptions .....                            | 1768 |
| 16-50. | IRQDMATHOLDRX12 Register Field Descriptions .....                            | 1769 |
| 16-51. | IRQDMATHOLDRX13 Register Field Descriptions .....                            | 1770 |
| 16-52. | IRQDMAENABLE0 Register Field Descriptions .....                              | 1771 |

|  |      |
|--|------|
| 16-53. IRQDMAENABLE1 Register Field Descriptions .....     | 1772 |
| 16-54. IRQFRAMETHOLDTX00 Register Field Descriptions ..... | 1773 |
| 16-55. IRQFRAMETHOLDTX01 Register Field Descriptions ..... | 1774 |
| 16-56. IRQFRAMETHOLDTX02 Register Field Descriptions ..... | 1775 |
| 16-57. IRQFRAMETHOLDTX03 Register Field Descriptions ..... | 1776 |
| 16-58. IRQFRAMETHOLDRX00 Register Field Descriptions.....  | 1777 |
| 16-59. IRQFRAMETHOLDRX01 Register Field Descriptions.....  | 1778 |
| 16-60. IRQFRAMETHOLDRX02 Register Field Descriptions.....  | 1779 |
| 16-61. IRQFRAMETHOLDRX03 Register Field Descriptions.....  | 1780 |
| 16-62. IRQFRAMETHOLDTX10 Register Field Descriptions ..... | 1781 |
| 16-63. IRQFRAMETHOLDTX11 Register Field Descriptions ..... | 1782 |
| 16-64. IRQFRAMETHOLDTX12 Register Field Descriptions ..... | 1783 |
| 16-65. IRQFRAMETHOLDTX13 Register Field Descriptions ..... | 1784 |
| 16-66. IRQFRAMETHOLDRX10 Register Field Descriptions.....  | 1785 |
| 16-67. IRQFRAMETHOLDRX11 Register Field Descriptions.....  | 1786 |
| 16-68. IRQFRAMETHOLDRX12 Register Field Descriptions.....  | 1787 |
| 16-69. IRQFRAMETHOLDRX13 Register Field Descriptions.....  | 1788 |
| 16-70. IRQFRAMEENABLE0 Register Field Descriptions .....   | 1789 |
| 16-71. IRQFRAMEENABLE1 Register Field Descriptions .....   | 1790 |
| 16-72. USB0_CTRL REGISTERS .....                           | 1790 |
| 16-73. USB0REV Register Field Descriptions .....           | 1792 |
| 16-74. USB0CTRL Register Field Descriptions.....           | 1793 |
| 16-75. USB0STAT Register Field Descriptions .....          | 1795 |
| 16-76. USB0IRQMSTAT Register Field Descriptions.....       | 1796 |
| 16-77. USB0IRQSTATRAW0 Register Field Descriptions .....   | 1797 |
| 16-78. USB0IRQSTATRAW1 Register Field Descriptions .....   | 1799 |
| 16-79. USB0IRQSTAT0 Register Field Descriptions.....       | 1801 |
| 16-80. USB0IRQSTAT1 Register Field Descriptions.....       | 1803 |
| 16-81. USB0IRQENABLESET0 Register Field Descriptions.....  | 1805 |
| 16-82. USB0IRQENABLESET1 Register Field Descriptions.....  | 1807 |
| 16-83. USB0IRQENABLECLR0 Register Field Descriptions ..... | 1809 |
| 16-84. USB0IRQENABLECLR1 Register Field Descriptions ..... | 1811 |
| 16-85. USB0TXMODE Register Field Descriptions .....        | 1813 |
| 16-86. USB0RXMODE Register Field Descriptions .....        | 1815 |
| 16-87. USB0GENRNDISEP1 Register Field Descriptions .....   | 1819 |
| 16-88. USB0GENRNDISEP2 Register Field Descriptions .....   | 1820 |
| 16-89. USB0GENRNDISEP3 Register Field Descriptions .....   | 1821 |
| 16-90. USB0GENRNDISEP4 Register Field Descriptions .....   | 1822 |
| 16-91. USB0GENRNDISEP5 Register Field Descriptions .....   | 1823 |
| 16-92. USB0GENRNDISEP6 Register Field Descriptions .....   | 1824 |
| 16-93. USB0GENRNDISEP7 Register Field Descriptions .....   | 1825 |
| 16-94. USB0GENRNDISEP8 Register Field Descriptions .....   | 1826 |
| 16-95. USB0GENRNDISEP9 Register Field Descriptions .....   | 1827 |
| 16-96. USB0GENRNDISEP10 Register Field Descriptions .....  | 1828 |
| 16-97. USB0GENRNDISEP11 Register Field Descriptions .....  | 1829 |
| 16-98. USB0GENRNDISEP12 Register Field Descriptions .....  | 1830 |
| 16-99. USB0GENRNDISEP13 Register Field Descriptions .....  | 1831 |
| 16-100. USB0GENRNDISEP14 Register Field Descriptions.....  | 1832 |
| 16-101. USB0GENRNDISEP15 Register Field Descriptions.....  | 1833 |

|         |  |      |
|---------|--|------|
| 16-102. | USB0AUTOREQ Register Field Descriptions .....        | 1834 |
| 16-103. | USB0SRPFIETIME Register Field Descriptions .....     | 1836 |
| 16-104. | USB0_TDOWN Register Field Descriptions.....          | 1837 |
| 16-105. | USB0UTMI Register Field Descriptions .....           | 1838 |
| 16-106. | USB0MGCUTMILB Register Field Descriptions .....      | 1839 |
| 16-107. | USB0MODE Register Field Descriptions .....           | 1840 |
| 16-108. | USB1_CTRL REGISTERS.....                             | 1840 |
| 16-109. | USB1REV Register Field Descriptions .....            | 1842 |
| 16-110. | USB1CTRL Register Field Descriptions .....           | 1843 |
| 16-111. | USB1STAT Register Field Descriptions .....           | 1845 |
| 16-112. | USB1IRQMSTAT Register Field Descriptions .....       | 1846 |
| 16-113. | USB1IRQSTATRAW0 Register Field Descriptions .....    | 1847 |
| 16-114. | USB1IRQSTATRAW1 Register Field Descriptions .....    | 1849 |
| 16-115. | USB1IRQSTAT0 Register Field Descriptions .....       | 1851 |
| 16-116. | USB1IRQSTAT1 Register Field Descriptions .....       | 1853 |
| 16-117. | USB1IRQENABLESET0 Register Field Descriptions .....  | 1855 |
| 16-118. | USB1IRQENABLESET1 Register Field Descriptions .....  | 1857 |
| 16-119. | USB1IRQENABLECLR0 Register Field Descriptions .....  | 1859 |
| 16-120. | USB1IRQENABLECLR1 Register Field Descriptions .....  | 1861 |
| 16-121. | USB1TXMODE Register Field Descriptions .....         | 1863 |
| 16-122. | USB1RXMODE Register Field Descriptions.....          | 1865 |
| 16-123. | USB1GENRNDISEP1 Register Field Descriptions .....    | 1867 |
| 16-124. | USB1GENRNDISEP2 Register Field Descriptions .....    | 1868 |
| 16-125. | USB1GENRNDISEP3 Register Field Descriptions .....    | 1869 |
| 16-126. | USB1GENRNDISEP4 Register Field Descriptions .....    | 1870 |
| 16-127. | USB1GENRNDISEP5 Register Field Descriptions .....    | 1871 |
| 16-128. | USB1GENRNDISEP6 Register Field Descriptions .....    | 1872 |
| 16-129. | USB1GENRNDISEP7 Register Field Descriptions .....    | 1873 |
| 16-130. | USB1GENRNDISEP8 Register Field Descriptions .....    | 1874 |
| 16-131. | USB1GENRNDISEP9 Register Field Descriptions .....    | 1875 |
| 16-132. | USB1GENRNDISEP10 Register Field Descriptions.....    | 1876 |
| 16-133. | USB1GENRNDISEP11 Register Field Descriptions.....    | 1877 |
| 16-134. | USB1GENRNDISEP12 Register Field Descriptions.....    | 1878 |
| 16-135. | USB1GENRNDISEP13 Register Field Descriptions.....    | 1879 |
| 16-136. | USB1GENRNDISEP14 Register Field Descriptions.....    | 1880 |
| 16-137. | USB1GENRNDISEP15 Register Field Descriptions.....    | 1881 |
| 16-138. | USB1AUTOREQ Register Field Descriptions .....        | 1882 |
| 16-139. | USB1SRPFIETIME Register Field Descriptions .....     | 1884 |
| 16-140. | USB1TDOWN Register Field Descriptions .....          | 1885 |
| 16-141. | USB1UTMI Register Field Descriptions.....            | 1886 |
| 16-142. | USB1UTMILB Register Field Descriptions .....         | 1887 |
| 16-143. | USB1MODE Register Field Descriptions .....           | 1888 |
| 16-144. | USB2PHY REGISTERS.....                               | 1888 |
| 16-145. | Termination_control Register Field Descriptions..... | 1890 |
| 16-146. | RX_CALIB Register Field Descriptions .....           | 1891 |
| 16-147. | DLLHS_2 Register Field Descriptions.....             | 1893 |
| 16-148. | RX_TEST_2 Register Field Descriptions .....          | 1894 |
| 16-149. | CHRG_DET Register Field Descriptions.....            | 1895 |
| 16-150. | PWR_CNTL Register Field Descriptions.....            | 1897 |

|   |      |
|---|------|
| 16-151. UTMI_INTERFACE_CNTL_1 Register Field Descriptions ..... | 1898 |
| 16-152. UTMI_INTERFACE_CNTL_2 Register Field Descriptions ..... | 1899 |
| 16-153. BIST Register Field Descriptions .....                  | 1901 |
| 16-154. BIST_CRC Register Field Descriptions .....              | 1902 |
| 16-155. CDR_BIST2 Register Field Descriptions.....              | 1903 |
| 16-156. GPIO Register Field Descriptions.....                   | 1904 |
| 16-157. DLLHS Register Field Descriptions.....                  | 1905 |
| 16-158. USB2PHYCM_CONFIG Register Field Descriptions.....       | 1906 |
| 16-159. AD_INTERFACE_REG1 Register Field Descriptions .....     | 1907 |
| 16-160. AD_INTERFACE_REG2 Register Field Descriptions .....     | 1909 |
| 16-161. AD_INTERFACE_REG3 Register Field Descriptions .....     | 1911 |
| 16-162. ANA_CONFIG2 Register Field Descriptions .....           | 1912 |
| 16-163. CPPI_DMA REGISTERS .....                                | 1912 |
| 16-164. DMAREVID Register Field Descriptions .....              | 1916 |
| 16-165. TDFDQ Register Field Descriptions .....                 | 1917 |
| 16-166. DMAEMU Register Field Descriptions .....                | 1918 |
| 16-167. TXGCR0 Register Field Descriptions .....                | 1919 |
| 16-168. RXGCR0 Register Field Descriptions .....                | 1920 |
| 16-169. RXHPCRA0 Register Field Descriptions.....               | 1922 |
| 16-170. RXHPCRB0 Register Field Descriptions .....              | 1923 |
| 16-171. TXGCR1 Register Field Descriptions .....                | 1924 |
| 16-172. RXHPCRA1 Register Field Descriptions .....              | 1925 |
| 16-173. RXHPCRB1 Register Field Descriptions.....               | 1927 |
| 16-174. RXHPCRB2 Register Field Descriptions .....              | 1928 |
| 16-175. TXGCR2 Register Field Descriptions .....                | 1929 |
| 16-176. RXHPCRA2 Register Field Descriptions .....              | 1930 |
| 16-177. RXHPCRA3 Register Field Descriptions .....              | 1932 |
| 16-178. RXHPCRB3 Register Field Descriptions .....              | 1933 |
| 16-179. TXGCR3 Register Field Descriptions .....                | 1934 |
| 16-180. RXHPCRA4 Register Field Descriptions .....              | 1935 |
| 16-181. RXHPCRB4 Register Field Descriptions .....              | 1937 |
| 16-182. RXHPCRA5 Register Field Descriptions .....              | 1938 |
| 16-183. TXGCR4 Register Field Descriptions .....                | 1939 |
| 16-184. RXHPCRB5 Register Field Descriptions .....              | 1940 |
| 16-185. RXHPCRA6 Register Field Descriptions .....              | 1942 |
| 16-186. RXHPCRB6 Register Field Descriptions .....              | 1943 |
| 16-187. TXGCR5 Register Field Descriptions .....                | 1944 |
| 16-188. RXGCR5 Register Field Descriptions .....                | 1945 |
| 16-189. RXHPCRA7 Register Field Descriptions .....              | 1947 |
| 16-190. RXHPCRB7 Register Field Descriptions .....              | 1948 |
| 16-191. TXGCR6 Register Field Descriptions .....                | 1949 |
| 16-192. RXGCR6 Register Field Descriptions .....                | 1950 |
| 16-193. RXHPCRA8 Register Field Descriptions .....              | 1952 |
| 16-194. RXHPCRB8 Register Field Descriptions .....              | 1953 |
| 16-195. TXGCR7 Register Field Descriptions .....                | 1954 |
| 16-196. RXGCR7 Register Field Descriptions .....                | 1955 |
| 16-197. RXHPCRA9 Register Field Descriptions .....              | 1957 |
| 16-198. RXHPCRB9 Register Field Descriptions .....              | 1958 |
| 16-199. TXGCR8 Register Field Descriptions .....                | 1959 |

|         |   |      |
|---------|---|------|
| 16-200. | RXGCR8 Register Field Descriptions .....    | 1960 |
| 16-201. | RXHPCRA8 Register Field Descriptions .....  | 1962 |
| 16-202. | RXHPCRB8 Register Field Descriptions .....  | 1963 |
| 16-203. | TXGCR9 Register Field Descriptions .....    | 1964 |
| 16-204. | RXGCR9 Register Field Descriptions .....    | 1965 |
| 16-205. | RXHPCRA9 Register Field Descriptions .....  | 1967 |
| 16-206. | RXHPCRB9 Register Field Descriptions .....  | 1968 |
| 16-207. | TXGCR10 Register Field Descriptions .....   | 1969 |
| 16-208. | RXGCR10 Register Field Descriptions .....   | 1970 |
| 16-209. | RXHPCRA10 Register Field Descriptions ..... | 1972 |
| 16-210. | RXHPCRB10 Register Field Descriptions ..... | 1973 |
| 16-211. | TXGCR11 Register Field Descriptions .....   | 1974 |
| 16-212. | RXGCR11 Register Field Descriptions .....   | 1975 |
| 16-213. | RXHPCRA11 Register Field Descriptions ..... | 1977 |
| 16-214. | RXHPCRB11 Register Field Descriptions ..... | 1978 |
| 16-215. | TXGCR12 Register Field Descriptions .....   | 1979 |
| 16-216. | RXGCR12 Register Field Descriptions .....   | 1980 |
| 16-217. | RXHPCRA12 Register Field Descriptions ..... | 1982 |
| 16-218. | RXHPCRB12 Register Field Descriptions ..... | 1983 |
| 16-219. | TXGCR13 Register Field Descriptions .....   | 1984 |
| 16-220. | RXGCR13 Register Field Descriptions .....   | 1985 |
| 16-221. | RXHPCRA13 Register Field Descriptions ..... | 1987 |
| 16-222. | RXHPCRB13 Register Field Descriptions ..... | 1988 |
| 16-223. | TXGCR14 Register Field Descriptions .....   | 1989 |
| 16-224. | RXGCR14 Register Field Descriptions .....   | 1990 |
| 16-225. | RXHPCRA14 Register Field Descriptions ..... | 1992 |
| 16-226. | RXHPCRB14 Register Field Descriptions ..... | 1993 |
| 16-227. | TXGCR15 Register Field Descriptions .....   | 1994 |
| 16-228. | RXGCR15 Register Field Descriptions .....   | 1995 |
| 16-229. | RXHPCRA15 Register Field Descriptions ..... | 1997 |
| 16-230. | RXHPCRB15 Register Field Descriptions ..... | 1998 |
| 16-231. | TXGCR16 Register Field Descriptions .....   | 1999 |
| 16-232. | RXGCR16 Register Field Descriptions .....   | 2000 |
| 16-233. | RXHPCRA16 Register Field Descriptions ..... | 2002 |
| 16-234. | RXHPCRB16 Register Field Descriptions ..... | 2003 |
| 16-235. | TXGCR17 Register Field Descriptions .....   | 2004 |
| 16-236. | RXGCR17 Register Field Descriptions .....   | 2005 |
| 16-237. | RXHPCRA17 Register Field Descriptions ..... | 2007 |
| 16-238. | RXHPCRB17 Register Field Descriptions ..... | 2008 |
| 16-239. | TXGCR18 Register Field Descriptions .....   | 2009 |
| 16-240. | RXGCR18 Register Field Descriptions .....   | 2010 |
| 16-241. | RXHPCRA18 Register Field Descriptions ..... | 2012 |
| 16-242. | RXHPCRB18 Register Field Descriptions ..... | 2013 |
| 16-243. | TXGCR19 Register Field Descriptions .....   | 2014 |
| 16-244. | RXGCR19 Register Field Descriptions .....   | 2015 |
| 16-245. | RXHPCRA19 Register Field Descriptions ..... | 2017 |
| 16-246. | RXHPCRB19 Register Field Descriptions ..... | 2018 |
| 16-247. | TXGCR20 Register Field Descriptions .....   | 2019 |
| 16-248. | RXGCR20 Register Field Descriptions .....   | 2020 |

|   |      |
|---|------|
| 16-249. RXHPCRA20 Register Field Descriptions .....       | 2022 |
| 16-250. RXHPCRB20 Register Field Descriptions .....       | 2023 |
| 16-251. TXGCR21 Register Field Descriptions .....         | 2024 |
| 16-252. RXGCR21 Register Field Descriptions .....         | 2025 |
| 16-253. RXHPCRA21 Register Field Descriptions .....       | 2027 |
| 16-254. RXHPCRB21 Register Field Descriptions .....       | 2028 |
| 16-255. TXGCR22 Register Field Descriptions .....         | 2029 |
| 16-256. RXGCR22 Register Field Descriptions .....         | 2030 |
| 16-257. RXHPCRA22 Register Field Descriptions .....       | 2032 |
| 16-258. RXHPCRB22 Register Field Descriptions .....       | 2033 |
| 16-259. TXGCR23 Register Field Descriptions .....         | 2034 |
| 16-260. RXGCR23 Register Field Descriptions .....         | 2035 |
| 16-261. RXHPCRA23 Register Field Descriptions .....       | 2037 |
| 16-262. RXHPCRB23 Register Field Descriptions .....       | 2038 |
| 16-263. TXGCR24 Register Field Descriptions .....         | 2039 |
| 16-264. RXGCR24 Register Field Descriptions .....         | 2040 |
| 16-265. RXHPCRA24 Register Field Descriptions .....       | 2042 |
| 16-266. RXHPCRB24 Register Field Descriptions .....       | 2043 |
| 16-267. TXGCR25 Register Field Descriptions .....         | 2044 |
| 16-268. RXGCR25 Register Field Descriptions .....         | 2045 |
| 16-269. RXHPCRA25 Register Field Descriptions .....       | 2047 |
| 16-270. RXHPCRB25 Register Field Descriptions .....       | 2048 |
| 16-271. TXGCR26 Register Field Descriptions .....         | 2049 |
| 16-272. RXGCR26 Register Field Descriptions .....         | 2050 |
| 16-273. RXHPCRA26 Register Field Descriptions .....       | 2052 |
| 16-274. RXHPCRB26 Register Field Descriptions .....       | 2053 |
| 16-275. TXGCR27 Register Field Descriptions .....         | 2054 |
| 16-276. RXGCR27 Register Field Descriptions .....         | 2055 |
| 16-277. RXHPCRA27 Register Field Descriptions .....       | 2057 |
| 16-278. RXHPCRB27 Register Field Descriptions .....       | 2058 |
| 16-279. TXGCR28 Register Field Descriptions .....         | 2059 |
| 16-280. RXGCR28 Register Field Descriptions .....         | 2060 |
| 16-281. RXHPCRA28 Register Field Descriptions .....       | 2062 |
| 16-282. RXHPCRB28 Register Field Descriptions .....       | 2063 |
| 16-283. TXGCR29 Register Field Descriptions .....         | 2064 |
| 16-284. RXGCR29 Register Field Descriptions .....         | 2065 |
| 16-285. RXHPCRA29 Register Field Descriptions .....       | 2067 |
| 16-286. RXHPCRB29 Register Field Descriptions .....       | 2068 |
| 16-287. CPPI_DMA_SCHEDULER REGISTERS .....                | 2068 |
| 16-288. DMA_SCHED_CTRL Register Field Descriptions .....  | 2069 |
| 16-289. WORD0 to WORD63 Register Field Descriptions ..... | 2070 |
| 16-290. QUEUE_MGR REGISTERS .....                         | 2071 |
| 16-291. QMGRREVID Register Field Descriptions .....       | 2096 |
| 16-292. QMGRRST Register Field Descriptions .....         | 2097 |
| 16-293. FDBSC0 Register Field Descriptions .....          | 2098 |
| 16-294. FDBSC1 Register Field Descriptions .....          | 2099 |
| 16-295. FDBSC2 Register Field Descriptions .....          | 2100 |
| 16-296. FDBSC3 Register Field Descriptions .....          | 2101 |
| 16-297. FDBSC4 Register Field Descriptions .....          | 2102 |

|  |      |
|--|------|
| 16-298. FDBSC5 Register Field Descriptions .....     | 2103 |
| 16-299. FDBSC6 Register Field Descriptions .....     | 2104 |
| 16-300. FDBSC7 Register Field Descriptions .....     | 2105 |
| 16-301. LRAM0BASE Register Field Descriptions .....  | 2106 |
| 16-302. LRAM0SIZE Register Field Descriptions .....  | 2107 |
| 16-303. LRAM1BASE Register Field Descriptions .....  | 2108 |
| 16-304. PEND0 Register Field Descriptions .....      | 2109 |
| 16-305. PEND1 Register Field Descriptions .....      | 2110 |
| 16-306. PEND2 Register Field Descriptions .....      | 2111 |
| 16-307. PEND3 Register Field Descriptions .....      | 2112 |
| 16-308. PEND4 Register Field Descriptions .....      | 2113 |
| 16-309. QMEMRBASE0 Register Field Descriptions ..... | 2114 |
| 16-310. QMEMCTRL0 Register Field Descriptions .....  | 2115 |
| 16-311. QMEMRBASE1 Register Field Descriptions ..... | 2116 |
| 16-312. QMEMCTRL1 Register Field Descriptions .....  | 2117 |
| 16-313. QMEMRBASE2 Register Field Descriptions ..... | 2118 |
| 16-314. QMEMCTRL2 Register Field Descriptions .....  | 2119 |
| 16-315. QMEMRBASE3 Register Field Descriptions ..... | 2120 |
| 16-316. QMEMCTRL3 Register Field Descriptions .....  | 2121 |
| 16-317. QMEMRBASE4 Register Field Descriptions ..... | 2122 |
| 16-318. QMEMCTRL4 Register Field Descriptions .....  | 2123 |
| 16-319. QMEMRBASE5 Register Field Descriptions ..... | 2124 |
| 16-320. QMEMCTRL5 Register Field Descriptions .....  | 2125 |
| 16-321. QMEMRBASE6 Register Field Descriptions ..... | 2126 |
| 16-322. QMEMCTRL6 Register Field Descriptions .....  | 2127 |
| 16-323. QMEMRBASE7 Register Field Descriptions ..... | 2128 |
| 16-324. QMEMCTRL7 Register Field Descriptions .....  | 2129 |
| 16-325. QUEUE_0_A Register Field Descriptions .....  | 2130 |
| 16-326. QUEUE_0_B Register Field Descriptions .....  | 2131 |
| 16-327. QUEUE_0_C Register Field Descriptions .....  | 2132 |
| 16-328. QUEUE_0_D Register Field Descriptions .....  | 2133 |
| 16-329. QUEUE_1_A Register Field Descriptions .....  | 2134 |
| 16-330. QUEUE_1_B Register Field Descriptions .....  | 2135 |
| 16-331. QUEUE_1_C Register Field Descriptions .....  | 2136 |
| 16-332. QUEUE_1_D Register Field Descriptions .....  | 2137 |
| 16-333. QUEUE_2_A Register Field Descriptions .....  | 2138 |
| 16-334. QUEUE_2_B Register Field Descriptions .....  | 2139 |
| 16-335. QUEUE_2_C Register Field Descriptions .....  | 2140 |
| 16-336. QUEUE_2_D Register Field Descriptions .....  | 2141 |
| 16-337. QUEUE_3_A Register Field Descriptions .....  | 2142 |
| 16-338. QUEUE_3_B Register Field Descriptions .....  | 2143 |
| 16-339. QUEUE_3_C Register Field Descriptions .....  | 2144 |
| 16-340. QUEUE_3_D Register Field Descriptions .....  | 2145 |
| 16-341. QUEUE_4_A Register Field Descriptions .....  | 2146 |
| 16-342. QUEUE_4_B Register Field Descriptions .....  | 2147 |
| 16-343. QUEUE_4_C Register Field Descriptions .....  | 2148 |
| 16-344. QUEUE_4_D Register Field Descriptions .....  | 2149 |
| 16-345. QUEUE_5_A Register Field Descriptions .....  | 2150 |
| 16-346. QUEUE_5_B Register Field Descriptions .....  | 2151 |

|  |      |
|--|------|
| 16-347. QUEUE_5_C Register Field Descriptions .....  | 2152 |
| 16-348. QUEUE_5_D Register Field Descriptions .....  | 2153 |
| 16-349. QUEUE_6_A Register Field Descriptions.....   | 2154 |
| 16-350. QUEUE_6_B Register Field Descriptions.....   | 2155 |
| 16-351. QUEUE_6_C Register Field Descriptions .....  | 2156 |
| 16-352. QUEUE_6_D Register Field Descriptions .....  | 2157 |
| 16-353. QUEUE_7_A Register Field Descriptions.....   | 2158 |
| 16-354. QUEUE_7_B Register Field Descriptions.....   | 2159 |
| 16-355. QUEUE_7_C Register Field Descriptions .....  | 2160 |
| 16-356. QUEUE_7_D Register Field Descriptions .....  | 2161 |
| 16-357. QUEUE_8_A Register Field Descriptions.....   | 2162 |
| 16-358. QUEUE_8_B Register Field Descriptions.....   | 2163 |
| 16-359. QUEUE_8_C Register Field Descriptions .....  | 2164 |
| 16-360. QUEUE_8_D Register Field Descriptions .....  | 2165 |
| 16-361. QUEUE_9_A Register Field Descriptions.....   | 2166 |
| 16-362. QUEUE_9_B Register Field Descriptions.....   | 2167 |
| 16-363. QUEUE_9_C Register Field Descriptions .....  | 2168 |
| 16-364. QUEUE_9_D Register Field Descriptions .....  | 2169 |
| 16-365. QUEUE_10_A Register Field Descriptions ..... | 2170 |
| 16-366. QUEUE_10_B Register Field Descriptions ..... | 2171 |
| 16-367. QUEUE_10_C Register Field Descriptions ..... | 2172 |
| 16-368. QUEUE_10_D Register Field Descriptions ..... | 2173 |
| 16-369. QUEUE_11_A Register Field Descriptions ..... | 2174 |
| 16-370. QUEUE_11_B Register Field Descriptions ..... | 2175 |
| 16-371. QUEUE_11_C Register Field Descriptions ..... | 2176 |
| 16-372. QUEUE_11_D Register Field Descriptions ..... | 2177 |
| 16-373. QUEUE_12_A Register Field Descriptions ..... | 2178 |
| 16-374. QUEUE_12_B Register Field Descriptions ..... | 2179 |
| 16-375. QUEUE_12_C Register Field Descriptions ..... | 2180 |
| 16-376. QUEUE_12_D Register Field Descriptions ..... | 2181 |
| 16-377. QUEUE_13_A Register Field Descriptions ..... | 2182 |
| 16-378. QUEUE_13_B Register Field Descriptions ..... | 2183 |
| 16-379. QUEUE_13_C Register Field Descriptions ..... | 2184 |
| 16-380. QUEUE_13_D Register Field Descriptions ..... | 2185 |
| 16-381. QUEUE_14_A Register Field Descriptions ..... | 2186 |
| 16-382. QUEUE_14_B Register Field Descriptions ..... | 2187 |
| 16-383. QUEUE_14_C Register Field Descriptions ..... | 2188 |
| 16-384. QUEUE_14_D Register Field Descriptions ..... | 2189 |
| 16-385. QUEUE_15_A Register Field Descriptions ..... | 2190 |
| 16-386. QUEUE_15_B Register Field Descriptions ..... | 2191 |
| 16-387. QUEUE_15_C Register Field Descriptions ..... | 2192 |
| 16-388. QUEUE_15_D Register Field Descriptions ..... | 2193 |
| 16-389. QUEUE_16_A Register Field Descriptions ..... | 2194 |
| 16-390. QUEUE_16_B Register Field Descriptions ..... | 2195 |
| 16-391. QUEUE_16_C Register Field Descriptions ..... | 2196 |
| 16-392. QUEUE_16_D Register Field Descriptions ..... | 2197 |
| 16-393. QUEUE_17_A Register Field Descriptions ..... | 2198 |
| 16-394. QUEUE_17_B Register Field Descriptions ..... | 2199 |
| 16-395. QUEUE_17_C Register Field Descriptions ..... | 2200 |

---

|  |      |
|--|------|
| 16-396. QUEUE_17_D Register Field Descriptions ..... | 2201 |
| 16-397. QUEUE_18_A Register Field Descriptions ..... | 2202 |
| 16-398. QUEUE_18_B Register Field Descriptions ..... | 2203 |
| 16-399. QUEUE_18_C Register Field Descriptions ..... | 2204 |
| 16-400. QUEUE_18_D Register Field Descriptions ..... | 2205 |
| 16-401. QUEUE_19_A Register Field Descriptions ..... | 2206 |
| 16-402. QUEUE_19_B Register Field Descriptions ..... | 2207 |
| 16-403. QUEUE_19_C Register Field Descriptions ..... | 2208 |
| 16-404. QUEUE_19_D Register Field Descriptions ..... | 2209 |
| 16-405. QUEUE_20_A Register Field Descriptions ..... | 2210 |
| 16-406. QUEUE_20_B Register Field Descriptions ..... | 2211 |
| 16-407. QUEUE_20_C Register Field Descriptions ..... | 2212 |
| 16-408. QUEUE_20_D Register Field Descriptions ..... | 2213 |
| 16-409. QUEUE_21_A Register Field Descriptions ..... | 2214 |
| 16-410. QUEUE_21_B Register Field Descriptions ..... | 2215 |
| 16-411. QUEUE_21_C Register Field Descriptions ..... | 2216 |
| 16-412. QUEUE_21_D Register Field Descriptions ..... | 2217 |
| 16-413. QUEUE_22_A Register Field Descriptions ..... | 2218 |
| 16-414. QUEUE_22_B Register Field Descriptions ..... | 2219 |
| 16-415. QUEUE_22_C Register Field Descriptions ..... | 2220 |
| 16-416. QUEUE_22_D Register Field Descriptions ..... | 2221 |
| 16-417. QUEUE_23_A Register Field Descriptions ..... | 2222 |
| 16-418. QUEUE_23_B Register Field Descriptions ..... | 2223 |
| 16-419. QUEUE_23_C Register Field Descriptions ..... | 2224 |
| 16-420. QUEUE_23_D Register Field Descriptions ..... | 2225 |
| 16-421. QUEUE_24_A Register Field Descriptions ..... | 2226 |
| 16-422. QUEUE_24_B Register Field Descriptions ..... | 2227 |
| 16-423. QUEUE_24_C Register Field Descriptions ..... | 2228 |
| 16-424. QUEUE_24_D Register Field Descriptions ..... | 2229 |
| 16-425. QUEUE_25_A Register Field Descriptions ..... | 2230 |
| 16-426. QUEUE_25_B Register Field Descriptions ..... | 2231 |
| 16-427. QUEUE_25_C Register Field Descriptions ..... | 2232 |
| 16-428. QUEUE_25_D Register Field Descriptions ..... | 2233 |
| 16-429. QUEUE_26_A Register Field Descriptions ..... | 2234 |
| 16-430. QUEUE_26_B Register Field Descriptions ..... | 2235 |
| 16-431. QUEUE_26_C Register Field Descriptions ..... | 2236 |
| 16-432. QUEUE_26_D Register Field Descriptions ..... | 2237 |
| 16-433. QUEUE_27_A Register Field Descriptions ..... | 2238 |
| 16-434. QUEUE_27_B Register Field Descriptions ..... | 2239 |
| 16-435. QUEUE_27_C Register Field Descriptions ..... | 2240 |
| 16-436. QUEUE_27_D Register Field Descriptions ..... | 2241 |
| 16-437. QUEUE_28_A Register Field Descriptions ..... | 2242 |
| 16-438. QUEUE_28_B Register Field Descriptions ..... | 2243 |
| 16-439. QUEUE_28_C Register Field Descriptions ..... | 2244 |
| 16-440. QUEUE_28_D Register Field Descriptions ..... | 2245 |
| 16-441. QUEUE_29_A Register Field Descriptions ..... | 2246 |
| 16-442. QUEUE_29_B Register Field Descriptions ..... | 2247 |
| 16-443. QUEUE_29_C Register Field Descriptions ..... | 2248 |
| 16-444. QUEUE_29_D Register Field Descriptions ..... | 2249 |

|  |      |
|--|------|
| 16-445. QUEUE_30_A Register Field Descriptions ..... | 2250 |
| 16-446. QUEUE_30_B Register Field Descriptions ..... | 2251 |
| 16-447. QUEUE_30_C Register Field Descriptions ..... | 2252 |
| 16-448. QUEUE_30_D Register Field Descriptions ..... | 2253 |
| 16-449. QUEUE_31_A Register Field Descriptions ..... | 2254 |
| 16-450. QUEUE_31_B Register Field Descriptions ..... | 2255 |
| 16-451. QUEUE_31_C Register Field Descriptions ..... | 2256 |
| 16-452. QUEUE_31_D Register Field Descriptions ..... | 2257 |
| 16-453. QUEUE_32_A Register Field Descriptions ..... | 2258 |
| 16-454. QUEUE_32_B Register Field Descriptions ..... | 2259 |
| 16-455. QUEUE_32_C Register Field Descriptions ..... | 2260 |
| 16-456. QUEUE_32_D Register Field Descriptions ..... | 2261 |
| 16-457. QUEUE_33_A Register Field Descriptions ..... | 2262 |
| 16-458. QUEUE_33_B Register Field Descriptions ..... | 2263 |
| 16-459. QUEUE_33_C Register Field Descriptions ..... | 2264 |
| 16-460. QUEUE_33_D Register Field Descriptions ..... | 2265 |
| 16-461. QUEUE_34_A Register Field Descriptions ..... | 2266 |
| 16-462. QUEUE_34_B Register Field Descriptions ..... | 2267 |
| 16-463. QUEUE_34_C Register Field Descriptions ..... | 2268 |
| 16-464. QUEUE_34_D Register Field Descriptions ..... | 2269 |
| 16-465. QUEUE_35_A Register Field Descriptions ..... | 2270 |
| 16-466. QUEUE_35_B Register Field Descriptions ..... | 2271 |
| 16-467. QUEUE_35_C Register Field Descriptions ..... | 2272 |
| 16-468. QUEUE_35_D Register Field Descriptions ..... | 2273 |
| 16-469. QUEUE_36_A Register Field Descriptions ..... | 2274 |
| 16-470. QUEUE_36_B Register Field Descriptions ..... | 2275 |
| 16-471. QUEUE_36_C Register Field Descriptions ..... | 2276 |
| 16-472. QUEUE_36_D Register Field Descriptions ..... | 2277 |
| 16-473. QUEUE_37_A Register Field Descriptions ..... | 2278 |
| 16-474. QUEUE_37_B Register Field Descriptions ..... | 2279 |
| 16-475. QUEUE_37_C Register Field Descriptions ..... | 2280 |
| 16-476. QUEUE_37_D Register Field Descriptions ..... | 2281 |
| 16-477. QUEUE_38_A Register Field Descriptions ..... | 2282 |
| 16-478. QUEUE_38_B Register Field Descriptions ..... | 2283 |
| 16-479. QUEUE_38_C Register Field Descriptions ..... | 2284 |
| 16-480. QUEUE_38_D Register Field Descriptions ..... | 2285 |
| 16-481. QUEUE_39_A Register Field Descriptions ..... | 2286 |
| 16-482. QUEUE_39_B Register Field Descriptions ..... | 2287 |
| 16-483. QUEUE_39_C Register Field Descriptions ..... | 2288 |
| 16-484. QUEUE_39_D Register Field Descriptions ..... | 2289 |
| 16-485. QUEUE_40_A Register Field Descriptions ..... | 2290 |
| 16-486. QUEUE_40_B Register Field Descriptions ..... | 2291 |
| 16-487. QUEUE_40_C Register Field Descriptions ..... | 2292 |
| 16-488. QUEUE_40_D Register Field Descriptions ..... | 2293 |
| 16-489. QUEUE_41_A Register Field Descriptions ..... | 2294 |
| 16-490. QUEUE_41_B Register Field Descriptions ..... | 2295 |
| 16-491. QUEUE_41_C Register Field Descriptions ..... | 2296 |
| 16-492. QUEUE_41_D Register Field Descriptions ..... | 2297 |
| 16-493. QUEUE_42_A Register Field Descriptions ..... | 2298 |

---

|  |      |
|--|------|
| 16-494. QUEUE_42_B Register Field Descriptions ..... | 2299 |
| 16-495. QUEUE_42_C Register Field Descriptions ..... | 2300 |
| 16-496. QUEUE_42_D Register Field Descriptions ..... | 2301 |
| 16-497. QUEUE_43_A Register Field Descriptions ..... | 2302 |
| 16-498. QUEUE_43_B Register Field Descriptions ..... | 2303 |
| 16-499. QUEUE_43_C Register Field Descriptions ..... | 2304 |
| 16-500. QUEUE_43_D Register Field Descriptions ..... | 2305 |
| 16-501. QUEUE_44_A Register Field Descriptions ..... | 2306 |
| 16-502. QUEUE_44_B Register Field Descriptions ..... | 2307 |
| 16-503. QUEUE_44_C Register Field Descriptions ..... | 2308 |
| 16-504. QUEUE_44_D Register Field Descriptions ..... | 2309 |
| 16-505. QUEUE_45_A Register Field Descriptions ..... | 2310 |
| 16-506. QUEUE_45_B Register Field Descriptions ..... | 2311 |
| 16-507. QUEUE_45_C Register Field Descriptions ..... | 2312 |
| 16-508. QUEUE_45_D Register Field Descriptions ..... | 2313 |
| 16-509. QUEUE_46_A Register Field Descriptions ..... | 2314 |
| 16-510. QUEUE_46_B Register Field Descriptions ..... | 2315 |
| 16-511. QUEUE_46_C Register Field Descriptions ..... | 2316 |
| 16-512. QUEUE_46_D Register Field Descriptions ..... | 2317 |
| 16-513. QUEUE_47_A Register Field Descriptions ..... | 2318 |
| 16-514. QUEUE_47_B Register Field Descriptions ..... | 2319 |
| 16-515. QUEUE_47_C Register Field Descriptions ..... | 2320 |
| 16-516. QUEUE_47_D Register Field Descriptions ..... | 2321 |
| 16-517. QUEUE_48_A Register Field Descriptions ..... | 2322 |
| 16-518. QUEUE_48_B Register Field Descriptions ..... | 2323 |
| 16-519. QUEUE_48_C Register Field Descriptions ..... | 2324 |
| 16-520. QUEUE_48_D Register Field Descriptions ..... | 2325 |
| 16-521. QUEUE_49_A Register Field Descriptions ..... | 2326 |
| 16-522. QUEUE_49_B Register Field Descriptions ..... | 2327 |
| 16-523. QUEUE_49_C Register Field Descriptions ..... | 2328 |
| 16-524. QUEUE_49_D Register Field Descriptions ..... | 2329 |
| 16-525. QUEUE_50_A Register Field Descriptions ..... | 2330 |
| 16-526. QUEUE_50_B Register Field Descriptions ..... | 2331 |
| 16-527. QUEUE_50_C Register Field Descriptions ..... | 2332 |
| 16-528. QUEUE_50_D Register Field Descriptions ..... | 2333 |
| 16-529. QUEUE_51_A Register Field Descriptions ..... | 2334 |
| 16-530. QUEUE_51_B Register Field Descriptions ..... | 2335 |
| 16-531. QUEUE_51_C Register Field Descriptions ..... | 2336 |
| 16-532. QUEUE_51_D Register Field Descriptions ..... | 2337 |
| 16-533. QUEUE_52_A Register Field Descriptions ..... | 2338 |
| 16-534. QUEUE_52_B Register Field Descriptions ..... | 2339 |
| 16-535. QUEUE_52_C Register Field Descriptions ..... | 2340 |
| 16-536. QUEUE_52_D Register Field Descriptions ..... | 2341 |
| 16-537. QUEUE_53_A Register Field Descriptions ..... | 2342 |
| 16-538. QUEUE_53_B Register Field Descriptions ..... | 2343 |
| 16-539. QUEUE_53_C Register Field Descriptions ..... | 2344 |
| 16-540. QUEUE_53_D Register Field Descriptions ..... | 2345 |
| 16-541. QUEUE_54_A Register Field Descriptions ..... | 2346 |
| 16-542. QUEUE_54_B Register Field Descriptions ..... | 2347 |

|  |      |
|--|------|
| 16-543. QUEUE_54_C Register Field Descriptions ..... | 2348 |
| 16-544. QUEUE_54_D Register Field Descriptions ..... | 2349 |
| 16-545. QUEUE_55_A Register Field Descriptions ..... | 2350 |
| 16-546. QUEUE_55_B Register Field Descriptions ..... | 2351 |
| 16-547. QUEUE_55_C Register Field Descriptions ..... | 2352 |
| 16-548. QUEUE_55_D Register Field Descriptions ..... | 2353 |
| 16-549. QUEUE_56_A Register Field Descriptions ..... | 2354 |
| 16-550. QUEUE_56_B Register Field Descriptions ..... | 2355 |
| 16-551. QUEUE_56_C Register Field Descriptions ..... | 2356 |
| 16-552. QUEUE_56_D Register Field Descriptions ..... | 2357 |
| 16-553. QUEUE_57_A Register Field Descriptions ..... | 2358 |
| 16-554. QUEUE_57_B Register Field Descriptions ..... | 2359 |
| 16-555. QUEUE_57_C Register Field Descriptions ..... | 2360 |
| 16-556. QUEUE_57_D Register Field Descriptions ..... | 2361 |
| 16-557. QUEUE_58_A Register Field Descriptions ..... | 2362 |
| 16-558. QUEUE_58_B Register Field Descriptions ..... | 2363 |
| 16-559. QUEUE_58_C Register Field Descriptions ..... | 2364 |
| 16-560. QUEUE_58_D Register Field Descriptions ..... | 2365 |
| 16-561. QUEUE_59_A Register Field Descriptions ..... | 2366 |
| 16-562. QUEUE_59_B Register Field Descriptions ..... | 2367 |
| 16-563. QUEUE_59_C Register Field Descriptions ..... | 2368 |
| 16-564. QUEUE_59_D Register Field Descriptions ..... | 2369 |
| 16-565. QUEUE_60_A Register Field Descriptions ..... | 2370 |
| 16-566. QUEUE_60_B Register Field Descriptions ..... | 2371 |
| 16-567. QUEUE_60_C Register Field Descriptions ..... | 2372 |
| 16-568. QUEUE_60_D Register Field Descriptions ..... | 2373 |
| 16-569. QUEUE_61_A Register Field Descriptions ..... | 2374 |
| 16-570. QUEUE_61_B Register Field Descriptions ..... | 2375 |
| 16-571. QUEUE_61_C Register Field Descriptions ..... | 2376 |
| 16-572. QUEUE_61_D Register Field Descriptions ..... | 2377 |
| 16-573. QUEUE_62_A Register Field Descriptions ..... | 2378 |
| 16-574. QUEUE_62_B Register Field Descriptions ..... | 2379 |
| 16-575. QUEUE_62_C Register Field Descriptions ..... | 2380 |
| 16-576. QUEUE_62_D Register Field Descriptions ..... | 2381 |
| 16-577. QUEUE_63_A Register Field Descriptions ..... | 2382 |
| 16-578. QUEUE_63_B Register Field Descriptions ..... | 2383 |
| 16-579. QUEUE_63_C Register Field Descriptions ..... | 2384 |
| 16-580. QUEUE_63_D Register Field Descriptions ..... | 2385 |
| 16-581. QUEUE_64_A Register Field Descriptions ..... | 2386 |
| 16-582. QUEUE_64_B Register Field Descriptions ..... | 2387 |
| 16-583. QUEUE_64_C Register Field Descriptions ..... | 2388 |
| 16-584. QUEUE_64_D Register Field Descriptions ..... | 2389 |
| 16-585. QUEUE_65_A Register Field Descriptions ..... | 2390 |
| 16-586. QUEUE_65_B Register Field Descriptions ..... | 2391 |
| 16-587. QUEUE_65_C Register Field Descriptions ..... | 2392 |
| 16-588. QUEUE_65_D Register Field Descriptions ..... | 2393 |
| 16-589. QUEUE_66_A Register Field Descriptions ..... | 2394 |
| 16-590. QUEUE_66_B Register Field Descriptions ..... | 2395 |
| 16-591. QUEUE_66_C Register Field Descriptions ..... | 2396 |

---

|  |      |
|--|------|
| 16-592. QUEUE_66_D Register Field Descriptions ..... | 2397 |
| 16-593. QUEUE_67_A Register Field Descriptions ..... | 2398 |
| 16-594. QUEUE_67_B Register Field Descriptions ..... | 2399 |
| 16-595. QUEUE_67_C Register Field Descriptions ..... | 2400 |
| 16-596. QUEUE_67_D Register Field Descriptions ..... | 2401 |
| 16-597. QUEUE_68_A Register Field Descriptions ..... | 2402 |
| 16-598. QUEUE_68_B Register Field Descriptions ..... | 2403 |
| 16-599. QUEUE_68_C Register Field Descriptions ..... | 2404 |
| 16-600. QUEUE_68_D Register Field Descriptions ..... | 2405 |
| 16-601. QUEUE_69_A Register Field Descriptions ..... | 2406 |
| 16-602. QUEUE_69_B Register Field Descriptions ..... | 2407 |
| 16-603. QUEUE_69_C Register Field Descriptions ..... | 2408 |
| 16-604. QUEUE_69_D Register Field Descriptions ..... | 2409 |
| 16-605. QUEUE_70_A Register Field Descriptions ..... | 2410 |
| 16-606. QUEUE_70_B Register Field Descriptions ..... | 2411 |
| 16-607. QUEUE_70_C Register Field Descriptions ..... | 2412 |
| 16-608. QUEUE_70_D Register Field Descriptions ..... | 2413 |
| 16-609. QUEUE_71_A Register Field Descriptions ..... | 2414 |
| 16-610. QUEUE_71_B Register Field Descriptions ..... | 2415 |
| 16-611. QUEUE_71_C Register Field Descriptions ..... | 2416 |
| 16-612. QUEUE_71_D Register Field Descriptions ..... | 2417 |
| 16-613. QUEUE_72_A Register Field Descriptions ..... | 2418 |
| 16-614. QUEUE_72_B Register Field Descriptions ..... | 2419 |
| 16-615. QUEUE_72_C Register Field Descriptions ..... | 2420 |
| 16-616. QUEUE_72_D Register Field Descriptions ..... | 2421 |
| 16-617. QUEUE_73_A Register Field Descriptions ..... | 2422 |
| 16-618. QUEUE_73_B Register Field Descriptions ..... | 2423 |
| 16-619. QUEUE_73_C Register Field Descriptions ..... | 2424 |
| 16-620. QUEUE_73_D Register Field Descriptions ..... | 2425 |
| 16-621. QUEUE_74_A Register Field Descriptions ..... | 2426 |
| 16-622. QUEUE_74_B Register Field Descriptions ..... | 2427 |
| 16-623. QUEUE_74_C Register Field Descriptions ..... | 2428 |
| 16-624. QUEUE_74_D Register Field Descriptions ..... | 2429 |
| 16-625. QUEUE_75_A Register Field Descriptions ..... | 2430 |
| 16-626. QUEUE_75_B Register Field Descriptions ..... | 2431 |
| 16-627. QUEUE_75_C Register Field Descriptions ..... | 2432 |
| 16-628. QUEUE_75_D Register Field Descriptions ..... | 2433 |
| 16-629. QUEUE_76_A Register Field Descriptions ..... | 2434 |
| 16-630. QUEUE_76_B Register Field Descriptions ..... | 2435 |
| 16-631. QUEUE_76_C Register Field Descriptions ..... | 2436 |
| 16-632. QUEUE_76_D Register Field Descriptions ..... | 2437 |
| 16-633. QUEUE_77_A Register Field Descriptions ..... | 2438 |
| 16-634. QUEUE_77_B Register Field Descriptions ..... | 2439 |
| 16-635. QUEUE_77_C Register Field Descriptions ..... | 2440 |
| 16-636. QUEUE_77_D Register Field Descriptions ..... | 2441 |
| 16-637. QUEUE_78_A Register Field Descriptions ..... | 2442 |
| 16-638. QUEUE_78_B Register Field Descriptions ..... | 2443 |
| 16-639. QUEUE_78_C Register Field Descriptions ..... | 2444 |
| 16-640. QUEUE_78_D Register Field Descriptions ..... | 2445 |

|  |      |
|--|------|
| 16-641. QUEUE_79_A Register Field Descriptions ..... | 2446 |
| 16-642. QUEUE_79_B Register Field Descriptions ..... | 2447 |
| 16-643. QUEUE_79_C Register Field Descriptions ..... | 2448 |
| 16-644. QUEUE_79_D Register Field Descriptions ..... | 2449 |
| 16-645. QUEUE_80_A Register Field Descriptions ..... | 2450 |
| 16-646. QUEUE_80_B Register Field Descriptions ..... | 2451 |
| 16-647. QUEUE_80_C Register Field Descriptions ..... | 2452 |
| 16-648. QUEUE_80_D Register Field Descriptions ..... | 2453 |
| 16-649. QUEUE_81_A Register Field Descriptions ..... | 2454 |
| 16-650. QUEUE_81_B Register Field Descriptions ..... | 2455 |
| 16-651. QUEUE_81_C Register Field Descriptions ..... | 2456 |
| 16-652. QUEUE_81_D Register Field Descriptions ..... | 2457 |
| 16-653. QUEUE_82_A Register Field Descriptions ..... | 2458 |
| 16-654. QUEUE_82_B Register Field Descriptions ..... | 2459 |
| 16-655. QUEUE_82_C Register Field Descriptions ..... | 2460 |
| 16-656. QUEUE_82_D Register Field Descriptions ..... | 2461 |
| 16-657. QUEUE_83_A Register Field Descriptions ..... | 2462 |
| 16-658. QUEUE_83_B Register Field Descriptions ..... | 2463 |
| 16-659. QUEUE_83_C Register Field Descriptions ..... | 2464 |
| 16-660. QUEUE_83_D Register Field Descriptions ..... | 2465 |
| 16-661. QUEUE_84_A Register Field Descriptions ..... | 2466 |
| 16-662. QUEUE_84_B Register Field Descriptions ..... | 2467 |
| 16-663. QUEUE_84_C Register Field Descriptions ..... | 2468 |
| 16-664. QUEUE_84_D Register Field Descriptions ..... | 2469 |
| 16-665. QUEUE_85_A Register Field Descriptions ..... | 2470 |
| 16-666. QUEUE_85_B Register Field Descriptions ..... | 2471 |
| 16-667. QUEUE_85_C Register Field Descriptions ..... | 2472 |
| 16-668. QUEUE_85_D Register Field Descriptions ..... | 2473 |
| 16-669. QUEUE_86_A Register Field Descriptions ..... | 2474 |
| 16-670. QUEUE_86_B Register Field Descriptions ..... | 2475 |
| 16-671. QUEUE_86_C Register Field Descriptions ..... | 2476 |
| 16-672. QUEUE_86_D Register Field Descriptions ..... | 2477 |
| 16-673. QUEUE_87_A Register Field Descriptions ..... | 2478 |
| 16-674. QUEUE_87_B Register Field Descriptions ..... | 2479 |
| 16-675. QUEUE_87_C Register Field Descriptions ..... | 2480 |
| 16-676. QUEUE_87_D Register Field Descriptions ..... | 2481 |
| 16-677. QUEUE_88_A Register Field Descriptions ..... | 2482 |
| 16-678. QUEUE_88_B Register Field Descriptions ..... | 2483 |
| 16-679. QUEUE_88_C Register Field Descriptions ..... | 2484 |
| 16-680. QUEUE_88_D Register Field Descriptions ..... | 2485 |
| 16-681. QUEUE_89_A Register Field Descriptions ..... | 2486 |
| 16-682. QUEUE_89_B Register Field Descriptions ..... | 2487 |
| 16-683. QUEUE_89_C Register Field Descriptions ..... | 2488 |
| 16-684. QUEUE_89_D Register Field Descriptions ..... | 2489 |
| 16-685. QUEUE_90_A Register Field Descriptions ..... | 2490 |
| 16-686. QUEUE_90_B Register Field Descriptions ..... | 2491 |
| 16-687. QUEUE_90_C Register Field Descriptions ..... | 2492 |
| 16-688. QUEUE_90_D Register Field Descriptions ..... | 2493 |
| 16-689. QUEUE_91_A Register Field Descriptions ..... | 2494 |

---

|   |      |
|---|------|
| 16-690. QUEUE_91_B Register Field Descriptions .....  | 2495 |
| 16-691. QUEUE_91_C Register Field Descriptions .....  | 2496 |
| 16-692. QUEUE_91_D Register Field Descriptions .....  | 2497 |
| 16-693. QUEUE_92_A Register Field Descriptions .....  | 2498 |
| 16-694. QUEUE_92_B Register Field Descriptions .....  | 2499 |
| 16-695. QUEUE_92_C Register Field Descriptions .....  | 2500 |
| 16-696. QUEUE_92_D Register Field Descriptions .....  | 2501 |
| 16-697. QUEUE_93_A Register Field Descriptions .....  | 2502 |
| 16-698. QUEUE_93_B Register Field Descriptions .....  | 2503 |
| 16-699. QUEUE_93_C Register Field Descriptions .....  | 2504 |
| 16-700. QUEUE_93_D Register Field Descriptions .....  | 2505 |
| 16-701. QUEUE_94_A Register Field Descriptions .....  | 2506 |
| 16-702. QUEUE_94_B Register Field Descriptions .....  | 2507 |
| 16-703. QUEUE_94_C Register Field Descriptions .....  | 2508 |
| 16-704. QUEUE_94_D Register Field Descriptions .....  | 2509 |
| 16-705. QUEUE_95_A Register Field Descriptions .....  | 2510 |
| 16-706. QUEUE_95_B Register Field Descriptions .....  | 2511 |
| 16-707. QUEUE_95_C Register Field Descriptions .....  | 2512 |
| 16-708. QUEUE_95_D Register Field Descriptions .....  | 2513 |
| 16-709. QUEUE_96_A Register Field Descriptions .....  | 2514 |
| 16-710. QUEUE_96_B Register Field Descriptions .....  | 2515 |
| 16-711. QUEUE_96_C Register Field Descriptions .....  | 2516 |
| 16-712. QUEUE_96_D Register Field Descriptions .....  | 2517 |
| 16-713. QUEUE_97_A Register Field Descriptions .....  | 2518 |
| 16-714. QUEUE_97_B Register Field Descriptions .....  | 2519 |
| 16-715. QUEUE_97_C Register Field Descriptions .....  | 2520 |
| 16-716. QUEUE_97_D Register Field Descriptions .....  | 2521 |
| 16-717. QUEUE_98_A Register Field Descriptions .....  | 2522 |
| 16-718. QUEUE_98_B Register Field Descriptions .....  | 2523 |
| 16-719. QUEUE_98_C Register Field Descriptions .....  | 2524 |
| 16-720. QUEUE_98_D Register Field Descriptions .....  | 2525 |
| 16-721. QUEUE_99_A Register Field Descriptions .....  | 2526 |
| 16-722. QUEUE_99_B Register Field Descriptions .....  | 2527 |
| 16-723. QUEUE_99_C Register Field Descriptions .....  | 2528 |
| 16-724. QUEUE_99_D Register Field Descriptions .....  | 2529 |
| 16-725. QUEUE_100_A Register Field Descriptions ..... | 2530 |
| 16-726. QUEUE_100_B Register Field Descriptions ..... | 2531 |
| 16-727. QUEUE_100_C Register Field Descriptions ..... | 2532 |
| 16-728. QUEUE_100_D Register Field Descriptions ..... | 2533 |
| 16-729. QUEUE_101_A Register Field Descriptions ..... | 2534 |
| 16-730. QUEUE_101_B Register Field Descriptions ..... | 2535 |
| 16-731. QUEUE_101_C Register Field Descriptions ..... | 2536 |
| 16-732. QUEUE_101_D Register Field Descriptions ..... | 2537 |
| 16-733. QUEUE_102_A Register Field Descriptions ..... | 2538 |
| 16-734. QUEUE_102_B Register Field Descriptions ..... | 2539 |
| 16-735. QUEUE_102_C Register Field Descriptions ..... | 2540 |
| 16-736. QUEUE_102_D Register Field Descriptions ..... | 2541 |
| 16-737. QUEUE_103_A Register Field Descriptions ..... | 2542 |
| 16-738. QUEUE_103_B Register Field Descriptions ..... | 2543 |

|   |      |
|---|------|
| 16-739. QUEUE_103_C Register Field Descriptions ..... | 2544 |
| 16-740. QUEUE_103_D Register Field Descriptions ..... | 2545 |
| 16-741. QUEUE_104_A Register Field Descriptions.....  | 2546 |
| 16-742. QUEUE_104_B Register Field Descriptions.....  | 2547 |
| 16-743. QUEUE_104_C Register Field Descriptions ..... | 2548 |
| 16-744. QUEUE_104_D Register Field Descriptions ..... | 2549 |
| 16-745. QUEUE_105_A Register Field Descriptions.....  | 2550 |
| 16-746. QUEUE_105_B Register Field Descriptions.....  | 2551 |
| 16-747. QUEUE_105_C Register Field Descriptions ..... | 2552 |
| 16-748. QUEUE_105_D Register Field Descriptions ..... | 2553 |
| 16-749. QUEUE_106_A Register Field Descriptions.....  | 2554 |
| 16-750. QUEUE_106_B Register Field Descriptions.....  | 2555 |
| 16-751. QUEUE_106_C Register Field Descriptions ..... | 2556 |
| 16-752. QUEUE_106_D Register Field Descriptions ..... | 2557 |
| 16-753. QUEUE_107_A Register Field Descriptions.....  | 2558 |
| 16-754. QUEUE_107_B Register Field Descriptions.....  | 2559 |
| 16-755. QUEUE_107_C Register Field Descriptions ..... | 2560 |
| 16-756. QUEUE_107_D Register Field Descriptions ..... | 2561 |
| 16-757. QUEUE_108_A Register Field Descriptions.....  | 2562 |
| 16-758. QUEUE_108_B Register Field Descriptions.....  | 2563 |
| 16-759. QUEUE_108_C Register Field Descriptions ..... | 2564 |
| 16-760. QUEUE_108_D Register Field Descriptions ..... | 2565 |
| 16-761. QUEUE_109_A Register Field Descriptions.....  | 2566 |
| 16-762. QUEUE_109_B Register Field Descriptions.....  | 2567 |
| 16-763. QUEUE_109_C Register Field Descriptions ..... | 2568 |
| 16-764. QUEUE_109_D Register Field Descriptions ..... | 2569 |
| 16-765. QUEUE_110_A Register Field Descriptions.....  | 2570 |
| 16-766. QUEUE_110_B Register Field Descriptions.....  | 2571 |
| 16-767. QUEUE_110_C Register Field Descriptions ..... | 2572 |
| 16-768. QUEUE_110_D Register Field Descriptions ..... | 2573 |
| 16-769. QUEUE_111_A Register Field Descriptions.....  | 2574 |
| 16-770. QUEUE_111_B Register Field Descriptions.....  | 2575 |
| 16-771. QUEUE_111_C Register Field Descriptions ..... | 2576 |
| 16-772. QUEUE_111_D Register Field Descriptions ..... | 2577 |
| 16-773. QUEUE_112_A Register Field Descriptions.....  | 2578 |
| 16-774. QUEUE_112_B Register Field Descriptions.....  | 2579 |
| 16-775. QUEUE_112_C Register Field Descriptions ..... | 2580 |
| 16-776. QUEUE_112_D Register Field Descriptions ..... | 2581 |
| 16-777. QUEUE_113_A Register Field Descriptions.....  | 2582 |
| 16-778. QUEUE_113_B Register Field Descriptions.....  | 2583 |
| 16-779. QUEUE_113_C Register Field Descriptions ..... | 2584 |
| 16-780. QUEUE_113_D Register Field Descriptions ..... | 2585 |
| 16-781. QUEUE_114_A Register Field Descriptions.....  | 2586 |
| 16-782. QUEUE_114_B Register Field Descriptions.....  | 2587 |
| 16-783. QUEUE_114_C Register Field Descriptions ..... | 2588 |
| 16-784. QUEUE_114_D Register Field Descriptions ..... | 2589 |
| 16-785. QUEUE_115_A Register Field Descriptions.....  | 2590 |
| 16-786. QUEUE_115_B Register Field Descriptions.....  | 2591 |
| 16-787. QUEUE_115_C Register Field Descriptions ..... | 2592 |

|   |      |
|---|------|
| 16-788. QUEUE_115_D Register Field Descriptions ..... | 2593 |
| 16-789. QUEUE_116_A Register Field Descriptions.....  | 2594 |
| 16-790. QUEUE_116_B Register Field Descriptions.....  | 2595 |
| 16-791. QUEUE_116_C Register Field Descriptions ..... | 2596 |
| 16-792. QUEUE_116_D Register Field Descriptions ..... | 2597 |
| 16-793. QUEUE_117_A Register Field Descriptions.....  | 2598 |
| 16-794. QUEUE_117_B Register Field Descriptions.....  | 2599 |
| 16-795. QUEUE_117_C Register Field Descriptions ..... | 2600 |
| 16-796. QUEUE_117_D Register Field Descriptions ..... | 2601 |
| 16-797. QUEUE_118_A Register Field Descriptions.....  | 2602 |
| 16-798. QUEUE_118_B Register Field Descriptions.....  | 2603 |
| 16-799. QUEUE_118_C Register Field Descriptions ..... | 2604 |
| 16-800. QUEUE_118_D Register Field Descriptions ..... | 2605 |
| 16-801. QUEUE_119_A Register Field Descriptions.....  | 2606 |
| 16-802. QUEUE_119_B Register Field Descriptions.....  | 2607 |
| 16-803. QUEUE_119_C Register Field Descriptions ..... | 2608 |
| 16-804. QUEUE_119_D Register Field Descriptions ..... | 2609 |
| 16-805. QUEUE_120_A Register Field Descriptions.....  | 2610 |
| 16-806. QUEUE_120_B Register Field Descriptions.....  | 2611 |
| 16-807. QUEUE_120_C Register Field Descriptions ..... | 2612 |
| 16-808. QUEUE_120_D Register Field Descriptions ..... | 2613 |
| 16-809. QUEUE_121_A Register Field Descriptions.....  | 2614 |
| 16-810. QUEUE_121_B Register Field Descriptions.....  | 2615 |
| 16-811. QUEUE_121_C Register Field Descriptions ..... | 2616 |
| 16-812. QUEUE_121_D Register Field Descriptions ..... | 2617 |
| 16-813. QUEUE_122_A Register Field Descriptions.....  | 2618 |
| 16-814. QUEUE_122_B Register Field Descriptions.....  | 2619 |
| 16-815. QUEUE_122_C Register Field Descriptions ..... | 2620 |
| 16-816. QUEUE_122_D Register Field Descriptions ..... | 2621 |
| 16-817. QUEUE_123_A Register Field Descriptions.....  | 2622 |
| 16-818. QUEUE_123_B Register Field Descriptions.....  | 2623 |
| 16-819. QUEUE_123_C Register Field Descriptions ..... | 2624 |
| 16-820. QUEUE_123_D Register Field Descriptions ..... | 2625 |
| 16-821. QUEUE_124_A Register Field Descriptions.....  | 2626 |
| 16-822. QUEUE_124_B Register Field Descriptions.....  | 2627 |
| 16-823. QUEUE_124_C Register Field Descriptions ..... | 2628 |
| 16-824. QUEUE_124_D Register Field Descriptions ..... | 2629 |
| 16-825. QUEUE_125_A Register Field Descriptions.....  | 2630 |
| 16-826. QUEUE_125_B Register Field Descriptions.....  | 2631 |
| 16-827. QUEUE_125_C Register Field Descriptions ..... | 2632 |
| 16-828. QUEUE_125_D Register Field Descriptions ..... | 2633 |
| 16-829. QUEUE_126_A Register Field Descriptions.....  | 2634 |
| 16-830. QUEUE_126_B Register Field Descriptions.....  | 2635 |
| 16-831. QUEUE_126_C Register Field Descriptions ..... | 2636 |
| 16-832. QUEUE_126_D Register Field Descriptions ..... | 2637 |
| 16-833. QUEUE_127_A Register Field Descriptions.....  | 2638 |
| 16-834. QUEUE_127_B Register Field Descriptions.....  | 2639 |
| 16-835. QUEUE_127_C Register Field Descriptions ..... | 2640 |
| 16-836. QUEUE_127_D Register Field Descriptions ..... | 2641 |

|   |      |
|---|------|
| 16-837. QUEUE_128_A Register Field Descriptions.....  | 2642 |
| 16-838. QUEUE_128_B Register Field Descriptions.....  | 2643 |
| 16-839. QUEUE_128_C Register Field Descriptions ..... | 2644 |
| 16-840. QUEUE_128_D Register Field Descriptions ..... | 2645 |
| 16-841. QUEUE_129_A Register Field Descriptions.....  | 2646 |
| 16-842. QUEUE_129_B Register Field Descriptions.....  | 2647 |
| 16-843. QUEUE_129_C Register Field Descriptions ..... | 2648 |
| 16-844. QUEUE_129_D Register Field Descriptions ..... | 2649 |
| 16-845. QUEUE_130_A Register Field Descriptions.....  | 2650 |
| 16-846. QUEUE_130_B Register Field Descriptions.....  | 2651 |
| 16-847. QUEUE_130_C Register Field Descriptions ..... | 2652 |
| 16-848. QUEUE_130_D Register Field Descriptions ..... | 2653 |
| 16-849. QUEUE_131_A Register Field Descriptions.....  | 2654 |
| 16-850. QUEUE_131_B Register Field Descriptions.....  | 2655 |
| 16-851. QUEUE_131_C Register Field Descriptions ..... | 2656 |
| 16-852. QUEUE_131_D Register Field Descriptions ..... | 2657 |
| 16-853. QUEUE_132_A Register Field Descriptions.....  | 2658 |
| 16-854. QUEUE_132_B Register Field Descriptions.....  | 2659 |
| 16-855. QUEUE_132_C Register Field Descriptions ..... | 2660 |
| 16-856. QUEUE_132_D Register Field Descriptions ..... | 2661 |
| 16-857. QUEUE_133_A Register Field Descriptions.....  | 2662 |
| 16-858. QUEUE_133_B Register Field Descriptions.....  | 2663 |
| 16-859. QUEUE_133_C Register Field Descriptions ..... | 2664 |
| 16-860. QUEUE_133_D Register Field Descriptions ..... | 2665 |
| 16-861. QUEUE_134_A Register Field Descriptions.....  | 2666 |
| 16-862. QUEUE_134_B Register Field Descriptions.....  | 2667 |
| 16-863. QUEUE_134_C Register Field Descriptions ..... | 2668 |
| 16-864. QUEUE_134_D Register Field Descriptions ..... | 2669 |
| 16-865. QUEUE_135_A Register Field Descriptions.....  | 2670 |
| 16-866. QUEUE_135_B Register Field Descriptions.....  | 2671 |
| 16-867. QUEUE_135_C Register Field Descriptions ..... | 2672 |
| 16-868. QUEUE_135_D Register Field Descriptions ..... | 2673 |
| 16-869. QUEUE_136_A Register Field Descriptions.....  | 2674 |
| 16-870. QUEUE_136_B Register Field Descriptions.....  | 2675 |
| 16-871. QUEUE_136_C Register Field Descriptions ..... | 2676 |
| 16-872. QUEUE_136_D Register Field Descriptions ..... | 2677 |
| 16-873. QUEUE_137_A Register Field Descriptions.....  | 2678 |
| 16-874. QUEUE_137_B Register Field Descriptions.....  | 2679 |
| 16-875. QUEUE_137_C Register Field Descriptions ..... | 2680 |
| 16-876. QUEUE_137_D Register Field Descriptions ..... | 2681 |
| 16-877. QUEUE_138_A Register Field Descriptions.....  | 2682 |
| 16-878. QUEUE_138_B Register Field Descriptions.....  | 2683 |
| 16-879. QUEUE_138_C Register Field Descriptions ..... | 2684 |
| 16-880. QUEUE_138_D Register Field Descriptions ..... | 2685 |
| 16-881. QUEUE_139_A Register Field Descriptions.....  | 2686 |
| 16-882. QUEUE_139_B Register Field Descriptions.....  | 2687 |
| 16-883. QUEUE_139_C Register Field Descriptions ..... | 2688 |
| 16-884. QUEUE_139_D Register Field Descriptions ..... | 2689 |
| 16-885. QUEUE_140_A Register Field Descriptions.....  | 2690 |

---

|   |      |
|---|------|
| 16-886. QUEUE_140_B Register Field Descriptions.....  | 2691 |
| 16-887. QUEUE_140_C Register Field Descriptions ..... | 2692 |
| 16-888. QUEUE_140_D Register Field Descriptions ..... | 2693 |
| 16-889. QUEUE_141_A Register Field Descriptions.....  | 2694 |
| 16-890. QUEUE_141_B Register Field Descriptions.....  | 2695 |
| 16-891. QUEUE_141_C Register Field Descriptions ..... | 2696 |
| 16-892. QUEUE_141_D Register Field Descriptions ..... | 2697 |
| 16-893. QUEUE_142_A Register Field Descriptions.....  | 2698 |
| 16-894. QUEUE_142_B Register Field Descriptions.....  | 2699 |
| 16-895. QUEUE_142_C Register Field Descriptions ..... | 2700 |
| 16-896. QUEUE_142_D Register Field Descriptions ..... | 2701 |
| 16-897. QUEUE_143_A Register Field Descriptions.....  | 2702 |
| 16-898. QUEUE_143_B Register Field Descriptions.....  | 2703 |
| 16-899. QUEUE_143_C Register Field Descriptions ..... | 2704 |
| 16-900. QUEUE_143_D Register Field Descriptions ..... | 2705 |
| 16-901. QUEUE_144_A Register Field Descriptions.....  | 2706 |
| 16-902. QUEUE_144_B Register Field Descriptions.....  | 2707 |
| 16-903. QUEUE_144_C Register Field Descriptions ..... | 2708 |
| 16-904. QUEUE_144_D Register Field Descriptions ..... | 2709 |
| 16-905. QUEUE_145_A Register Field Descriptions.....  | 2710 |
| 16-906. QUEUE_145_B Register Field Descriptions ..... | 2711 |
| 16-907. QUEUE_145_C Register Field Descriptions ..... | 2712 |
| 16-908. QUEUE_145_D Register Field Descriptions ..... | 2713 |
| 16-909. QUEUE_146_A Register Field Descriptions.....  | 2714 |
| 16-910. QUEUE_146_B Register Field Descriptions ..... | 2715 |
| 16-911. QUEUE_146_C Register Field Descriptions ..... | 2716 |
| 16-912. QUEUE_146_D Register Field Descriptions ..... | 2717 |
| 16-913. QUEUE_147_A Register Field Descriptions.....  | 2718 |
| 16-914. QUEUE_147_B Register Field Descriptions ..... | 2719 |
| 16-915. QUEUE_147_C Register Field Descriptions ..... | 2720 |
| 16-916. QUEUE_147_D Register Field Descriptions ..... | 2721 |
| 16-917. QUEUE_148_A Register Field Descriptions.....  | 2722 |
| 16-918. QUEUE_148_B Register Field Descriptions ..... | 2723 |
| 16-919. QUEUE_148_C Register Field Descriptions ..... | 2724 |
| 16-920. QUEUE_148_D Register Field Descriptions ..... | 2725 |
| 16-921. QUEUE_149_A Register Field Descriptions.....  | 2726 |
| 16-922. QUEUE_149_B Register Field Descriptions ..... | 2727 |
| 16-923. QUEUE_149_C Register Field Descriptions ..... | 2728 |
| 16-924. QUEUE_149_D Register Field Descriptions ..... | 2729 |
| 16-925. QUEUE_150_A Register Field Descriptions.....  | 2730 |
| 16-926. QUEUE_150_B Register Field Descriptions ..... | 2731 |
| 16-927. QUEUE_150_C Register Field Descriptions ..... | 2732 |
| 16-928. QUEUE_150_D Register Field Descriptions ..... | 2733 |
| 16-929. QUEUE_151_A Register Field Descriptions.....  | 2734 |
| 16-930. QUEUE_151_B Register Field Descriptions ..... | 2735 |
| 16-931. QUEUE_151_C Register Field Descriptions ..... | 2736 |
| 16-932. QUEUE_151_D Register Field Descriptions ..... | 2737 |
| 16-933. QUEUE_152_A Register Field Descriptions.....  | 2738 |
| 16-934. QUEUE_152_B Register Field Descriptions ..... | 2739 |

|   |      |
|---|------|
| 16-935. QUEUE_152_C Register Field Descriptions .....       | 2740 |
| 16-936. QUEUE_152_D Register Field Descriptions .....       | 2741 |
| 16-937. QUEUE_153_A Register Field Descriptions.....        | 2742 |
| 16-938. QUEUE_153_B Register Field Descriptions.....        | 2743 |
| 16-939. QUEUE_153_C Register Field Descriptions .....       | 2744 |
| 16-940. QUEUE_153_D Register Field Descriptions .....       | 2745 |
| 16-941. QUEUE_154_A Register Field Descriptions.....        | 2746 |
| 16-942. QUEUE_154_B Register Field Descriptions.....        | 2747 |
| 16-943. QUEUE_154_C Register Field Descriptions .....       | 2748 |
| 16-944. QUEUE_154_D Register Field Descriptions .....       | 2749 |
| 16-945. QUEUE_155_A Register Field Descriptions.....        | 2750 |
| 16-946. QUEUE_155_B Register Field Descriptions.....        | 2751 |
| 16-947. QUEUE_155_C Register Field Descriptions .....       | 2752 |
| 16-948. QUEUE_155_D Register Field Descriptions .....       | 2753 |
| 16-949. QUEUE_0_STATUS_A Register Field Descriptions .....  | 2754 |
| 16-950. QUEUE_0_STATUS_B Register Field Descriptions .....  | 2755 |
| 16-951. QUEUE_0_STATUS_C Register Field Descriptions .....  | 2756 |
| 16-952. QUEUE_1_STATUS_A Register Field Descriptions .....  | 2757 |
| 16-953. QUEUE_1_STATUS_B Register Field Descriptions .....  | 2758 |
| 16-954. QUEUE_1_STATUS_C Register Field Descriptions .....  | 2759 |
| 16-955. QUEUE_2_STATUS_A Register Field Descriptions .....  | 2760 |
| 16-956. QUEUE_2_STATUS_B Register Field Descriptions .....  | 2761 |
| 16-957. QUEUE_2_STATUS_C Register Field Descriptions .....  | 2762 |
| 16-958. QUEUE_3_STATUS_A Register Field Descriptions .....  | 2763 |
| 16-959. QUEUE_3_STATUS_B Register Field Descriptions .....  | 2764 |
| 16-960. QUEUE_3_STATUS_C Register Field Descriptions .....  | 2765 |
| 16-961. QUEUE_4_STATUS_A Register Field Descriptions .....  | 2766 |
| 16-962. QUEUE_4_STATUS_B Register Field Descriptions .....  | 2767 |
| 16-963. QUEUE_4_STATUS_C Register Field Descriptions .....  | 2768 |
| 16-964. QUEUE_5_STATUS_A Register Field Descriptions .....  | 2769 |
| 16-965. QUEUE_5_STATUS_B Register Field Descriptions .....  | 2770 |
| 16-966. QUEUE_5_STATUS_C Register Field Descriptions .....  | 2771 |
| 16-967. QUEUE_6_STATUS_A Register Field Descriptions .....  | 2772 |
| 16-968. QUEUE_6_STATUS_B Register Field Descriptions .....  | 2773 |
| 16-969. QUEUE_6_STATUS_C Register Field Descriptions .....  | 2774 |
| 16-970. QUEUE_7_STATUS_A Register Field Descriptions .....  | 2775 |
| 16-971. QUEUE_7_STATUS_B Register Field Descriptions .....  | 2776 |
| 16-972. QUEUE_7_STATUS_C Register Field Descriptions .....  | 2777 |
| 16-973. QUEUE_8_STATUS_A Register Field Descriptions .....  | 2778 |
| 16-974. QUEUE_8_STATUS_B Register Field Descriptions .....  | 2779 |
| 16-975. QUEUE_8_STATUS_C Register Field Descriptions .....  | 2780 |
| 16-976. QUEUE_9_STATUS_A Register Field Descriptions .....  | 2781 |
| 16-977. QUEUE_9_STATUS_B Register Field Descriptions .....  | 2782 |
| 16-978. QUEUE_9_STATUS_C Register Field Descriptions .....  | 2783 |
| 16-979. QUEUE_10_STATUS_A Register Field Descriptions ..... | 2784 |
| 16-980. QUEUE_10_STATUS_B Register Field Descriptions ..... | 2785 |
| 16-981. QUEUE_10_STATUS_C Register Field Descriptions.....  | 2786 |
| 16-982. QUEUE_11_STATUS_A Register Field Descriptions ..... | 2787 |
| 16-983. QUEUE_11_STATUS_B Register Field Descriptions ..... | 2788 |

|  |      |
|--|------|
| 16-984. QUEUE_11_STATUS_C Register Field Descriptions .....  | 2789 |
| 16-985. QUEUE_12_STATUS_A Register Field Descriptions .....  | 2790 |
| 16-986. QUEUE_12_STATUS_B Register Field Descriptions .....  | 2791 |
| 16-987. QUEUE_12_STATUS_C Register Field Descriptions .....  | 2792 |
| 16-988. QUEUE_13_STATUS_A Register Field Descriptions .....  | 2793 |
| 16-989. QUEUE_13_STATUS_B Register Field Descriptions .....  | 2794 |
| 16-990. QUEUE_13_STATUS_C Register Field Descriptions .....  | 2795 |
| 16-991. QUEUE_14_STATUS_A Register Field Descriptions .....  | 2796 |
| 16-992. QUEUE_14_STATUS_B Register Field Descriptions .....  | 2797 |
| 16-993. QUEUE_14_STATUS_C Register Field Descriptions .....  | 2798 |
| 16-994. QUEUE_15_STATUS_A Register Field Descriptions .....  | 2799 |
| 16-995. QUEUE_15_STATUS_B Register Field Descriptions .....  | 2800 |
| 16-996. QUEUE_15_STATUS_C Register Field Descriptions .....  | 2801 |
| 16-997. QUEUE_16_STATUS_A Register Field Descriptions .....  | 2802 |
| 16-998. QUEUE_16_STATUS_B Register Field Descriptions .....  | 2803 |
| 16-999. QUEUE_16_STATUS_C Register Field Descriptions .....  | 2804 |
| 16-1000. QUEUE_17_STATUS_A Register Field Descriptions ..... | 2805 |
| 16-1001. QUEUE_17_STATUS_B Register Field Descriptions ..... | 2806 |
| 16-1002. QUEUE_17_STATUS_C Register Field Descriptions ..... | 2807 |
| 16-1003. QUEUE_18_STATUS_A Register Field Descriptions ..... | 2808 |
| 16-1004. QUEUE_18_STATUS_B Register Field Descriptions ..... | 2809 |
| 16-1005. QUEUE_18_STATUS_C Register Field Descriptions ..... | 2810 |
| 16-1006. QUEUE_19_STATUS_A Register Field Descriptions ..... | 2811 |
| 16-1007. QUEUE_19_STATUS_B Register Field Descriptions ..... | 2812 |
| 16-1008. QUEUE_19_STATUS_C Register Field Descriptions ..... | 2813 |
| 16-1009. QUEUE_20_STATUS_A Register Field Descriptions ..... | 2814 |
| 16-1010. QUEUE_20_STATUS_B Register Field Descriptions ..... | 2815 |
| 16-1011. QUEUE_20_STATUS_C Register Field Descriptions ..... | 2816 |
| 16-1012. QUEUE_21_STATUS_A Register Field Descriptions ..... | 2817 |
| 16-1013. QUEUE_21_STATUS_B Register Field Descriptions ..... | 2818 |
| 16-1014. QUEUE_21_STATUS_C Register Field Descriptions ..... | 2819 |
| 16-1015. QUEUE_22_STATUS_A Register Field Descriptions ..... | 2820 |
| 16-1016. QUEUE_22_STATUS_B Register Field Descriptions ..... | 2821 |
| 16-1017. QUEUE_22_STATUS_C Register Field Descriptions ..... | 2822 |
| 16-1018. QUEUE_23_STATUS_A Register Field Descriptions ..... | 2823 |
| 16-1019. QUEUE_23_STATUS_B Register Field Descriptions ..... | 2824 |
| 16-1020. QUEUE_23_STATUS_C Register Field Descriptions ..... | 2825 |
| 16-1021. QUEUE_24_STATUS_A Register Field Descriptions ..... | 2826 |
| 16-1022. QUEUE_24_STATUS_B Register Field Descriptions ..... | 2827 |
| 16-1023. QUEUE_24_STATUS_C Register Field Descriptions ..... | 2828 |
| 16-1024. QUEUE_25_STATUS_A Register Field Descriptions ..... | 2829 |
| 16-1025. QUEUE_25_STATUS_B Register Field Descriptions ..... | 2830 |
| 16-1026. QUEUE_25_STATUS_C Register Field Descriptions ..... | 2831 |
| 16-1027. QUEUE_26_STATUS_A Register Field Descriptions ..... | 2832 |
| 16-1028. QUEUE_26_STATUS_B Register Field Descriptions ..... | 2833 |
| 16-1029. QUEUE_26_STATUS_C Register Field Descriptions ..... | 2834 |
| 16-1030. QUEUE_27_STATUS_A Register Field Descriptions ..... | 2835 |
| 16-1031. QUEUE_27_STATUS_B Register Field Descriptions ..... | 2836 |
| 16-1032. QUEUE_27_STATUS_C Register Field Descriptions ..... | 2837 |

|  |      |
|--|------|
| 16-1033. QUEUE_28_STATUS_A Register Field Descriptions ..... | 2838 |
| 16-1034. QUEUE_28_STATUS_B Register Field Descriptions ..... | 2839 |
| 16-1035. QUEUE_28_STATUS_C Register Field Descriptions ..... | 2840 |
| 16-1036. QUEUE_29_STATUS_A Register Field Descriptions ..... | 2841 |
| 16-1037. QUEUE_29_STATUS_B Register Field Descriptions ..... | 2842 |
| 16-1038. QUEUE_29_STATUS_C Register Field Descriptions ..... | 2843 |
| 16-1039. QUEUE_30_STATUS_A Register Field Descriptions ..... | 2844 |
| 16-1040. QUEUE_30_STATUS_B Register Field Descriptions ..... | 2845 |
| 16-1041. QUEUE_30_STATUS_C Register Field Descriptions ..... | 2846 |
| 16-1042. QUEUE_31_STATUS_A Register Field Descriptions ..... | 2847 |
| 16-1043. QUEUE_31_STATUS_B Register Field Descriptions ..... | 2848 |
| 16-1044. QUEUE_31_STATUS_C Register Field Descriptions ..... | 2849 |
| 16-1045. QUEUE_32_STATUS_A Register Field Descriptions ..... | 2850 |
| 16-1046. QUEUE_32_STATUS_B Register Field Descriptions ..... | 2851 |
| 16-1047. QUEUE_32_STATUS_C Register Field Descriptions ..... | 2852 |
| 16-1048. QUEUE_33_STATUS_A Register Field Descriptions ..... | 2853 |
| 16-1049. QUEUE_33_STATUS_B Register Field Descriptions ..... | 2854 |
| 16-1050. QUEUE_33_STATUS_C Register Field Descriptions ..... | 2855 |
| 16-1051. QUEUE_34_STATUS_A Register Field Descriptions ..... | 2856 |
| 16-1052. QUEUE_34_STATUS_B Register Field Descriptions ..... | 2857 |
| 16-1053. QUEUE_34_STATUS_C Register Field Descriptions ..... | 2858 |
| 16-1054. QUEUE_35_STATUS_A Register Field Descriptions ..... | 2859 |
| 16-1055. QUEUE_35_STATUS_B Register Field Descriptions ..... | 2860 |
| 16-1056. QUEUE_35_STATUS_C Register Field Descriptions ..... | 2861 |
| 16-1057. QUEUE_36_STATUS_A Register Field Descriptions ..... | 2862 |
| 16-1058. QUEUE_36_STATUS_B Register Field Descriptions ..... | 2863 |
| 16-1059. QUEUE_36_STATUS_C Register Field Descriptions ..... | 2864 |
| 16-1060. QUEUE_37_STATUS_A Register Field Descriptions ..... | 2865 |
| 16-1061. QUEUE_37_STATUS_B Register Field Descriptions ..... | 2866 |
| 16-1062. QUEUE_37_STATUS_C Register Field Descriptions ..... | 2867 |
| 16-1063. QUEUE_38_STATUS_A Register Field Descriptions ..... | 2868 |
| 16-1064. QUEUE_38_STATUS_B Register Field Descriptions ..... | 2869 |
| 16-1065. QUEUE_38_STATUS_C Register Field Descriptions ..... | 2870 |
| 16-1066. QUEUE_39_STATUS_A Register Field Descriptions ..... | 2871 |
| 16-1067. QUEUE_39_STATUS_B Register Field Descriptions ..... | 2872 |
| 16-1068. QUEUE_39_STATUS_C Register Field Descriptions ..... | 2873 |
| 16-1069. QUEUE_40_STATUS_A Register Field Descriptions ..... | 2874 |
| 16-1070. QUEUE_40_STATUS_B Register Field Descriptions ..... | 2875 |
| 16-1071. QUEUE_40_STATUS_C Register Field Descriptions ..... | 2876 |
| 16-1072. QUEUE_41_STATUS_A Register Field Descriptions ..... | 2877 |
| 16-1073. QUEUE_41_STATUS_B Register Field Descriptions ..... | 2878 |
| 16-1074. QUEUE_41_STATUS_C Register Field Descriptions ..... | 2879 |
| 16-1075. QUEUE_42_STATUS_A Register Field Descriptions ..... | 2880 |
| 16-1076. QUEUE_42_STATUS_B Register Field Descriptions ..... | 2881 |
| 16-1077. QUEUE_42_STATUS_C Register Field Descriptions ..... | 2882 |
| 16-1078. QUEUE_43_STATUS_A Register Field Descriptions ..... | 2883 |
| 16-1079. QUEUE_43_STATUS_B Register Field Descriptions ..... | 2884 |
| 16-1080. QUEUE_43_STATUS_C Register Field Descriptions ..... | 2885 |
| 16-1081. QUEUE_44_STATUS_A Register Field Descriptions ..... | 2886 |

---

|  |      |
|--|------|
| 16-1082. QUEUE_44_STATUS_B Register Field Descriptions ..... | 2887 |
| 16-1083. QUEUE_44_STATUS_C Register Field Descriptions ..... | 2888 |
| 16-1084. QUEUE_45_STATUS_A Register Field Descriptions ..... | 2889 |
| 16-1085. QUEUE_45_STATUS_B Register Field Descriptions ..... | 2890 |
| 16-1086. QUEUE_45_STATUS_C Register Field Descriptions ..... | 2891 |
| 16-1087. QUEUE_46_STATUS_A Register Field Descriptions ..... | 2892 |
| 16-1088. QUEUE_46_STATUS_B Register Field Descriptions ..... | 2893 |
| 16-1089. QUEUE_46_STATUS_C Register Field Descriptions ..... | 2894 |
| 16-1090. QUEUE_47_STATUS_A Register Field Descriptions ..... | 2895 |
| 16-1091. QUEUE_47_STATUS_B Register Field Descriptions ..... | 2896 |
| 16-1092. QUEUE_47_STATUS_C Register Field Descriptions ..... | 2897 |
| 16-1093. QUEUE_48_STATUS_A Register Field Descriptions ..... | 2898 |
| 16-1094. QUEUE_48_STATUS_B Register Field Descriptions ..... | 2899 |
| 16-1095. QUEUE_48_STATUS_C Register Field Descriptions ..... | 2900 |
| 16-1096. QUEUE_49_STATUS_A Register Field Descriptions ..... | 2901 |
| 16-1097. QUEUE_49_STATUS_B Register Field Descriptions ..... | 2902 |
| 16-1098. QUEUE_49_STATUS_C Register Field Descriptions ..... | 2903 |
| 16-1099. QUEUE_50_STATUS_A Register Field Descriptions ..... | 2904 |
| 16-1100. QUEUE_50_STATUS_B Register Field Descriptions ..... | 2905 |
| 16-1101. QUEUE_50_STATUS_C Register Field Descriptions ..... | 2906 |
| 16-1102. QUEUE_51_STATUS_A Register Field Descriptions ..... | 2907 |
| 16-1103. QUEUE_51_STATUS_B Register Field Descriptions ..... | 2908 |
| 16-1104. QUEUE_51_STATUS_C Register Field Descriptions ..... | 2909 |
| 16-1105. QUEUE_52_STATUS_A Register Field Descriptions ..... | 2910 |
| 16-1106. QUEUE_52_STATUS_B Register Field Descriptions ..... | 2911 |
| 16-1107. QUEUE_52_STATUS_C Register Field Descriptions ..... | 2912 |
| 16-1108. QUEUE_53_STATUS_A Register Field Descriptions ..... | 2913 |
| 16-1109. QUEUE_53_STATUS_B Register Field Descriptions ..... | 2914 |
| 16-1110. QUEUE_53_STATUS_C Register Field Descriptions ..... | 2915 |
| 16-1111. QUEUE_54_STATUS_A Register Field Descriptions ..... | 2916 |
| 16-1112. QUEUE_54_STATUS_B Register Field Descriptions ..... | 2917 |
| 16-1113. QUEUE_54_STATUS_C Register Field Descriptions ..... | 2918 |
| 16-1114. QUEUE_55_STATUS_A Register Field Descriptions ..... | 2919 |
| 16-1115. QUEUE_55_STATUS_B Register Field Descriptions ..... | 2920 |
| 16-1116. QUEUE_55_STATUS_C Register Field Descriptions ..... | 2921 |
| 16-1117. QUEUE_56_STATUS_A Register Field Descriptions ..... | 2922 |
| 16-1118. QUEUE_56_STATUS_B Register Field Descriptions ..... | 2923 |
| 16-1119. QUEUE_56_STATUS_C Register Field Descriptions ..... | 2924 |
| 16-1120. QUEUE_57_STATUS_A Register Field Descriptions ..... | 2925 |
| 16-1121. QUEUE_57_STATUS_B Register Field Descriptions ..... | 2926 |
| 16-1122. QUEUE_57_STATUS_C Register Field Descriptions ..... | 2927 |
| 16-1123. QUEUE_58_STATUS_A Register Field Descriptions ..... | 2928 |
| 16-1124. QUEUE_58_STATUS_B Register Field Descriptions ..... | 2929 |
| 16-1125. QUEUE_58_STATUS_C Register Field Descriptions ..... | 2930 |
| 16-1126. QUEUE_59_STATUS_A Register Field Descriptions ..... | 2931 |
| 16-1127. QUEUE_59_STATUS_B Register Field Descriptions ..... | 2932 |
| 16-1128. QUEUE_59_STATUS_C Register Field Descriptions ..... | 2933 |
| 16-1129. QUEUE_60_STATUS_A Register Field Descriptions ..... | 2934 |
| 16-1130. QUEUE_60_STATUS_B Register Field Descriptions ..... | 2935 |

|  |      |
|--|------|
| 16-1131. QUEUE_60_STATUS_C Register Field Descriptions ..... | 2936 |
| 16-1132. QUEUE_61_STATUS_A Register Field Descriptions ..... | 2937 |
| 16-1133. QUEUE_61_STATUS_B Register Field Descriptions ..... | 2938 |
| 16-1134. QUEUE_61_STATUS_C Register Field Descriptions ..... | 2939 |
| 16-1135. QUEUE_62_STATUS_A Register Field Descriptions ..... | 2940 |
| 16-1136. QUEUE_62_STATUS_B Register Field Descriptions ..... | 2941 |
| 16-1137. QUEUE_62_STATUS_C Register Field Descriptions ..... | 2942 |
| 16-1138. QUEUE_63_STATUS_A Register Field Descriptions ..... | 2943 |
| 16-1139. QUEUE_63_STATUS_B Register Field Descriptions ..... | 2944 |
| 16-1140. QUEUE_63_STATUS_C Register Field Descriptions ..... | 2945 |
| 16-1141. QUEUE_64_STATUS_A Register Field Descriptions ..... | 2946 |
| 16-1142. QUEUE_64_STATUS_B Register Field Descriptions ..... | 2947 |
| 16-1143. QUEUE_64_STATUS_C Register Field Descriptions ..... | 2948 |
| 16-1144. QUEUE_65_STATUS_A Register Field Descriptions ..... | 2949 |
| 16-1145. QUEUE_65_STATUS_B Register Field Descriptions ..... | 2950 |
| 16-1146. QUEUE_65_STATUS_C Register Field Descriptions ..... | 2951 |
| 16-1147. QUEUE_66_STATUS_A Register Field Descriptions ..... | 2952 |
| 16-1148. QUEUE_66_STATUS_B Register Field Descriptions ..... | 2953 |
| 16-1149. QUEUE_66_STATUS_C Register Field Descriptions ..... | 2954 |
| 16-1150. QUEUE_67_STATUS_A Register Field Descriptions ..... | 2955 |
| 16-1151. QUEUE_67_STATUS_B Register Field Descriptions ..... | 2956 |
| 16-1152. QUEUE_67_STATUS_C Register Field Descriptions ..... | 2957 |
| 16-1153. QUEUE_68_STATUS_A Register Field Descriptions ..... | 2958 |
| 16-1154. QUEUE_68_STATUS_B Register Field Descriptions ..... | 2959 |
| 16-1155. QUEUE_68_STATUS_C Register Field Descriptions ..... | 2960 |
| 16-1156. QUEUE_69_STATUS_A Register Field Descriptions ..... | 2961 |
| 16-1157. QUEUE_69_STATUS_B Register Field Descriptions ..... | 2962 |
| 16-1158. QUEUE_69_STATUS_C Register Field Descriptions ..... | 2963 |
| 16-1159. QUEUE_70_STATUS_A Register Field Descriptions ..... | 2964 |
| 16-1160. QUEUE_70_STATUS_B Register Field Descriptions ..... | 2965 |
| 16-1161. QUEUE_70_STATUS_C Register Field Descriptions ..... | 2966 |
| 16-1162. QUEUE_71_STATUS_A Register Field Descriptions ..... | 2967 |
| 16-1163. QUEUE_71_STATUS_B Register Field Descriptions ..... | 2968 |
| 16-1164. QUEUE_71_STATUS_C Register Field Descriptions ..... | 2969 |
| 16-1165. QUEUE_72_STATUS_A Register Field Descriptions ..... | 2970 |
| 16-1166. QUEUE_72_STATUS_B Register Field Descriptions ..... | 2971 |
| 16-1167. QUEUE_72_STATUS_C Register Field Descriptions ..... | 2972 |
| 16-1168. QUEUE_73_STATUS_A Register Field Descriptions ..... | 2973 |
| 16-1169. QUEUE_73_STATUS_B Register Field Descriptions ..... | 2974 |
| 16-1170. QUEUE_73_STATUS_C Register Field Descriptions ..... | 2975 |
| 16-1171. QUEUE_74_STATUS_A Register Field Descriptions ..... | 2976 |
| 16-1172. QUEUE_74_STATUS_B Register Field Descriptions ..... | 2977 |
| 16-1173. QUEUE_74_STATUS_C Register Field Descriptions ..... | 2978 |
| 16-1174. QUEUE_75_STATUS_A Register Field Descriptions ..... | 2979 |
| 16-1175. QUEUE_75_STATUS_B Register Field Descriptions ..... | 2980 |
| 16-1176. QUEUE_75_STATUS_C Register Field Descriptions ..... | 2981 |
| 16-1177. QUEUE_76_STATUS_A Register Field Descriptions ..... | 2982 |
| 16-1178. QUEUE_76_STATUS_B Register Field Descriptions ..... | 2983 |
| 16-1179. QUEUE_76_STATUS_C Register Field Descriptions ..... | 2984 |

|  |      |
|--|------|
| 16-1180. QUEUE_77_STATUS_A Register Field Descriptions ..... | 2985 |
| 16-1181. QUEUE_77_STATUS_B Register Field Descriptions ..... | 2986 |
| 16-1182. QUEUE_77_STATUS_C Register Field Descriptions ..... | 2987 |
| 16-1183. QUEUE_78_STATUS_A Register Field Descriptions ..... | 2988 |
| 16-1184. QUEUE_78_STATUS_B Register Field Descriptions ..... | 2989 |
| 16-1185. QUEUE_78_STATUS_C Register Field Descriptions ..... | 2990 |
| 16-1186. QUEUE_79_STATUS_A Register Field Descriptions ..... | 2991 |
| 16-1187. QUEUE_79_STATUS_B Register Field Descriptions ..... | 2992 |
| 16-1188. QUEUE_79_STATUS_C Register Field Descriptions ..... | 2993 |
| 16-1189. QUEUE_80_STATUS_A Register Field Descriptions ..... | 2994 |
| 16-1190. QUEUE_80_STATUS_B Register Field Descriptions ..... | 2995 |
| 16-1191. QUEUE_80_STATUS_C Register Field Descriptions ..... | 2996 |
| 16-1192. QUEUE_81_STATUS_A Register Field Descriptions ..... | 2997 |
| 16-1193. QUEUE_81_STATUS_B Register Field Descriptions ..... | 2998 |
| 16-1194. QUEUE_81_STATUS_C Register Field Descriptions ..... | 2999 |
| 16-1195. QUEUE_82_STATUS_A Register Field Descriptions ..... | 3000 |
| 16-1196. QUEUE_82_STATUS_B Register Field Descriptions ..... | 3001 |
| 16-1197. QUEUE_82_STATUS_C Register Field Descriptions ..... | 3002 |
| 16-1198. QUEUE_83_STATUS_A Register Field Descriptions ..... | 3003 |
| 16-1199. QUEUE_83_STATUS_B Register Field Descriptions ..... | 3004 |
| 16-1200. QUEUE_83_STATUS_C Register Field Descriptions ..... | 3005 |
| 16-1201. QUEUE_84_STATUS_A Register Field Descriptions ..... | 3006 |
| 16-1202. QUEUE_84_STATUS_B Register Field Descriptions ..... | 3007 |
| 16-1203. QUEUE_84_STATUS_C Register Field Descriptions ..... | 3008 |
| 16-1204. QUEUE_85_STATUS_A Register Field Descriptions ..... | 3009 |
| 16-1205. QUEUE_85_STATUS_B Register Field Descriptions ..... | 3010 |
| 16-1206. QUEUE_85_STATUS_C Register Field Descriptions ..... | 3011 |
| 16-1207. QUEUE_86_STATUS_A Register Field Descriptions ..... | 3012 |
| 16-1208. QUEUE_86_STATUS_B Register Field Descriptions ..... | 3013 |
| 16-1209. QUEUE_86_STATUS_C Register Field Descriptions ..... | 3014 |
| 16-1210. QUEUE_87_STATUS_A Register Field Descriptions ..... | 3015 |
| 16-1211. QUEUE_87_STATUS_B Register Field Descriptions ..... | 3016 |
| 16-1212. QUEUE_87_STATUS_C Register Field Descriptions ..... | 3017 |
| 16-1213. QUEUE_88_STATUS_A Register Field Descriptions ..... | 3018 |
| 16-1214. QUEUE_88_STATUS_B Register Field Descriptions ..... | 3019 |
| 16-1215. QUEUE_88_STATUS_C Register Field Descriptions ..... | 3020 |
| 16-1216. QUEUE_89_STATUS_A Register Field Descriptions ..... | 3021 |
| 16-1217. QUEUE_89_STATUS_B Register Field Descriptions ..... | 3022 |
| 16-1218. QUEUE_89_STATUS_C Register Field Descriptions ..... | 3023 |
| 16-1219. QUEUE_90_STATUS_A Register Field Descriptions ..... | 3024 |
| 16-1220. QUEUE_90_STATUS_B Register Field Descriptions ..... | 3025 |
| 16-1221. QUEUE_90_STATUS_C Register Field Descriptions ..... | 3026 |
| 16-1222. QUEUE_91_STATUS_A Register Field Descriptions ..... | 3027 |
| 16-1223. QUEUE_91_STATUS_B Register Field Descriptions ..... | 3028 |
| 16-1224. QUEUE_91_STATUS_C Register Field Descriptions ..... | 3029 |
| 16-1225. QUEUE_92_STATUS_A Register Field Descriptions ..... | 3030 |
| 16-1226. QUEUE_92_STATUS_B Register Field Descriptions ..... | 3031 |
| 16-1227. QUEUE_92_STATUS_C Register Field Descriptions ..... | 3032 |
| 16-1228. QUEUE_93_STATUS_A Register Field Descriptions ..... | 3033 |

|   |      |
|---|------|
| 16-1229. QUEUE_93_STATUS_B Register Field Descriptions .....  | 3034 |
| 16-1230. QUEUE_93_STATUS_C Register Field Descriptions .....  | 3035 |
| 16-1231. QUEUE_94_STATUS_A Register Field Descriptions .....  | 3036 |
| 16-1232. QUEUE_94_STATUS_B Register Field Descriptions .....  | 3037 |
| 16-1233. QUEUE_94_STATUS_C Register Field Descriptions .....  | 3038 |
| 16-1234. QUEUE_95_STATUS_A Register Field Descriptions .....  | 3039 |
| 16-1235. QUEUE_95_STATUS_B Register Field Descriptions .....  | 3040 |
| 16-1236. QUEUE_95_STATUS_C Register Field Descriptions .....  | 3041 |
| 16-1237. QUEUE_96_STATUS_A Register Field Descriptions .....  | 3042 |
| 16-1238. QUEUE_96_STATUS_B Register Field Descriptions .....  | 3043 |
| 16-1239. QUEUE_96_STATUS_C Register Field Descriptions .....  | 3044 |
| 16-1240. QUEUE_97_STATUS_A Register Field Descriptions .....  | 3045 |
| 16-1241. QUEUE_97_STATUS_B Register Field Descriptions .....  | 3046 |
| 16-1242. QUEUE_97_STATUS_C Register Field Descriptions .....  | 3047 |
| 16-1243. QUEUE_98_STATUS_A Register Field Descriptions .....  | 3048 |
| 16-1244. QUEUE_98_STATUS_B Register Field Descriptions .....  | 3049 |
| 16-1245. QUEUE_98_STATUS_C Register Field Descriptions .....  | 3050 |
| 16-1246. QUEUE_99_STATUS_A Register Field Descriptions .....  | 3051 |
| 16-1247. QUEUE_99_STATUS_B Register Field Descriptions .....  | 3052 |
| 16-1248. QUEUE_99_STATUS_C Register Field Descriptions .....  | 3053 |
| 16-1249. QUEUE_100_STATUS_A Register Field Descriptions ..... | 3054 |
| 16-1250. QUEUE_100_STATUS_B Register Field Descriptions ..... | 3055 |
| 16-1251. QUEUE_100_STATUS_C Register Field Descriptions ..... | 3056 |
| 16-1252. QUEUE_101_STATUS_A Register Field Descriptions ..... | 3057 |
| 16-1253. QUEUE_101_STATUS_B Register Field Descriptions ..... | 3058 |
| 16-1254. QUEUE_101_STATUS_C Register Field Descriptions ..... | 3059 |
| 16-1255. QUEUE_102_STATUS_A Register Field Descriptions ..... | 3060 |
| 16-1256. QUEUE_102_STATUS_B Register Field Descriptions ..... | 3061 |
| 16-1257. QUEUE_102_STATUS_C Register Field Descriptions ..... | 3062 |
| 16-1258. QUEUE_103_STATUS_A Register Field Descriptions ..... | 3063 |
| 16-1259. QUEUE_103_STATUS_B Register Field Descriptions ..... | 3064 |
| 16-1260. QUEUE_103_STATUS_C Register Field Descriptions ..... | 3065 |
| 16-1261. QUEUE_104_STATUS_A Register Field Descriptions ..... | 3066 |
| 16-1262. QUEUE_104_STATUS_B Register Field Descriptions ..... | 3067 |
| 16-1263. QUEUE_104_STATUS_C Register Field Descriptions ..... | 3068 |
| 16-1264. QUEUE_105_STATUS_A Register Field Descriptions ..... | 3069 |
| 16-1265. QUEUE_105_STATUS_B Register Field Descriptions ..... | 3070 |
| 16-1266. QUEUE_105_STATUS_C Register Field Descriptions ..... | 3071 |
| 16-1267. QUEUE_106_STATUS_A Register Field Descriptions ..... | 3072 |
| 16-1268. QUEUE_106_STATUS_B Register Field Descriptions ..... | 3073 |
| 16-1269. QUEUE_106_STATUS_C Register Field Descriptions ..... | 3074 |
| 16-1270. QUEUE_107_STATUS_A Register Field Descriptions ..... | 3075 |
| 16-1271. QUEUE_107_STATUS_B Register Field Descriptions ..... | 3076 |
| 16-1272. QUEUE_107_STATUS_C Register Field Descriptions ..... | 3077 |
| 16-1273. QUEUE_108_STATUS_A Register Field Descriptions ..... | 3078 |
| 16-1274. QUEUE_108_STATUS_B Register Field Descriptions ..... | 3079 |
| 16-1275. QUEUE_108_STATUS_C Register Field Descriptions ..... | 3080 |
| 16-1276. QUEUE_109_STATUS_A Register Field Descriptions ..... | 3081 |
| 16-1277. QUEUE_109_STATUS_B Register Field Descriptions ..... | 3082 |

|   |      |
|---|------|
| 16-1278. QUEUE_109_STATUS_C Register Field Descriptions.....  | 3083 |
| 16-1279. QUEUE_110_STATUS_A Register Field Descriptions ..... | 3084 |
| 16-1280. QUEUE_110_STATUS_B Register Field Descriptions ..... | 3085 |
| 16-1281. QUEUE_110_STATUS_C Register Field Descriptions ..... | 3086 |
| 16-1282. QUEUE_111_STATUS_A Register Field Descriptions ..... | 3087 |
| 16-1283. QUEUE_111_STATUS_B Register Field Descriptions ..... | 3088 |
| 16-1284. QUEUE_111_STATUS_C Register Field Descriptions ..... | 3089 |
| 16-1285. QUEUE_112_STATUS_A Register Field Descriptions ..... | 3090 |
| 16-1286. QUEUE_112_STATUS_B Register Field Descriptions ..... | 3091 |
| 16-1287. QUEUE_112_STATUS_C Register Field Descriptions ..... | 3092 |
| 16-1288. QUEUE_113_STATUS_A Register Field Descriptions ..... | 3093 |
| 16-1289. QUEUE_113_STATUS_B Register Field Descriptions ..... | 3094 |
| 16-1290. QUEUE_113_STATUS_C Register Field Descriptions ..... | 3095 |
| 16-1291. QUEUE_114_STATUS_A Register Field Descriptions ..... | 3096 |
| 16-1292. QUEUE_114_STATUS_B Register Field Descriptions ..... | 3097 |
| 16-1293. QUEUE_114_STATUS_C Register Field Descriptions ..... | 3098 |
| 16-1294. QUEUE_115_STATUS_A Register Field Descriptions ..... | 3099 |
| 16-1295. QUEUE_115_STATUS_B Register Field Descriptions ..... | 3100 |
| 16-1296. QUEUE_115_STATUS_C Register Field Descriptions ..... | 3101 |
| 16-1297. QUEUE_116_STATUS_A Register Field Descriptions ..... | 3102 |
| 16-1298. QUEUE_116_STATUS_B Register Field Descriptions ..... | 3103 |
| 16-1299. QUEUE_116_STATUS_C Register Field Descriptions ..... | 3104 |
| 16-1300. QUEUE_117_STATUS_A Register Field Descriptions ..... | 3105 |
| 16-1301. QUEUE_117_STATUS_B Register Field Descriptions ..... | 3106 |
| 16-1302. QUEUE_117_STATUS_C Register Field Descriptions ..... | 3107 |
| 16-1303. QUEUE_118_STATUS_A Register Field Descriptions ..... | 3108 |
| 16-1304. QUEUE_118_STATUS_B Register Field Descriptions ..... | 3109 |
| 16-1305. QUEUE_118_STATUS_C Register Field Descriptions ..... | 3110 |
| 16-1306. QUEUE_119_STATUS_A Register Field Descriptions ..... | 3111 |
| 16-1307. QUEUE_119_STATUS_B Register Field Descriptions ..... | 3112 |
| 16-1308. QUEUE_119_STATUS_C Register Field Descriptions ..... | 3113 |
| 16-1309. QUEUE_120_STATUS_A Register Field Descriptions ..... | 3114 |
| 16-1310. QUEUE_120_STATUS_B Register Field Descriptions ..... | 3115 |
| 16-1311. QUEUE_120_STATUS_C Register Field Descriptions ..... | 3116 |
| 16-1312. QUEUE_121_STATUS_A Register Field Descriptions ..... | 3117 |
| 16-1313. QUEUE_121_STATUS_B Register Field Descriptions ..... | 3118 |
| 16-1314. QUEUE_121_STATUS_C Register Field Descriptions ..... | 3119 |
| 16-1315. QUEUE_122_STATUS_A Register Field Descriptions ..... | 3120 |
| 16-1316. QUEUE_122_STATUS_B Register Field Descriptions ..... | 3121 |
| 16-1317. QUEUE_122_STATUS_C Register Field Descriptions ..... | 3122 |
| 16-1318. QUEUE_123_STATUS_A Register Field Descriptions ..... | 3123 |
| 16-1319. QUEUE_123_STATUS_B Register Field Descriptions ..... | 3124 |
| 16-1320. QUEUE_123_STATUS_C Register Field Descriptions ..... | 3125 |
| 16-1321. QUEUE_124_STATUS_A Register Field Descriptions ..... | 3126 |
| 16-1322. QUEUE_124_STATUS_B Register Field Descriptions ..... | 3127 |
| 16-1323. QUEUE_124_STATUS_C Register Field Descriptions ..... | 3128 |
| 16-1324. QUEUE_125_STATUS_A Register Field Descriptions ..... | 3129 |
| 16-1325. QUEUE_125_STATUS_B Register Field Descriptions ..... | 3130 |
| 16-1326. QUEUE_125_STATUS_C Register Field Descriptions ..... | 3131 |

|   |      |
|---|------|
| 16-1327. QUEUE_126_STATUS_A Register Field Descriptions ..... | 3132 |
| 16-1328. QUEUE_126_STATUS_B Register Field Descriptions ..... | 3133 |
| 16-1329. QUEUE_126_STATUS_C Register Field Descriptions.....  | 3134 |
| 16-1330. QUEUE_127_STATUS_A Register Field Descriptions ..... | 3135 |
| 16-1331. QUEUE_127_STATUS_B Register Field Descriptions ..... | 3136 |
| 16-1332. QUEUE_127_STATUS_C Register Field Descriptions.....  | 3137 |
| 16-1333. QUEUE_128_STATUS_A Register Field Descriptions ..... | 3138 |
| 16-1334. QUEUE_128_STATUS_B Register Field Descriptions ..... | 3139 |
| 16-1335. QUEUE_128_STATUS_C Register Field Descriptions.....  | 3140 |
| 16-1336. QUEUE_129_STATUS_A Register Field Descriptions ..... | 3141 |
| 16-1337. QUEUE_129_STATUS_B Register Field Descriptions ..... | 3142 |
| 16-1338. QUEUE_129_STATUS_C Register Field Descriptions.....  | 3143 |
| 16-1339. QUEUE_130_STATUS_A Register Field Descriptions ..... | 3144 |
| 16-1340. QUEUE_130_STATUS_B Register Field Descriptions ..... | 3145 |
| 16-1341. QUEUE_130_STATUS_C Register Field Descriptions.....  | 3146 |
| 16-1342. QUEUE_131_STATUS_A Register Field Descriptions ..... | 3147 |
| 16-1343. QUEUE_131_STATUS_B Register Field Descriptions ..... | 3148 |
| 16-1344. QUEUE_131_STATUS_C Register Field Descriptions.....  | 3149 |
| 16-1345. QUEUE_132_STATUS_A Register Field Descriptions ..... | 3150 |
| 16-1346. QUEUE_132_STATUS_B Register Field Descriptions ..... | 3151 |
| 16-1347. QUEUE_132_STATUS_C Register Field Descriptions.....  | 3152 |
| 16-1348. QUEUE_133_STATUS_A Register Field Descriptions ..... | 3153 |
| 16-1349. QUEUE_133_STATUS_B Register Field Descriptions ..... | 3154 |
| 16-1350. QUEUE_133_STATUS_C Register Field Descriptions.....  | 3155 |
| 16-1351. QUEUE_134_STATUS_A Register Field Descriptions ..... | 3156 |
| 16-1352. QUEUE_134_STATUS_B Register Field Descriptions ..... | 3157 |
| 16-1353. QUEUE_134_STATUS_C Register Field Descriptions.....  | 3158 |
| 16-1354. QUEUE_135_STATUS_A Register Field Descriptions ..... | 3159 |
| 16-1355. QUEUE_135_STATUS_B Register Field Descriptions ..... | 3160 |
| 16-1356. QUEUE_135_STATUS_C Register Field Descriptions.....  | 3161 |
| 16-1357. QUEUE_136_STATUS_A Register Field Descriptions ..... | 3162 |
| 16-1358. QUEUE_136_STATUS_B Register Field Descriptions ..... | 3163 |
| 16-1359. QUEUE_136_STATUS_C Register Field Descriptions.....  | 3164 |
| 16-1360. QUEUE_137_STATUS_A Register Field Descriptions ..... | 3165 |
| 16-1361. QUEUE_137_STATUS_B Register Field Descriptions ..... | 3166 |
| 16-1362. QUEUE_137_STATUS_C Register Field Descriptions.....  | 3167 |
| 16-1363. QUEUE_138_STATUS_A Register Field Descriptions ..... | 3168 |
| 16-1364. QUEUE_138_STATUS_B Register Field Descriptions ..... | 3169 |
| 16-1365. QUEUE_138_STATUS_C Register Field Descriptions.....  | 3170 |
| 16-1366. QUEUE_139_STATUS_A Register Field Descriptions ..... | 3171 |
| 16-1367. QUEUE_139_STATUS_B Register Field Descriptions ..... | 3172 |
| 16-1368. QUEUE_139_STATUS_C Register Field Descriptions.....  | 3173 |
| 16-1369. QUEUE_140_STATUS_A Register Field Descriptions ..... | 3174 |
| 16-1370. QUEUE_140_STATUS_B Register Field Descriptions ..... | 3175 |
| 16-1371. QUEUE_140_STATUS_C Register Field Descriptions.....  | 3176 |
| 16-1372. QUEUE_141_STATUS_A Register Field Descriptions ..... | 3177 |
| 16-1373. QUEUE_141_STATUS_B Register Field Descriptions ..... | 3178 |
| 16-1374. QUEUE_141_STATUS_C Register Field Descriptions.....  | 3179 |
| 16-1375. QUEUE_142_STATUS_A Register Field Descriptions ..... | 3180 |

|   |      |
|---|------|
| 16-1376. QUEUE_142_STATUS_B Register Field Descriptions .....               | 3181 |
| 16-1377. QUEUE_142_STATUS_C Register Field Descriptions .....               | 3182 |
| 16-1378. QUEUE_143_STATUS_A Register Field Descriptions .....               | 3183 |
| 16-1379. QUEUE_143_STATUS_B Register Field Descriptions .....               | 3184 |
| 16-1380. QUEUE_143_STATUS_C Register Field Descriptions .....               | 3185 |
| 16-1381. QUEUE_144_STATUS_A Register Field Descriptions .....               | 3186 |
| 16-1382. QUEUE_144_STATUS_B Register Field Descriptions .....               | 3187 |
| 16-1383. QUEUE_144_STATUS_C Register Field Descriptions .....               | 3188 |
| 16-1384. QUEUE_145_STATUS_A Register Field Descriptions .....               | 3189 |
| 16-1385. QUEUE_145_STATUS_B Register Field Descriptions .....               | 3190 |
| 16-1386. QUEUE_145_STATUS_C Register Field Descriptions .....               | 3191 |
| 16-1387. QUEUE_146_STATUS_A Register Field Descriptions .....               | 3192 |
| 16-1388. QUEUE_146_STATUS_B Register Field Descriptions .....               | 3193 |
| 16-1389. QUEUE_146_STATUS_C Register Field Descriptions .....               | 3194 |
| 16-1390. QUEUE_147_STATUS_A Register Field Descriptions .....               | 3195 |
| 16-1391. QUEUE_147_STATUS_B Register Field Descriptions .....               | 3196 |
| 16-1392. QUEUE_147_STATUS_C Register Field Descriptions .....               | 3197 |
| 16-1393. QUEUE_148_STATUS_A Register Field Descriptions .....               | 3198 |
| 16-1394. QUEUE_148_STATUS_B Register Field Descriptions .....               | 3199 |
| 16-1395. QUEUE_148_STATUS_C Register Field Descriptions .....               | 3200 |
| 16-1396. QUEUE_149_STATUS_A Register Field Descriptions .....               | 3201 |
| 16-1397. QUEUE_149_STATUS_B Register Field Descriptions .....               | 3202 |
| 16-1398. QUEUE_149_STATUS_C Register Field Descriptions .....               | 3203 |
| 16-1399. QUEUE_150_STATUS_A Register Field Descriptions .....               | 3204 |
| 16-1400. QUEUE_150_STATUS_B Register Field Descriptions .....               | 3205 |
| 16-1401. QUEUE_150_STATUS_C Register Field Descriptions .....               | 3206 |
| 16-1402. QUEUE_151_STATUS_A Register Field Descriptions .....               | 3207 |
| 16-1403. QUEUE_151_STATUS_B Register Field Descriptions .....               | 3208 |
| 16-1404. QUEUE_151_STATUS_C Register Field Descriptions .....               | 3209 |
| 16-1405. QUEUE_152_STATUS_A Register Field Descriptions .....               | 3210 |
| 16-1406. QUEUE_152_STATUS_B Register Field Descriptions .....               | 3211 |
| 16-1407. QUEUE_152_STATUS_C Register Field Descriptions .....               | 3212 |
| 16-1408. QUEUE_153_STATUS_A Register Field Descriptions .....               | 3213 |
| 16-1409. QUEUE_153_STATUS_B Register Field Descriptions .....               | 3214 |
| 16-1410. QUEUE_153_STATUS_C Register Field Descriptions .....               | 3215 |
| 16-1411. QUEUE_154_STATUS_A Register Field Descriptions .....               | 3216 |
| 16-1412. QUEUE_154_STATUS_B Register Field Descriptions .....               | 3217 |
| 16-1413. QUEUE_154_STATUS_C Register Field Descriptions .....               | 3218 |
| 16-1414. QUEUE_155_STATUS_A Register Field Descriptions .....               | 3219 |
| 16-1415. QUEUE_155_STATUS_B Register Field Descriptions .....               | 3220 |
| 16-1416. QUEUE_155_STATUS_C Register Field Descriptions .....               | 3221 |
| 17-1. Mailbox Connectivity Attributes .....                                 | 3224 |
| 17-2. Mailbox Clock Signals .....   | 3225 |
| 17-3. Mailbox Implementation .....  | 3225 |
| 17-4. Local Power Management Features .....                                 | 3226 |
| 17-5. Interrupt Events .....  | 3227 |
| 17-6. Global Initialization of Surrounding Modules for System Mailbox ..... | 3229 |
| 17-7. Mailbox Global Initialization .....                                   | 3230 |
| 17-8. Sending a Message (Polling Method) .....                              | 3230 |

|        |   |      |
|--------|---|------|
| 17-9.  | Sending a Message (Interrupt Method).....         | 3230 |
| 17-10. | Receiving a Message (Polling Method).....         | 3231 |
| 17-11. | Receiving a Message (Interrupt Method).....       | 3231 |
| 17-12. | Events Servicing in Sending Mode.....             | 3231 |
| 17-13. | Events Servicing in Receiving Mode.....           | 3231 |
| 17-14. | MAILBOX REGISTERS.....                            | 3232 |
| 17-15. | REVISION Register Field Descriptions .....        | 3235 |
| 17-16. | SYSCONFIG Register Field Descriptions .....       | 3236 |
| 17-17. | MESSAGE_0 Register Field Descriptions .....       | 3237 |
| 17-18. | MESSAGE_1 Register Field Descriptions .....       | 3238 |
| 17-19. | MESSAGE_2 Register Field Descriptions .....       | 3239 |
| 17-20. | MESSAGE_3 Register Field Descriptions .....       | 3240 |
| 17-21. | MESSAGE_4 Register Field Descriptions .....       | 3241 |
| 17-22. | MESSAGE_5 Register Field Descriptions .....       | 3242 |
| 17-23. | MESSAGE_6 Register Field Descriptions .....       | 3243 |
| 17-24. | MESSAGE_7 Register Field Descriptions .....       | 3244 |
| 17-25. | FIFOSTATUS_0 Register Field Descriptions .....    | 3245 |
| 17-26. | FIFOSTATUS_1 Register Field Descriptions .....    | 3246 |
| 17-27. | FIFOSTATUS_2 Register Field Descriptions .....    | 3247 |
| 17-28. | FIFOSTATUS_3 Register Field Descriptions .....    | 3248 |
| 17-29. | FIFOSTATUS_4 Register Field Descriptions .....    | 3249 |
| 17-30. | FIFOSTATUS_5 Register Field Descriptions .....    | 3250 |
| 17-31. | FIFOSTATUS_6 Register Field Descriptions .....    | 3251 |
| 17-32. | FIFOSTATUS_7 Register Field Descriptions .....    | 3252 |
| 17-33. | MSGSTATUS_0 Register Field Descriptions .....     | 3253 |
| 17-34. | MSGSTATUS_1 Register Field Descriptions .....     | 3254 |
| 17-35. | MSGSTATUS_2 Register Field Descriptions .....     | 3255 |
| 17-36. | MSGSTATUS_3 Register Field Descriptions .....     | 3256 |
| 17-37. | MSGSTATUS_4 Register Field Descriptions .....     | 3257 |
| 17-38. | MSGSTATUS_5 Register Field Descriptions .....     | 3258 |
| 17-39. | MSGSTATUS_6 Register Field Descriptions .....     | 3259 |
| 17-40. | MSGSTATUS_7 Register Field Descriptions .....     | 3260 |
| 17-41. | IRQSTATUS_RAW_0 Register Field Descriptions.....  | 3261 |
| 17-42. | IRQSTATUS_CLR_0 Register Field Descriptions ..... | 3263 |
| 17-43. | IRQENABLE_SET_0 Register Field Descriptions.....  | 3265 |
| 17-44. | IRQENABLE_CLR_0 Register Field Descriptions ..... | 3267 |
| 17-45. | IRQSTATUS_RAW_1 Register Field Descriptions.....  | 3269 |
| 17-46. | IRQSTATUS_CLR_1 Register Field Descriptions ..... | 3271 |
| 17-47. | IRQENABLE_SET_1 Register Field Descriptions.....  | 3273 |
| 17-48. | IRQENABLE_CLR_1 Register Field Descriptions ..... | 3275 |
| 17-49. | IRQSTATUS_RAW_2 Register Field Descriptions.....  | 3277 |
| 17-50. | IRQSTATUS_CLR_2 Register Field Descriptions ..... | 3279 |
| 17-51. | IRQENABLE_SET_2 Register Field Descriptions.....  | 3281 |
| 17-52. | IRQENABLE_CLR_2 Register Field Descriptions ..... | 3283 |
| 17-53. | IRQSTATUS_RAW_3 Register Field Descriptions.....  | 3285 |
| 17-54. | IRQSTATUS_CLR_3 Register Field Descriptions ..... | 3287 |
| 17-55. | IRQENABLE_SET_3 Register Field Descriptions.....  | 3289 |
| 17-56. | IRQENABLE_CLR_3 Register Field Descriptions ..... | 3291 |
| 17-57. | SPINLOCK REGISTERS .....                          | 3293 |

---

|   |      |
|---|------|
| 17-58. REV Register Field Descriptions .....                  | 3296 |
| 17-59. SYSCONFIG Register Field Descriptions .....            | 3297 |
| 17-60. SYSTATUS Register Field Descriptions.....              | 3298 |
| 17-61. LOCK_REG_0 Register Field Descriptions .....           | 3299 |
| 17-62. LOCK_REG_1 Register Field Descriptions .....           | 3300 |
| 17-63. LOCK_REG_2 Register Field Descriptions .....           | 3301 |
| 17-64. LOCK_REG_3 Register Field Descriptions .....           | 3302 |
| 17-65. LOCK_REG_4 Register Field Descriptions .....           | 3303 |
| 17-66. LOCK_REG_5 Register Field Descriptions .....           | 3304 |
| 17-67. LOCK_REG_6 Register Field Descriptions .....           | 3305 |
| 17-68. LOCK_REG_7 Register Field Descriptions .....           | 3306 |
| 17-69. LOCK_REG_8 Register Field Descriptions .....           | 3307 |
| 17-70. LOCK_REG_9 Register Field Descriptions .....           | 3308 |
| 17-71. LOCK_REG_10 Register Field Descriptions.....           | 3309 |
| 17-72. LOCK_REG_11 Register Field Descriptions.....           | 3310 |
| 17-73. LOCK_REG_12 Register Field Descriptions..              | 3311 |
| 17-74. LOCK_REG_13 Register Field Descriptions.....           | 3312 |
| 17-75. LOCK_REG_14 Register Field Descriptions.....           | 3313 |
| 17-76. LOCK_REG_15 Register Field Descriptions.....           | 3314 |
| 17-77. LOCK_REG_16 Register Field Descriptions.....           | 3315 |
| 17-78. LOCK_REG_17 Register Field Descriptions.....           | 3316 |
| 17-79. LOCK_REG_18 Register Field Descriptions.....           | 3317 |
| 17-80. LOCK_REG_19 Register Field Descriptions.....           | 3318 |
| 17-81. LOCK_REG_20 Register Field Descriptions.....           | 3319 |
| 17-82. LOCK_REG_21 Register Field Descriptions.....           | 3320 |
| 17-83. LOCK_REG_22 Register Field Descriptions.....           | 3321 |
| 17-84. LOCK_REG_23 Register Field Descriptions.....           | 3322 |
| 17-85. LOCK_REG_24 Register Field Descriptions.....           | 3323 |
| 17-86. LOCK_REG_25 Register Field Descriptions.....           | 3324 |
| 17-87. LOCK_REG_26 Register Field Descriptions.....           | 3325 |
| 17-88. LOCK_REG_27 Register Field Descriptions.....           | 3326 |
| 17-89. LOCK_REG_28 Register Field Descriptions.....           | 3327 |
| 17-90. LOCK_REG_29 Register Field Descriptions.....           | 3328 |
| 17-91. LOCK_REG_30 Register Field Descriptions.....           | 3329 |
| 17-92. LOCK_REG_31 Register Field Descriptions.....           | 3330 |
| 18-1. Unsupported MMCHS Features .....                        | 3332 |
| 18-2. MMCHS Connectivity Attributes .....                     | 3334 |
| 18-3. MMCHS Clock Signals.....                                | 3335 |
| 18-4. MMCHS Pin List .....                                    | 3335 |
| 18-5. DAT Line Direction for Data Transfer Modes .....        | 3335 |
| 18-6. ADPDATDIROQ and ADPDATDIRLS Signal States .....         | 3336 |
| 18-7. MMC/SD/SDIO Controller Pins and Descriptions .....      | 3339 |
| 18-8. Response Type Summary .....                             | 3342 |
| 18-9. Local Power Management Features .....                   | 3347 |
| 18-10. Clock Activity Settings .....                          | 3347 |
| 18-11. Events.....  | 3348 |
| 18-12. Memory Size, BLEN, and Buffer Relationship .....       | 3355 |
| 18-13. MMC, SD, SDIO Responses in the SD_RSPxx Registers..... | 3356 |
| 18-14. CC and TC Values Upon Error Detected.....              | 3357 |

|   |      |
|---|------|
| 18-15. MMC/SD/SDIO Controller Transfer Stop Command Summary ..... | 3364 |
| 18-16. MMC/SD/SDIO Hardware Status Features .....                 | 3370 |
| 18-17. Global Init for Surrounding Modules .....                  | 3371 |
| 18-18. MMC/SD/SDIO Controller Wake-Up Configuration .....         | 3372 |
| 18-19. MULTIMEDIA_CARD REGISTERS.....                             | 3376 |
| 18-20. SD_SYS CONFIG Register Field Descriptions.....             | 3377 |
| 18-21. SD_SYS STATUS Register Field Descriptions.....             | 3379 |
| 18-22. SD_CSRE Register Field Descriptions .....                  | 3380 |
| 18-23. SD_SYSTEST Register Field Descriptions .....               | 3381 |
| 18-24. SD_CON Register Field Descriptions.....                    | 3385 |
| 18-25. SD_PWCNT Register Field Descriptions.....                  | 3389 |
| 18-26. SD_SDMASA Register Field Descriptions .....                | 3390 |
| 18-27. SD_BLK Register Field Descriptions .....                   | 3391 |
| 18-28. SD_ARG Register Field Descriptions.....                    | 3392 |
| 18-29. SD_CMD Register Field Descriptions .....                   | 3394 |
| 18-30. SD_RSP10 Register Field Descriptions .....                 | 3398 |
| 18-31. SD_RSP32 Register Field Descriptions .....                 | 3399 |
| 18-32. SD_RSP54 Register Field Descriptions .....                 | 3400 |
| 18-33. SD_RSP76 Register Field Descriptions .....                 | 3401 |
| 18-34. SD_DATA Register Field Descriptions .....                  | 3402 |
| 18-35. SD_PSTATE Register Field Descriptions .....                | 3403 |
| 18-36. SD_HCTL Register Field Descriptions.....                   | 3406 |
| 18-37. SD_SYS CTL Register Field Descriptions .....               | 3409 |
| 18-38. SD_STAT Register Field Descriptions.....                   | 3411 |
| 18-39. SD_IE Register Field Descriptions .....                    | 3416 |
| 18-40. SD_ISE Register Field Descriptions .....                   | 3419 |
| 18-41. SD_AC12 Register Field Descriptions .....                  | 3422 |
| 18-42. SD_CAPA Register Field Descriptions .....                  | 3424 |
| 18-43. SD_CUR_CAPA Register Field Descriptions .....              | 3426 |
| 18-44. SD_FE Register Field Descriptions .....                    | 3427 |
| 18-45. SD_ADMAES Register Field Descriptions .....                | 3429 |
| 18-46. SD_ADMASAL Register Field Descriptions .....               | 3430 |
| 18-47. SD_ADMASAH Register Field Descriptions .....               | 3431 |
| 18-48. SD_REV Register Field Descriptions .....                   | 3432 |
| 19-1. Unsupported UART Features .....                             | 3435 |
| 19-2. UART0 Connectivity Attributes .....                         | 3436 |
| 19-3. UART1–5 Connectivity Attributes .....                       | 3437 |
| 19-4. UART0 Clock Signals .....                                   | 3437 |
| 19-5. UART1–5 Clock Signals .....                                 | 3437 |
| 19-6. UART Mode Baud and Error Rates .....                        | 3438 |
| 19-7. IrDA Mode Baud and Error Rates .....                        | 3438 |
| 19-8. UART Pin List.....  | 3439 |
| 19-9. UART Muxing Control.....                                    | 3439 |
| 19-10. Local Power-Management Features .....                      | 3443 |
| 19-11. UART Mode Interrupts .....                                 | 3443 |
| 19-12. IrDA Mode Interrupts .....                                 | 3444 |
| 19-13. CIR Mode Interrupts.....                                   | 3445 |
| 19-14. TX FIFO Trigger Level Setting Summary .....                | 3447 |
| 19-15. RX FIFO Trigger Level Setting Summary .....                | 3447 |

|   |      |
|---|------|
| 19-16. UART/IrDA/CIR Register Access Mode Programming (Using UART_LCR)..... | 3455 |
| 19-17. Subconfiguration Mode A Summary .....                                | 3455 |
| 19-18. Subconfiguration Mode B Summary .....                                | 3455 |
| 19-19. Suboperational Mode Summary .....                                    | 3455 |
| 19-20. UART/IrDA/CIR Register Access Mode Overview .....                    | 3455 |
| 19-21. UART Mode Selection .....  | 3457 |
| 19-22. UART Mode Register Overview .....                                    | 3457 |
| 19-23. IrDA Mode Register Overview .....                                    | 3458 |
| 19-24. CIR Mode Register Overview .....                                     | 3459 |
| 19-25. UART Baud Rate Settings (48-MHz Clock).....                          | 3462 |
| 19-26. UART Parity Bit Encoding .....                                       | 3462 |
| 19-27. UART_EFR[3:0] Software Flow Control Options.....                     | 3464 |
| 19-28. IrDA Baud Rate Settings .....  | 3473 |
| 19-29. UART Registers .....   | 3492 |
| 19-30. Receiver Holding Register (RHR) Field Descriptions.....              | 3494 |
| 19-31. Transmit Holding Register (THR) Field Descriptions .....             | 3494 |
| 19-32. UART Interrupt Enable Register (IER) Field Descriptions .....        | 3495 |
| 19-33. IrDA Interrupt Enable Register (IER) Field Descriptions .....        | 3496 |
| 19-34. CIR Interrupt Enable Register (IER) Field Descriptions .....         | 3497 |
| 19-35. UART Interrupt Identification Register (IIR) Field Descriptions..... | 3498 |
| 19-36. IrDA Interrupt Identification Register (IIR) Field Descriptions..... | 3499 |
| 19-37. CIR Interrupt Identification Register (IIR) Field Descriptions ..... | 3500 |
| 19-38. FIFO Control Register (FCR) Field Descriptions .....                 | 3501 |
| 19-39. Line Control Register (LCR) Field Descriptions .....                 | 3502 |
| 19-40. Modem Control Register (MCR) Field Descriptions.....                 | 3503 |
| 19-41. UART Line Status Register (LSR) Field Descriptions .....             | 3504 |
| 19-42. IrDA Line Status Register (LSR) Field Descriptions .....             | 3505 |
| 19-43. CIR Line Status Register (LSR) Field Descriptions .....              | 3506 |
| 19-44. Modem Status Register (MSR) Field Descriptions .....                 | 3507 |
| 19-45. Transmission Control Register (TCR) Field Descriptions .....         | 3508 |
| 19-46. Scratchpad Register (SPR) Field Descriptions .....                   | 3508 |
| 19-47. Trigger Level Register (TLR) Field Descriptions .....                | 3509 |
| 19-48. RX FIFO Trigger Level Setting Summary .....                          | 3509 |
| 19-49. TX FIFO Trigger Level Setting Summary .....                          | 3509 |
| 19-50. Mode Definition Register 1 (MDR1) Field Descriptions .....           | 3510 |
| 19-51. Mode Definition Register 2 (MDR2) Field Descriptions .....           | 3511 |
| 19-52. Status FIFO Line Status Register (SFLSR) Field Descriptions.....     | 3512 |
| 19-53. RESUME Register Field Descriptions .....                             | 3512 |
| 19-54. Status FIFO Register Low (SFREGL) Field Descriptions .....           | 3513 |
| 19-55. Status FIFO Register High (SFREGH) Field Descriptions .....          | 3513 |
| 19-56. BOF Control Register (BLR) Field Descriptions .....                  | 3514 |
| 19-57. Auxiliary Control Register (ACREG) Field Descriptions .....          | 3515 |
| 19-58. Supplementary Control Register (SCR) Field Descriptions.....         | 3516 |
| 19-59. Supplementary Status Register (SSR) Field Descriptions .....         | 3517 |
| 19-60. BOF Length Register (EBLR) Field Descriptions.....                   | 3518 |
| 19-61. Module Version Register (MVR) Field Descriptions .....               | 3519 |
| 19-62. System Configuration Register (SYSC) Field Descriptions .....        | 3520 |
| 19-63. System Status Register (SYSS) Field Descriptions.....                | 3520 |
| 19-64. Wake-Up Enable Register (WER) Field Descriptions .....               | 3521 |

|  |      |
|--|------|
| 19-65. Carrier Frequency Prescaler Register (CFPS) Field Descriptions .....          | 3522 |
| 19-66. Divisor Latches Low Register (DLL) Field Descriptions .....                   | 3523 |
| 19-67. Divisor Latches High Register (DLH) Field Descriptions .....                  | 3523 |
| 19-68. Enhanced Feature Register (EFR) Field Descriptions.....                       | 3524 |
| 19-69. EFR[3:0] Software Flow Control Options .....                                  | 3525 |
| 19-70. XON1/ADDR1 Register Field Descriptions.....                                   | 3525 |
| 19-71. XON2/ADDR2 Register Field Descriptions.....                                   | 3525 |
| 19-72. XOFF1 Register Field Descriptions .....                                       | 3526 |
| 19-73. XOFF2 Register Field Descriptions .....                                       | 3526 |
| 19-74. Transmit Frame Length Low Register (TXFLL) Field Descriptions .....           | 3527 |
| 19-75. Transmit Frame Length High Register (TXFLH) Field Descriptions .....          | 3527 |
| 19-76. Received Frame Length Low Register (RXFLL) Field Descriptions .....           | 3528 |
| 19-77. Received Frame Length High Register (RXFLH) Field Descriptions .....          | 3528 |
| 19-78. UART Autobauding Status Register (UASR) Field Descriptions .....              | 3529 |
| 19-79. RXFIFO_LVL Register Field Descriptions.....                                   | 3530 |
| 19-80. TXFIFO_LVL Register Field Descriptions .....                                  | 3531 |
| 19-81. IER2 Register Field Descriptions.....   | 3532 |
| 19-82. ISR2 Register Field Descriptions .....  | 3533 |
| 19-83. FREQ_SEL Register Field Descriptions.....                                     | 3534 |
| 19-84. Mode Definition Register 3 (MDR3) Register Field Descriptions.....            | 3535 |
| 19-85. TX_DMA_THRESHOLD Register Field Descriptions.....                             | 3536 |
| 20-1. Timer Resolution and Maximum Range .....                                       | 3538 |
| 20-2. Timer[0] Connectivity Attributes .....   | 3540 |
| 20-3. Timer[2-7] Connectivity Attributes .....                                       | 3540 |
| 20-4. Timer Clock Signals .....  | 3541 |
| 20-5. Timer Pin List .....   | 3541 |
| 20-6. Prescaler Functionality.....   | 3544 |
| 20-7. Prescaler Clock Ratios Value .....   | 3547 |
| 20-8. Value and Corresponding Interrupt Period.....                                  | 3547 |
| 20-9. OCP Error Reporting .....  | 3548 |
| 20-10. TIMER REGISTERS .....   | 3551 |
| 20-11. Identification Register Register Field Descriptions .....                     | 3552 |
| 20-12. Timer OCP Configuration Register Register Field Descriptions.....             | 3553 |
| 20-13. Timer IRQSTATUS Raw Register Register Field Descriptions.....                 | 3554 |
| 20-14. Timer IRQSTATUS Register Register Field Descriptions .....                    | 3555 |
| 20-15. Timer IRQENABLE Set Register Register Field Descriptions .....                | 3556 |
| 20-16. Timer IRQENABLE Clear Register Register Field Descriptions.....               | 3557 |
| 20-17. Timer IRQ Wakeup Enable Register Register Field Descriptions.....             | 3558 |
| 20-18. Timer Control Register Register Field Descriptions.....                       | 3559 |
| 20-19. Timer Counter Register Register Field Descriptions .....                      | 3561 |
| 20-20. Timer Load Register Register Field Descriptions .....                         | 3562 |
| 20-21. Timer Trigger Register Register Field Descriptions .....                      | 3563 |
| 20-22. Timer Write Posted Status Register Register Field Descriptions .....          | 3564 |
| 20-23. Timer Match Register Register Field Descriptions .....                        | 3565 |
| 20-24. Timer Capture Register Register Field Descriptions.....                       | 3566 |
| 20-25. Timer Synchronous Interface Control Register Register Field Descriptions..... | 3567 |
| 20-26. Timer Capture Register Register Field Descriptions.....                       | 3568 |
| 20-27. Timer1 Connectivity Attributes .....  | 3571 |
| 20-28. Timer Clock Signals .....   | 3572 |

|   |      |
|---|------|
| 20-29. Value Loaded in TCRR to Generate 1ms Tick .....      | 3574 |
| 20-30. Prescaler/Timer Reload Values Versus Contexts .....  | 3577 |
| 20-31. SmartIdle - Clock Activity Field Configuration ..... | 3579 |
| 20-32. Prescaler Clock Ratios Value .....                   | 3580 |
| 20-33. Value and Corresponding Interrupt Period .....       | 3581 |
| 20-34. DMTIMER_1MS REGISTERS .....                          | 3581 |
| 20-35. TIDR Register Field Descriptions .....               | 3583 |
| 20-36. TIOCP_CFG Register Field Descriptions .....          | 3584 |
| 20-37. TISTAT Register Field Descriptions .....             | 3585 |
| 20-38. TISR Register Field Descriptions .....               | 3586 |
| 20-39. TIER Register Field Descriptions .....               | 3587 |
| 20-40. TWER Register Field Descriptions .....               | 3588 |
| 20-41. TCLR Register Field Descriptions .....               | 3589 |
| 20-42. TCRR Register Field Descriptions .....               | 3591 |
| 20-43. TLDR Register Field Descriptions .....               | 3592 |
| 20-44. TTGR Register Field Descriptions .....               | 3593 |
| 20-45. TWPS Register Field Descriptions .....               | 3594 |
| 20-46. TMAR Register Field Descriptions .....               | 3596 |
| 20-47. TCAR1 Register Field Descriptions .....              | 3597 |
| 20-48. TSICR Register Field Descriptions .....              | 3598 |
| 20-49. TCAR2 Register Field Descriptions .....              | 3599 |
| 20-50. TPIR Register Field Descriptions .....               | 3600 |
| 20-51. TNIR Register Field Descriptions .....               | 3601 |
| 20-52. TCVR Register Field Descriptions .....               | 3602 |
| 20-53. TOCR Register Field Descriptions .....               | 3603 |
| 20-54. TOWR Register Field Descriptions .....               | 3604 |
| 20-55. RTC Module Connectivity Attributes .....             | 3606 |
| 20-56. RTC Clock Signals .....                              | 3606 |
| 20-57. RTC Pin List .....                                   | 3606 |
| 20-58. RTC Signals .....                                    | 3608 |
| 20-59. Interrupt Trigger Events .....                       | 3609 |
| 20-60. RTC Register Names and Values .....                  | 3611 |
| 20-61. pmic_power_en Description .....                      | 3614 |
| 20-62. RTC REGISTERS .....                                  | 3615 |
| 20-63. SECONDS_REG Register Field Descriptions .....        | 3617 |
| 20-64. MINUTES_REG Register Field Descriptions .....        | 3618 |
| 20-65. HOURS_REG Register Field Descriptions .....          | 3619 |
| 20-66. DAYS_REG Register Field Descriptions .....           | 3620 |
| 20-67. MONTHS_REG Register Field Descriptions .....         | 3621 |
| 20-68. YEARS_REG Register Field Descriptions .....          | 3622 |
| 20-69. WEEKS_REG Register Field Descriptions .....          | 3623 |
| 20-70. ALARM_SECONDS_REG Register Field Descriptions .....  | 3624 |
| 20-71. ALARM_MINUTES_REG Register Field Descriptions .....  | 3625 |
| 20-72. ALARM_HOURS_REG Register Field Descriptions .....    | 3626 |
| 20-73. ALARM_DAYS_REG Register Field Descriptions .....     | 3627 |
| 20-74. ALARM_MONTHS_REG Register Field Descriptions .....   | 3628 |
| 20-75. ALARM_YEARS_REG Register Field Descriptions .....    | 3629 |
| 20-76. RTC_CTRL_REG Register Field Descriptions .....       | 3630 |
| 20-77. RTC_STATUS_REG Register Field Descriptions .....     | 3632 |

|   |      |
|---|------|
| 20-78. RTC_INTERRUPTS_REG Register Field Descriptions ..... | 3633 |
| 20-79. RTC_COMP_LSB_REG Register Field Descriptions.....    | 3634 |
| 20-80. RTC_COMP_MSB_REG Register Field Descriptions .....   | 3635 |
| 20-81. RTC_OSC_REG Register Field Descriptions.....         | 3636 |
| 20-82. RTC_SCRATCH0_REG Register Field Descriptions .....   | 3637 |
| 20-83. RTC_SCRATCH1_REG Register Field Descriptions .....   | 3638 |
| 20-84. RTC_SCRATCH2_REG Register Field Descriptions .....   | 3639 |
| 20-85. KICK0R Register Field Descriptions .....             | 3640 |
| 20-86. KICK1R Register Field Descriptions .....             | 3641 |
| 20-87. RTC_REVISION Register Field Descriptions .....       | 3642 |
| 20-88. RTC_SYSCONFIG Register Field Descriptions .....      | 3643 |
| 20-89. RTC_IRQWAKEEN Register Field Descriptions .....      | 3644 |
| 20-90. ALARM2_SECONDS_REG Register Field Descriptions ..... | 3645 |
| 20-91. ALARM2_MINUTES_REG Register Field Descriptions ..... | 3646 |
| 20-92. ALARM2_HOURS_REG Register Field Descriptions .....   | 3647 |
| 20-93. ALARM2_DAYS_REG Register Field Descriptions .....    | 3648 |
| 20-94. ALARM2_MONTHS_REG Register Field Descriptions .....  | 3649 |
| 20-95. ALARM2_YEARS_REG Register Field Descriptions .....   | 3650 |
| 20-96. RTC_PMIC Register Field Descriptions .....           | 3651 |
| 20-97. RTC_DEBOUNCE Register Field Descriptions .....       | 3652 |
| 20-98. Public WD Timer Module Connectivity Attributes ..... | 3654 |
| 20-99. Public WD Timer Clock Signals .....                  | 3654 |
| 20-100. Watchdog Timer Events .....                         | 3655 |
| 20-101. Count and Prescaler Default Reset Values.....       | 3656 |
| 20-102. Prescaler Clock Ratio Values .....                  | 3657 |
| 20-103. Reset Period Examples .....                         | 3657 |
| 20-104. Default Watchdog Timer Reset Periods.....           | 3658 |
| 20-105. Global Initialization of Surrounding Modules .....  | 3661 |
| 20-106. Watchdog Timer Module Global Initialization.....    | 3661 |
| 20-107. Watchdog Timer Basic Configuration .....            | 3661 |
| 20-108. Disable the Watchdog Timer .....                    | 3662 |
| 20-109. Enable the Watchdog Timer .....                     | 3662 |
| 20-110. WATCHDOG_TIMER REGISTERS .....                      | 3662 |
| 20-111. WDT_WIDR Register Field Descriptions.....           | 3664 |
| 20-112. WDT_WDSC Register Field Descriptions.....           | 3665 |
| 20-113. WDT_WDST Register Field Descriptions .....          | 3666 |
| 20-114. WDT_WISR Register Field Descriptions .....          | 3667 |
| 20-115. WDT_WIER Register Field Descriptions.....           | 3668 |
| 20-116. WDT_WCLR Register Field Descriptions .....          | 3669 |
| 20-117. WDT_WCRR Register Field Descriptions .....          | 3670 |
| 20-118. WDT_WLDR Register Field Descriptions .....          | 3671 |
| 20-119. WDT_WTGR Register Field Descriptions.....           | 3672 |
| 20-120. WDT_WWPS Register Field Descriptions .....          | 3673 |
| 20-121. WDT_WDLY Register Field Descriptions .....          | 3674 |
| 20-122. WDT_WSPR Register Field Descriptions.....           | 3675 |
| 20-123. WDT_WIRQSTATRAW Register Field Descriptions .....   | 3676 |
| 20-124. WDT_WIRQSTAT Register Field Descriptions.....       | 3677 |
| 20-125. WDT_WIRQENSET Register Field Descriptions .....     | 3678 |
| 20-126. WDT_WIRQENCLR Register Field Descriptions .....     | 3679 |

|        |  |      |
|--------|--|------|
| 21-1.  | Unsupported I2C Features .....                           | 3681 |
| 21-2.  | I2C0 Connectivity Attributes.....                        | 3682 |
| 21-3.  | I2C(1–2) Connectivity Attributes .....                   | 3683 |
| 21-4.  | I2C Clock Signals.....                                   | 3683 |
| 21-5.  | I2C Pin List .....                                       | 3683 |
| 21-6.  | Signal Pads.....   | 3685 |
| 21-7.  | Reset State of I2C Signals .....                         | 3685 |
| 21-8.  | I2C REGISTERS .....                                      | 3698 |
| 21-9.  | I2C_REVNB_LO Register Field Descriptions.....            | 3699 |
| 21-10. | I2C_REVNB_HI Register Field Descriptions.....            | 3700 |
| 21-11. | I2C_SYSC Register Field Descriptions.....                | 3701 |
| 21-12. | I2C_IRQSTATUS_RAW Register Field Descriptions .....      | 3703 |
| 21-13. | I2C_IRQSTATUS Register Field Descriptions.....           | 3709 |
| 21-14. | I2C_IRQENABLE_SET Register Field Descriptions .....      | 3711 |
| 21-15. | I2C_IRQENABLE_CLR Register Field Descriptions.....       | 3713 |
| 21-16. | I2C_WE Register Field Descriptions .....                 | 3715 |
| 21-17. | I2C_DMARXENABLE_SET Register Field Descriptions .....    | 3718 |
| 21-18. | I2C_DMATXENABLE_SET Register Field Descriptions .....    | 3719 |
| 21-19. | I2C_DMARXENABLE_CLR Register Field Descriptions .....    | 3720 |
| 21-20. | I2C_DMATXENABLE_CLR Register Field Descriptions .....    | 3721 |
| 21-21. | I2C_DMARXWAKE_EN Register Field Descriptions .....       | 3722 |
| 21-22. | I2C_DMATXWAKE_EN Register Field Descriptions .....       | 3724 |
| 21-23. | I2C_SYSS Register Field Descriptions .....               | 3726 |
| 21-24. | I2C_BUF Register Field Descriptions.....                 | 3727 |
| 21-25. | I2C_CNT Register Field Descriptions.....                 | 3729 |
| 21-26. | I2C_DATA Register Field Descriptions .....               | 3730 |
| 21-27. | I2C_CON Register Field Descriptions .....                | 3731 |
| 21-28. | I2C_OA Register Field Descriptions .....                 | 3734 |
| 21-29. | I2C_SA Register Field Descriptions .....                 | 3735 |
| 21-30. | I2C_PSC Register Field Descriptions.....                 | 3736 |
| 21-31. | I2C_SCLL Register Field Descriptions .....               | 3737 |
| 21-32. | I2C_SCLH Register Field Descriptions .....               | 3738 |
| 21-33. | I2C_SYSTEST Register Field Descriptions .....            | 3739 |
| 21-34. | I2C_BUFSTAT Register Field Descriptions .....            | 3743 |
| 21-35. | I2C_OA1 Register Field Descriptions .....                | 3744 |
| 21-36. | I2C_OA2 Register Field Descriptions.....                 | 3745 |
| 21-37. | I2C_OA3 Register Field Descriptions .....                | 3746 |
| 21-38. | I2C_ACTOA Register Field Descriptions .....              | 3747 |
| 21-39. | I2C_SBLOCK Register Field Descriptions .....             | 3748 |
| 22-1.  | McASP Connectivity Attributes .....                      | 3753 |
| 22-2.  | McASP Clock Signals.....                                 | 3754 |
| 22-3.  | McASP Pin List .....                                     | 3754 |
| 22-4.  | Biphase-Mark Encoder .....                               | 3761 |
| 22-5.  | Preamble Codes .....                                     | 3762 |
| 22-6.  | McASP Interface Signals .....                            | 3769 |
| 22-7.  | Channel Status and User Data for Each DIT Block.....     | 3776 |
| 22-8.  | Transmit Bitstream Data Alignment .....                  | 3787 |
| 22-9.  | Receive Bitstream Data Alignment .....                   | 3789 |
| 22-10. | McASP Registers Accessed Through Configuration Bus ..... | 3808 |

|        |   |      |
|--------|---|------|
| 22-11. | McASP AFIFO Registers Accessed Through Peripheral Configuration Port .....          | 3809 |
| 22-12. | Revision Identification Register (REV) Field Descriptions .....                     | 3810 |
| 22-13. | Power Idle SYSCONFIG Register (PWRIDLESYSCONFIG) Field Descriptions.....            | 3811 |
| 22-14. | Pin Function Register (PFUNC) Field Descriptions .....                              | 3813 |
| 22-15. | Pin Direction Register (PDIR) Field Descriptions .....                              | 3815 |
| 22-16. | Pin Data Output Register (PDOUT) Field Descriptions .....                           | 3817 |
| 22-17. | Pin Data Input Register (PDIN) Field Descriptions.....                              | 3819 |
| 22-18. | Pin Data Set Register (PDSET) Field Descriptions .....                              | 3821 |
| 22-19. | Pin Data Clear Register (PDCLR) Field Descriptions .....                            | 3823 |
| 22-20. | Global Control Register (GBLCTL) Field Descriptions .....                           | 3824 |
| 22-21. | Audio Mute Control Register (AMUTE) Field Descriptions.....                         | 3826 |
| 22-22. | Digital Loopback Control Register (DLBCTL) Field Descriptions .....                 | 3828 |
| 22-23. | Digital Mode Control Register (DITCTL) Field Descriptions .....                     | 3829 |
| 22-24. | Receiver Global Control Register (RGBLCTL) Field Descriptions .....                 | 3830 |
| 22-25. | Receive Format Unit Bit Mask Register (RMASK) Field Descriptions.....               | 3831 |
| 22-26. | Receive Bit Stream Format Register (RFMT) Field Descriptions .....                  | 3832 |
| 22-27. | Receive Frame Sync Control Register (AFSRCTL) Field Descriptions .....              | 3834 |
| 22-28. | Receive Clock Control Register (ACLKRCTL) Field Descriptions .....                  | 3835 |
| 22-29. | Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions .....  | 3836 |
| 22-30. | Receive TDM Time Slot Register (RTDM) Field Descriptions.....                       | 3837 |
| 22-31. | Receiver Interrupt Control Register (RINTCTL) Field Descriptions.....               | 3838 |
| 22-32. | Receiver Status Register (RSTAT) Field Descriptions .....                           | 3839 |
| 22-33. | Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions .....            | 3840 |
| 22-34. | Receive Clock Check Control Register (RCLKCHK) Field Descriptions .....             | 3841 |
| 22-35. | Receiver DMA Event Control Register (REVTCTL) Field Descriptions .....              | 3842 |
| 22-36. | Transmitter Global Control Register (XGBLCTL) Field Descriptions .....              | 3843 |
| 22-37. | Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions .....             | 3844 |
| 22-38. | Transmit Bit Stream Format Register (XFMT) Field Descriptions.....                  | 3845 |
| 22-39. | Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions.....              | 3847 |
| 22-40. | Transmit Clock Control Register (ACLKXCTL) Field Descriptions.....                  | 3848 |
| 22-41. | Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions ..... | 3849 |
| 22-42. | Transmit TDM Time Slot Register (XTDM) Field Descriptions .....                     | 3850 |
| 22-43. | Transmitter Interrupt Control Register (XINTCTL) Field Descriptions .....           | 3851 |
| 22-44. | Transmitter Status Register (XSTAT) Field Descriptions .....                        | 3852 |
| 22-45. | Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions .....            | 3853 |
| 22-46. | Transmit Clock Check Control Register (XCLKCHK) Field Descriptions .....            | 3854 |
| 22-47. | Transmitter DMA Event Control Register (XEVCTL) Field Descriptions .....            | 3855 |
| 22-48. | Serializer Control Registers (SRCTL $n$ ) Field Descriptions.....                   | 3856 |
| 22-49. | Write FIFO Control Register (WFIFOCTL) Field Descriptions.....                      | 3859 |
| 22-50. | Write FIFO Status Register (WFIFOSTS) Field Descriptions.....                       | 3860 |
| 22-51. | Read FIFO Control Register (RFIFOCTL) Field Descriptions .....                      | 3861 |
| 22-52. | Read FIFO Status Register (RFIFOSTS) Field Descriptions .....                       | 3862 |
| 22-53. | McASP Registers Accessed Through Data Port .....                                    | 3862 |
| 23-1.  | DCAN Connectivity Attributes.....   | 3865 |
| 23-2.  | DCAN Clock Signals .....  | 3866 |
| 23-3.  | DCAN Pin List.....  | 3866 |
| 23-4.  | Initialization of a Transmit Object.....  | 3882 |
| 23-5.  | Initialization of a single Receive Object for Data Frames .....                     | 3882 |
| 23-6.  | Initialization of a Single Receive Object for Remote Frames .....                   | 3883 |

---

|  |      |
|--|------|
| 23-7. Parameters of the CAN Bit Time .....   | 3890 |
| 23-8. Structure of a Message Object .....  | 3900 |
| 23-9. Field Descriptions .....   | 3900 |
| 23-10. Message RAM addressing in Debug/Suspend and RDA Mode .....                                | 3902 |
| 23-11. Message RAM Representation in Debug/Suspend Mode .....                                    | 3903 |
| 23-12. Message RAM Representation in RAM Direct Access Mode .....                                | 3903 |
| 23-13. DCAN Control Register Summary Table .....   | 3905 |
| 23-14. CAN Control Register (DCAN CTL) Field Descriptions .....                                  | 3906 |
| 23-15. Error and Status Register (DCAN ES) Field Descriptions .....                              | 3908 |
| 23-16. Error Counter Register (DCAN ERRC) Field Descriptions .....                               | 3910 |
| 23-17. Bit Timing Register (DCAN BTR) Field Descriptions .....                                   | 3911 |
| 23-18. Interrupt Register (DCAN INT) Field Descriptions .....                                    | 3912 |
| 23-19. Test Register (DCAN TEST) Field Descriptions .....  | 3913 |
| 23-20. Parity Error Code Register (DCAN PERR) Field Descriptions .....                           | 3914 |
| 23-21. Auto-Bus-On Time Register (DCAN ABOTR) Field Descriptions .....                           | 3915 |
| 23-22. Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) .....                         | 3917 |
| 23-23. Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) Field Descriptions .....      | 3917 |
| 23-24. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) .....                                   | 3919 |
| 23-25. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) Field Descriptions .....                | 3919 |
| 23-26. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) .....                                   | 3921 |
| 23-27. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) Field Descriptions .....                | 3921 |
| 23-28. Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) .....                            | 3923 |
| 23-29. Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) Field Descriptions .....         | 3923 |
| 23-30. Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) .....                    | 3924 |
| 23-31. Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) Field Descriptions ..... | 3924 |
| 23-32. IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD) Field Descriptions .....             | 3926 |
| 23-33. IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK) Field Descriptions .....                | 3928 |
| 23-34. IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) Field Descriptions .....         | 3929 |
| 23-35. IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL) Field Descriptions .....   | 3931 |
| 23-36. IF3 Observation Register (DCAN IF3OBS) Field Descriptions .....                           | 3934 |
| 23-37. IF3 Mask Register (DCAN IF3MSK) Field Descriptions .....                                  | 3936 |
| 23-38. IF3 Arbitration Register (DCAN IF3ARB) Field Descriptions .....                           | 3937 |
| 23-39. IF3 Message Control Register (DCAN IF3MCTL) Field Descriptions .....                      | 3938 |
| 23-40. Update Enable Registers (DCAN IF3UPD12 to IF3UPD78) .....                                 | 3941 |
| 23-41. Update Enable Registers (DCAN IF3UPD12 to IF3UPD78) Field Descriptions .....              | 3941 |
| 23-42. CAN TX I/O Control Register (DCAN TIOC) Field Descriptions .....                          | 3942 |
| 23-43. CAN RX IO Control Register (DCAN RIOC) Field Descriptions .....                           | 3944 |
| 24-1. Unsupported McSPI Features .....   | 3947 |
| 24-2. McSPI Connectivity Attributes .....  | 3949 |
| 24-3. McSPI Clock Signals .....  | 3949 |
| 24-4. McSPI Pin List .....   | 3949 |
| 24-5. Phase and Polarity Combinations .....  | 3954 |
| 24-6. Chip Select ↔ Clock Edge Delay Depending on Configuration .....                            | 3964 |
| 24-7. CLKSPIO High/Low Time Computation .....  | 3966 |
| 24-8. Clock Granularity Examples .....   | 3966 |
| 24-9. FIFO Writes, Word Length Relationship .....  | 3967 |
| 24-10. SPI Registers .....   | 3985 |
| 24-11. McSPI Revision Register (MCSPI_REVISION) Field Descriptions .....                         | 3987 |
| 24-12. McSPI System Configuration Register (MCSPI_SYSCONFIG) Field Descriptions .....            | 3988 |

|  |      |
|--|------|
| 24-13. McSPI System Status Register (MCSPI_SYSSTATUS) Field Descriptions .....                               | 3989 |
| 24-14. McSPI Interrupt Status Register (MCSPI_IRQSTATUS) Field Descriptions .....                            | 3990 |
| 24-15. McSPI Interrupt Enable Register (MCSPI_IRQENABLE) Field Descriptions.....                             | 3993 |
| 24-16. McSPI System Register (MCSPI_SYST) Field Descriptions .....   | 3995 |
| 24-17. McSPI Module Control Register(MCSPI_MODULCTRL) Field Descriptions .....                               | 3997 |
| 24-18. McSPI Channel ( <i>i</i> ) Configuration Register (MCSPI_CH( <i>i</i> )CONF) Field Descriptions ..... | 3999 |
| 24-19. Data Lines Configurations .....   | 4002 |
| 24-20. McSPI Channel ( <i>i</i> ) Status Register (MCSPI_CH( <i>i</i> )STAT) Field Descriptions .....        | 4003 |
| 24-21. McSPI Channel ( <i>i</i> ) Control Register (MCSPI_CH( <i>i</i> )CTRL) Field Descriptions .....       | 4004 |
| 24-22. McSPI Channel ( <i>i</i> ) Transmit Register (MCSPI_TX( <i>i</i> ) Field Descriptions .....           | 4005 |
| 24-23. McSPI Channel ( <i>i</i> ) Receive Register (MCSPI_RX( <i>i</i> ) Field Descriptions .....            | 4005 |
| 24-24. McSPI Transfer Levels Register (MCSPI_XFERLEVEL) Field Descriptions .....                             | 4006 |
| 24-25. McSPI DMA Address Aligned FIFO Transmitter Register (MCSPI_DAFTX) Field Descriptions.....             | 4007 |
| 24-26. McSPI DMA Address Aligned FIFO Receiver Register (MCSPI_DAFRX) Field Descriptions .....               | 4008 |
| 25-1. GPIO0 Connectivity Attributes .....  | 4012 |
| 25-2. GPIO[1:3] Connectivity Attributes .....  | 4012 |
| 25-3. GPIO Clock Signals .....   | 4012 |
| 25-4. GPIO Pin List .....  | 4013 |
| 25-5. GPIO REGISTERS.....  | 4021 |
| 25-6. GPIO_REVISION Register Field Descriptions.....   | 4022 |
| 25-7. GPIO_SYSCONFIG Register Field Descriptions.....  | 4023 |
| 25-8. GPIO_IRQSTATUS_RAW_0 Register Field Descriptions .....   | 4024 |
| 25-9. GPIO_IRQSTATUS_RAW_1 Register Field Descriptions .....   | 4025 |
| 25-10. GPIO_IRQSTATUS_0 Register Field Descriptions .....  | 4026 |
| 25-11. GPIO_IRQSTATUS_1 Register Field Descriptions .....  | 4027 |
| 25-12. GPIO_IRQSTATUS_SET_0 Register Field Descriptions .....  | 4028 |
| 25-13. GPIO_IRQSTATUS_SET_1 Register Field Descriptions .....  | 4029 |
| 25-14. GPIO_IRQSTATUS_CLR_0 Register Field Descriptions .....  | 4030 |
| 25-15. GPIO_IRQSTATUS_CLR_1 Register Field Descriptions .....  | 4031 |
| 25-16. GPIO_IRQWAKEN_0 Register Field Descriptions .....   | 4032 |
| 25-17. GPIO_IRQWAKEN_1 Register Field Descriptions .....   | 4033 |
| 25-18. GPIO_SYSSTATUS Register Field Descriptions.....   | 4034 |
| 25-19. GPIO_CTRL Register Field Descriptions.....  | 4035 |
| 25-20. GPIO_OE Register Field Descriptions.....  | 4036 |
| 25-21. GPIO_DATAIN Register Field Descriptions.....  | 4037 |
| 25-22. GPIO_DATAOUT Register Field Descriptions.....   | 4038 |
| 25-23. GPIO_LEVELDETECT0 Register Field Descriptions .....   | 4039 |
| 25-24. GPIO_LEVELDETECT1 Register Field Descriptions .....   | 4040 |
| 25-25. GPIO_RISINGDETECT Register Field Descriptions .....   | 4041 |
| 25-26. GPIO_FALLINGDETECT Register Field Descriptions.....   | 4042 |
| 25-27. GPIO_DEBOUNCENABLE Register Field Descriptions .....  | 4043 |
| 25-28. GPIO_DEBOUNTINGTIME Register Field Descriptions .....   | 4044 |
| 25-29. GPIO_CLEARDATAOUT Register Field Descriptions.....  | 4045 |
| 25-30. GPIO_SETDATAOUT Register Field Descriptions .....   | 4046 |
| 26-1. ROM Exception Vectors .....  | 4051 |
| 26-2. Dead Loops .....   | 4051 |
| 26-3. RAM Exception Vectors .....  | 4052 |
| 26-4. Tracing Data .....   | 4053 |
| 26-5. Crystal Frequencies Supported.....   | 4054 |

---

|        |   |      |
|--------|---|------|
| 26-6.  | ROM Code Default Clock Settings .....                 | 4055 |
| 26-7.  | SYSBOOT Configuration Pins <sup>[4]</sup> .....       | 4058 |
| 26-8.  | XIP Timings Parameters.....                           | 4068 |
| 26-9.  | Pins Used for NOR Boot .....                          | 4069 |
| 26-10. | Special SYSBOOT Pins for NOR Boot .....               | 4069 |
| 26-11. | NAND Timings Parameters .....                         | 4071 |
| 26-12. | ONFI Parameters Page Description .....                | 4072 |
| 26-13. | Supported NAND Devices.....                           | 4072 |
| 26-14. | 4th NAND ID Data Byte.....                            | 4073 |
| 26-15. | Pins Used for NANDI2C Boot for I2C EEPROM Access..... | 4074 |
| 26-16. | NAND Geometry Information on I2C EEPROM .....         | 4074 |
| 26-17. | ECC Configuration for NAND Boot.....                  | 4074 |
| 26-18. | Pins Used for NAND Boot.....                          | 4079 |
| 26-19. | Configuration Header TOC Item.....                    | 4083 |
| 26-20. | Configuration Header Settings.....                    | 4083 |
| 26-21. | Master Boot Record Structure .....                    | 4085 |
| 26-22. | Partition Entry .....                                 | 4086 |
| 26-23. | Partition Types .....                                 | 4086 |
| 26-24. | FAT Boot Sector .....                                 | 4087 |
| 26-25. | FAT Directory Entry .....                             | 4091 |
| 26-26. | FAT Entry Description .....                           | 4092 |
| 26-27. | Pins Used for MMC0 Boot.....                          | 4092 |
| 26-28. | Pins Used for MMC1 Boot.....                          | 4092 |
| 26-29. | Pins Used for SPI Boot .....                          | 4093 |
| 26-30. | Blocks and Sectors Searched on Non-XIP Memories ..... | 4093 |
| 26-31. | Pins Used for EMAC Boot in MII Mode.....              | 4096 |
| 26-32. | Pins Used for EMAC Boot in RGMII Mode.....            | 4096 |
| 26-33. | Pins Used for EMAC Boot in RMII Mode.....             | 4096 |
| 26-34. | Ethernet PHY Mode Selection .....                     | 4096 |
| 26-35. | Pins Used for UART Boot .....                         | 4097 |
| 26-36. | Customized Descriptor Parameters.....                 | 4098 |
| 26-37. | Pins Used for USB Boot.....                           | 4099 |
| 26-38. | GP Device Image Format .....                          | 4100 |
| 26-39. | Booting Parameters Structure .....                    | 4100 |
| 26-40. | Tracing Vectors .....                                 | 4103 |
| A-1.   | Document Revision History .....                       | 4106 |

**Read This First****About This Manual**

This TRM has been designated *Preliminary* because the documentation is in the formative or design phase of development. Texas Instruments reserves the right to change this TRM without notice.

Revision History will not be provided for this TRM as long as it is in the *Preliminary* phase of development.

**Related Documentation From Texas Instruments**

For a complete listing of related documentation and development-support tools, visit [www.ti.com](http://www.ti.com).

Cortex is a trademark of ARM Limited.  
ARM is a registered trademark of ARM Limited.  
EtherCAT is a registered trademark of EtherCAT Technology Group.  
USSE is a trademark of Imagination Technologies Ltd..  
POWERVR is a registered trademark of Imagination Technologies Ltd..  
EtherNet/IP is a trademark of Open DeviceNet Vendor Association, Inc..

## Introduction

### 1.1 AM335x Family

#### 1.1.1 Device Features

This architecture is configured with different sets of features in different devices. This technical reference manual details all of the features available in current and future AM335x devices. Some features may not be available or supported in your particular device. The features supported across different AM335x devices are shown in [Table 1-1](#). For more information on the different packages, refer to your device-specific data manual.

**Table 1-1. Device Features**

| Subsystem/<br>Co-Processor/<br>Peripheral   | AM3352                      | AM3354                      | AM3356                                | AM3357                                | AM3358                                | AM3359                                |
|---|-----------------------------|-----------------------------|---------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| <a href="#">ARM MPU<br/>Subsystem</a>   | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Programmable<br/>Real-Time Unit<br/>and Industrial<br/>Communication<br/>Subsystem<br/>(PRU-ICSS)</a> | N                           | N                           | All features<br>excluding<br>EtherCAT | All features<br>including<br>EtherCAT | All features<br>excluding<br>EtherCAT | All features<br>including<br>EtherCAT |
| <a href="#">Graphics<br/>Accelerator<br/>(SGX)</a>  | N                           | Y                           | N                                     | N                                     | Y                                     | Y                                     |
| <a href="#">Memory Map</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Interrupts</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Memory<br/>Subsystem</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Power and Clock<br/>Management<br/>(PRCM)</a>   | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Control Module</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Interconnects</a>   | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Enhanced Direct<br/>Memory Access<br/>(EDMA)</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Touchscreen<br/>Controller</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">LCD Controller</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Ethernet<br/>Subsystem</a>  | ZCE: 1 port<br>ZCZ: 2 ports | ZCE: 1 port<br>ZCZ: 2 ports | ZCE: 1 port<br>ZCZ: 2 ports           | No ZCE<br>Available<br>ZCZ: 2 ports   | ZCE: 1 port<br>ZCZ: 2 ports           | No ZCE<br>Available<br>ZCZ: 2 ports   |
| <a href="#">Pulse-Width<br/>Modulation<br/>Subsystem</a>  | Y                           | Y                           | Y                                     | Y                                     | Y                                     | Y                                     |
| <a href="#">Universal Serial<br/>Bus (USB)</a>  | ZCE: 1 port<br>ZCZ: 2 ports | ZCE: 1 port<br>ZCZ: 2 ports | ZCE: 1 port<br>ZCZ: 2 ports           | No ZCE<br>Available<br>ZCZ: 2 ports   | ZCE: 1 port<br>ZCZ: 2 ports           | No ZCE<br>Available<br>ZCZ: 2 ports   |

**Table 1-1. Device Features (continued)**

| Subsystem/<br>Co-Processor/<br>Peripheral                                       | AM3352 | AM3354 | AM3356 | AM3357 | AM3358 | AM3359 |
|---|--------|--------|--------|--------|--------|--------|
| <a href="#">Interprocessor<br/>Communication</a>                                | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Multimedia Card<br/>(MMC)</a>                                       | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Universal<br/>Asynchronous<br/>Receiver/Transm<br/>itter (UART)</a> | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">DMTimer</a>   | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">DMTimer 1ms</a>   | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">RTCSS</a>   | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Watchdog</a>  | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">I2C</a>   | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Multichannel<br/>Audio Serial Port<br/>(McASP)</a>                  | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Controller Area<br/>Network (CAN)</a>                               | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Multichannel<br/>Serial Port<br/>Interface<br/>(McSPI)</a>          | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">General Purpose<br/>Input/Output<br/>(GPIO)</a>                     | Y      | Y      | Y      | Y      | Y      | Y      |
| <a href="#">Initialization</a>  | Y      | Y      | Y      | Y      | Y      | Y      |

### 1.1.2 Device Identification

Several registers help identify the type and available features of the device. [Table 1-2](#) and summarize these registers.

**Table 1-2. Device\_ID (Address 0x44E10600) Bit Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-28 | DEVREV   |       | Device revision<br>See AM335x errata for specific information on each revision.<br>Value is revision-dependent. |
| 27-12 | PARTNUM  |       | Device part number<br>0xB944  |
| 11-1  | MFGR     |       | Manufacturer's ID<br>0x017  |
| 0     | Reserved |       | Read always as 0<br>0x0   |

### 1.1.3 Feature Identification

The AM335x has many different feature sets available. The DEV\_FEATURE register in the control module identifies which features are available in each device in the AM335x family. See [Table 1-3, dev\\_feature Register](#), in the control module chapter for more information on the bits in the DEV\_FEATURE register.

**Table 1-3. DEV\_FEATURE (Address 0x44E10604) Register Values**

| Bit            | 3<br>1                               | 3<br>0      | 29       | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 1<br>7           | 1<br>6           | 15       | 14 | 13 | 12 | 11 | 10 | 9                                    | 8                | 7        | 6 | 5 | 4 | 3 | 2 | 1                | 0                | Register Value |
|----------------|--------------------------------------|-------------|----------|----|----|----|----|----|----|----|----|----|----|----|------------------|------------------|----------|----|----|----|----|----|--------------------------------------|------------------|----------|---|---|---|---|---|------------------|------------------|----------------|
| Bit Definition | R<br>e<br>s<br>e<br>r<br>v<br>e<br>d | S<br>G<br>X | Reserved |    |    |    |    |    |    |    |    |    |    |    | I<br>C<br>S<br>S | —<br>F<br>E<br>A | Reserved |    |    |    |    |    | R<br>e<br>s<br>e<br>r<br>v<br>e<br>d | D<br>C<br>A<br>N | Reserved |   |   |   |   |   | C<br>P<br>S<br>W | I<br>C<br>S<br>S |                |
| AM3352         | 0                                    | 0           | 0        | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0                | 0                | 0        | 0  | 0  | 0  | 0  | 1  | 1                                    | 1                | 0        | 0 | 0 | 0 | 1 | 0 | 0x00FC0382       |                  |                |
| AM3354         | 0                                    | 0           | 1        | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 0                | 0                | 0        | 0  | 0  | 0  | 0  | 1  | 1                                    | 1                | 0        | 0 | 0 | 0 | 1 | 0 | 0x20FC0382       |                  |                |
| AM3356         | 0                                    | 0           | 0        | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 0                | 1                | 0        | 0  | 0  | 0  | 0  | 1  | 1                                    | 1                | 0        | 0 | 0 | 0 | 1 | 1 | 0x00FD0383       |                  |                |
| AM3357         | 0                                    | 0           | 0        | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1                | 1                | 0        | 0  | 0  | 0  | 0  | 0  | 1                                    | 1                | 1        | 0 | 0 | 0 | 1 | 1 | 0x00FF0383       |                  |                |
| AM3358         | 0                                    | 0           | 1        | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 0                | 1                | 0        | 0  | 0  | 0  | 0  | 0  | 1                                    | 1                | 1        | 0 | 0 | 0 | 0 | 1 | 1                | 0x20FD0383       |                |
| AM3359         | 0                                    | 0           | 1        | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1                | 1                | 0        | 0  | 0  | 0  | 0  | 0  | 1                                    | 1                | 1        | 0 | 0 | 0 | 0 | 1 | 1                | 0x20FF0383       |                |

## Memory Map

This section describes the memory map for the device.

### 2.1 ARM Cortex-A8 Memory Map

**Table 2-1. L3 Memory Map**

| Block Name                | Start_address (hex) | End_address (hex) | Size   | Description  |
|---------------------------|---------------------|-------------------|--------|--|
| GPMC<br>(External Memory) | 0x0000_0000         | 0x1FFF_FFFF       | 512MB  | 8-/16-bit External Memory<br>(Ex/R/W) <sup>(1)</sup> |
| Reserved                  | 0x2000_0000         | 0x3FFF_FFFF       | 512MB  | Reserved   |
| Boot ROM                  | 0x4000_0000         | 0x4001_FFFF       | 128KB  |  |
|                           | 0x4002_0000         | 0x4002_BFFF       | 48KB   | 32-bit Ex/R <sup>(1)</sup> – Public                  |
| Reserved                  | 0x4002_C000         | 0x400F_FFFF       | 848KB  | Reserved   |
| Reserved                  | 0x4010_0000         | 0x401F_FFFF       | 1MB    | Reserved   |
| Reserved                  | 0x4020_0000         | 0x402E_FFFF       | 960KB  | Reserved   |
| Reserved                  | 0x402f_0000         | 0x4020_03FF       | 64KB   | Reserved   |
| SRAM internal             | 0x402F_0400         | 0x402F_FFFF       |        | 32-bit Ex/R/W <sup>(1)</sup>                         |
| L3 OCMC0                  | 0x4030_0000         | 0x4030_FFFF       | 64KB   | 32-bit Ex/R/W <sup>(1)</sup> OCMC SRAM               |
| Reserved                  | 0x4031_0000         | 0x403F_FFFF       | 960KB  | Reserved   |
| Reserved                  | 0x4040_0000         | 0x4041_FFFF       | 128KB  | Reserved   |
| Reserved                  | 0x4042_0000         | 0x404F_FFFF       | 896KB  | Reserved   |
| Reserved                  | 0x4050_0000         | 0x405F_FFFF       | 1MB    | Reserved   |
| Reserved                  | 0x4060_0000         | 0x407F_FFFF       | 2MB    | Reserved   |
| Reserved                  | 0x4080_0000         | 0x4083_FFFF       | 256KB  | Reserved   |
| Reserved                  | 0x4084_0000         | 0x40DF_FFFF       | 5888KB | Reserved   |
| Reserved                  | 0x40E0_0000         | 0x40E0_7FFF       | 32KB   | Reserved   |
| Reserved                  | 0x40E0_8000         | 0x40EF_FFFF       | 992KB  | Reserved   |
| Reserved                  | 0x40F0_0000         | 0x40F0_7FFF       | 32KB   | Reserved   |
| Reserved                  | 0x40F0_8000         | 0x40FF_FFFF       | 992KB  | Reserved   |
| Reserved                  | 0x4100_0000         | 0x41FF_FFFF       | 16MB   | Reserved   |
| Reserved                  | 0x4200_0000         | 0x43FF_FFFF       | 32MB   | Reserved   |
| L3F CFG Regs              | 0x4400_0000         | 0x443F_FFFF       | 4MB    | L3Fast configuration registers                       |
| Reserved                  | 0x4440_0000         | 0x447F_FFFF       | 4MB    | Reserved   |
| L3S CFG Regs              | 0x4480_0000         | 0x44BF_FFFF       | 4MB    | L3Slow configuration registers                       |
| <u>L4_WKUP</u>            | 0x44C0_0000         | 0x44FF_FFFF       | 4MB    | L4_WKUP  |
| Reserved                  | 0x4500_0000         | 0x45FF_FFFF       | 16MB   | Reserved   |
| <u>McASP0 Data</u>        | 0x4600_0000         | 0x463F_FFFF       | 4MB    | McASP0 Data Registers                                |
| <u>McASP1 Data</u>        | 0x4640_0000         | 0x467F_FFFF       | 4MB    | McASP1 Data Registers                                |
| Reserved                  | 0x4680_0000         | 0x46FF_FFFF       | 8MB    | Reserved   |
| Reserved                  | 0x4700_0000         | 0x473F_FFFF       | 4MB    | Reserved   |

<sup>(1)</sup> Ex/R/W – Execute/Read/Write.

**Table 2-1. L3 Memory Map (continued)**

| Block Name                                     | Start_address (hex) | End_address (hex) | Size      | Description                           |
|--|---------------------|-------------------|-----------|---------------------------------------|
| <a href="#"><u>USBSS</u></a>                   | 0x4740_0000         | 0x4740_0FFF       | 20KB      | USB Subsystem Registers               |
| <a href="#"><u>USB0</u></a>                    | 0x4740_1000         | 0x4740_12FF       |           | USB0 Controller Registers             |
| <a href="#"><u>USB0_PHY</u></a>                | 0x4740_1300         | 0x4740_13FF       |           | USB0 PHY Registers                    |
| USB0 Core                                      | 0x4740_1400         | 0x4740_17FF       |           | USB0 Core Registers                   |
| <a href="#"><u>USB1</u></a>                    | 0x4740_1800         | 0x4740_1AFF       |           | USB1 Controller Registers             |
| <a href="#"><u>USB1_PHY</u></a>                | 0x4740_1B00         | 0x4740_1BFF       |           | USB1 PHY Registers                    |
| USB1 Core                                      | 0x4740_1C00         | 0x4740_1FFF       |           | USB1 Core Registers                   |
| <a href="#"><u>USB CPPI DMA Controller</u></a> | 0x4740_2000         | 0x4740_2FFF       |           | USB CPPI DMA Controller Registers     |
| <a href="#"><u>USB CPPI DMA Scheduler</u></a>  | 0x4740_3000         | 0x4740_3FFF       |           | USB CPPI DMA Scheduler Registers      |
| <a href="#"><u>USB Queue Manager</u></a>       | 0x4740_4000         | 0x4740_4FFF       |           | USB Queue Manager Registers           |
| Reserved                                       | 0x4740_5000         | 0x477F_FFFF       | 4MB-20KB  | Reserved                              |
| Reserved                                       | 0x4780_0000         | 0x4780_FFFF       | 64KB      | Reserved                              |
| <a href="#"><u>MMCHS2</u></a>                  | 0x4781_0000         | 0x4781_FFFF       | 64KB      | MMCHS2                                |
| Reserved                                       | 0x4782_0000         | 0x47BF_FFFF       | 4MB-128KB | Reserved                              |
| Reserved                                       | 0x47C0_0000         | 0x47FF_FFFF       | 4MB       | Reserved                              |
| <a href="#"><u>L4_PER</u></a>                  | 0x4800_0000         | 0x48FF_FFFF       | 16MB      | L4 Peripheral (see L4_PER table)      |
| <a href="#"><u>TPCC</u> (EDMA3CC)</a>          | 0x4900_0000         | 0x490F_FFFF       | 1MB       | EDMA3 Channel Controller Registers    |
| Reserved                                       | 0x4910_0000         | 0x497F_FFFF       | 7MB       | Reserved                              |
| <a href="#"><u>TPTC0</u> (EDMA3TC0)</a>        | 0x4980_0000         | 0x498F_FFFF       | 1MB       | EDMA3 Transfer Controller 0 Registers |
| <a href="#"><u>TPTC1</u> (EDMA3TC1)</a>        | 0x4990_0000         | 0x499F_FFFF       | 1MB       | EDMA3 Transfer Controller 1 Registers |
| <a href="#"><u>TPTC2</u> (EDMA3TC2)</a>        | 0x49A0_0000         | 0x49AF_FFFF       | 1MB       | EDMA3 Transfer Controller 2 Registers |
| Reserved                                       | 0x49B0_0000         | 0x49BF_FFFF       | 1MB       | Reserved                              |
| Reserved                                       | 0x49C0_0000         | 0x49FF_FFFF       | 4MB       | Reserved                              |
| <a href="#"><u>L4_FAST</u></a>                 | 0x4A00_0000         | 0x4AFF_FFFF       | 16MB      | L4_FAST                               |
| DebugSS  | 0x4B00_0000         | 0x4BFF_FFFF       | 16MB      | Debug Subsystem region                |
| <a href="#"><u>EMIF0</u></a>                   | 0x4C00_0000         | 0x4CFF_FFFF       | 16MB      | EMIF0 Configuration registers         |
| Reserved                                       | 0x4D00_0000         | 0x4DFF_FFFF       | 16MB      | Reserved                              |
| Reserved                                       | 0x4E00_0000         | 0x4FFF_FFFF       | 32MB      | Reserved                              |
| <a href="#"><u>GPMC</u></a>                    | 0x5000_0000         | 0x50FF_FFFF       | 16MB      | GPMC Configuration registers          |
| Reserved                                       | 0x5100_0000         | 0x52FF_FFFF       | 32MB      | Reserved                              |
| Reserved                                       | 0x5300_0000         | 0x530F_FFFF       | 1MB       | Reserved                              |
|  | 0x5310_0000         | 0x531F_FFFF       | 1MB       | Reserved                              |
| Reserved                                       | 0x5320_0000         | 0x533F_FFFF       | 2MB       | Reserved                              |
| Reserved                                       | 0x5340_0000         | 0x534F_FFFF       | 1MB       | Reserved                              |
|  | 0x5350_0000         | 0x535F_FFFF       | 1MB       | Reserved                              |
| Reserved                                       | 0x5360_0000         | 0x54BF_FFFF       | 22MB      | Reserved                              |
| ADC_TSC DMA                                    | 0x54C0_0000         | 0x54FF_FFFF       | 4MB       | ADC_TSC DMA Port                      |
| Reserved                                       | 0x5500_0000         | 0x55FF_FFFF       | 16MB      | Reserved                              |
| SGX530   | 0x5600_0000         | 0x56FF_FFFF       | 16MB      | SGX530 Slave Port                     |
| Reserved                                       | 0x5700_0000         | 0x57FF_FFFF       | 16MB      | Reserved                              |
| Reserved                                       | 0x5800_0000         | 0x58FF_FFFF       | 16MB      | Reserved                              |
| Reserved                                       | 0x5900_0000         | 0x59FF_FFFF       | 16MB      | Reserved                              |
| Reserved                                       | 0x5A00_0000         | 0x5AFF_FFFF       | 16MB      | Reserved                              |

**Table 2-1. L3 Memory Map (continued)**

| Block Name  | Start_address (hex) | End_address (hex) | Size  | Description                                       |
|-------------|---------------------|-------------------|-------|---|
| Reserved    | 0x5B00_0000         | 0x5BFF_FFFF       | 16MB  | Reserved  |
| Reserved    | 0x5C00_0000         | 0x5DFF_FFFF       | 32MB  | Reserved  |
| Reserved    | 0x5E00_0000         | 0x5FFF_FFFF       | 32MB  | Reserved  |
| Reserved    | 0x6000_0000         | 0x7FFF_FFFF       | 512MB | Reserved  |
| EMIF0 SDRAM | 0x8000_0000         | 0xBFFF_FFFF       | 1GB   | 8-/16-bit External Memory (Ex/R/W) <sup>(2)</sup> |
| Reserved    | 0xC000_0000         | 0xFFFF_FFFF       | 1GB   | Reserved  |

<sup>(2)</sup> Ex/R/W – Execute/Read/Write.

**Table 2-2. L4\_WKUP Peripheral Memory Map**

| Region Name                | Start Address (hex) | End Address (hex) | Size      | Description  |
|----------------------------|---------------------|-------------------|-----------|--|
| L4_WKUP configuration      | 0x44C0_0000         | 0x44C0_07FF       | 2KB       | Address/Protection (AP)  |
|                            | 0x44C0_0800         | 0x44C0_0FFF       | 2KB       | Link Agent (LA)  |
|                            | 0x44C0_1000         | 0x44C0_13FF       | 1KB       | Initiator Port (IP0)   |
|                            | 0x44C0_1400         | 0x44C0_17FF       | 1KB       | Initiator Port (IP1)   |
| Reserved                   | 0x44C0_1800         | 0x44C0_1FFF       | 2KB       | Reserved (IP2 – IP3)   |
| Reserved                   | 0x44C0_2000         | 0x44CF_FFFF       | 1MB-8KB   | Reserved   |
| M3 UMEM                    | 0x44D0_0000         | 0x44D0_3FFF       | 16KB      | M3 Unified Code Space (M3 Icode & Dcode buses can access this RAM) |
|                            | 0x44D0_4000         | 0x44D0_4FFF       | 4KB       | Reserved   |
| Reserved                   | 0x44D0_5000         | 0x44D7_FFFF       | 492KB     | Reserved   |
| M3 DMEM                    | 0x44D8_0000         | 0x44D8_1FFF       | 8KB       | M3 Data Memory   |
|                            | 0x44D8_2000         | 0x44D8_2FFF       | 4KB       | Reserved   |
| Reserved                   | 0x44D8_3000         | 0x44DF_FFFF       | 500KB     | Reserved   |
| <a href="#">CM_PER</a>     | 0x44E0_0000         | 0x44E0_3FFF       | 1KB       | Clock Module Peripheral Registers                                  |
| <a href="#">CM_WKUP</a>    | 0x44E0_0400         | 0x44E0_04FF       | 256 Bytes | Clock Module Wakeup Registers                                      |
| <a href="#">CM_DPLL</a>    | 0x44E0_0500         | 0x44E0_05FF       | 256 Bytes | Clock Module PLL Registers   |
| <a href="#">CM_MPU</a>     | 0x44E0_0600         | 0x44E0_06FF       | 256 Bytes | Clock Module MPU Registers   |
| <a href="#">CM_DEVICE</a>  | 0x44E0_0700         | 0x44E0_07FF       | 256 Bytes | Clock Module Device Registers                                      |
| <a href="#">CM_RTC</a>     | 0x44E0_0800         | 0x44E0_08FF       | 256 Bytes | Clock Module RTC Registers   |
| <a href="#">CM_GFX</a>     | 0x44E0_0900         | 0x44E0_09FF       | 256 Bytes | Clock Module Graphics Controller Registers                         |
| <a href="#">CM_CEFUSE</a>  | 0x44E0_0A00         | 0x44E0_0AFF       | 256 Bytes | Clock Module Efuse Registers                                       |
| <a href="#">PRM_IRQ</a>    | 0x44E0_0B00         | 0x44E0_0BFF       | 256 Bytes | Power Reset Module Interrupt Registers                             |
| <a href="#">PRM_PER</a>    | 0x44E0_0C00         | 0x44E0_0CFF       | 256 Bytes | Power Reset Module Peripheral Registers                            |
| <a href="#">PRM_WKUP</a>   | 0x44E0_0D00         | 0x44E0_0DFF       | 256 Bytes | Power Reset Module Wakeup Registers                                |
| <a href="#">PRM_MPU</a>    | 0x44E0_0E00         | 0x44E0_0EFF       | 256 Bytes | Power Reset Module MPU Registers                                   |
| <a href="#">PRM_DEVICE</a> | 0x44E0_0F00         | 0x44E0_0FFF       | 256 Bytes | Power Reset Module Device Registers                                |
| <a href="#">PRM_RTC</a>    | 0x44E0_1000         | 0x44E0_10FF       | 256 Bytes | Power Reset Module RTC Registers                                   |
| <a href="#">PRM_GFX</a>    | 0x44E0_1100         | 0x44E0_11FF       | 256 Bytes | Power Reset Module Graphics Controller Registers                   |
| <a href="#">PRM_CEFUSE</a> | 0x44E0_1200         | 0x44E0_12FF       | 256 Bytes | Power Reset Module Efuse Registers                                 |
| Reserved                   | 0x44E0_3000         | 0x44E0_3FFF       | 4KB       | Reserved   |

**Table 2-2. L4\_WKUP Peripheral Memory Map (continued)**

| Region Name                                  | Start Address (hex) | End Address (hex) | Size  | Description               |
|--|---------------------|-------------------|-------|---------------------------|
|  | 0x44E0_4000         | 0x44E0_4FFF       | 4KB   | Reserved                  |
| <u>DMTIMER0</u>                              | 0x44E0_5000         | 0x44E0_5FFF       | 4KB   | DMTimer0 Registers        |
|  | 0x44E0_6000         | 0x44E0_6FFF       | 4KB   | Reserved                  |
| <u>GPIO0</u>                                 | 0x44E0_7000         | 0x44E0_7FFF       | 4KB   | GPIO Registers            |
|  | 0x44E0_8000         | 0x44E0_8FFF       | 4KB   | Reserved                  |
| <u>UART0</u>                                 | 0x44E0_9000         | 0x44E0_9FFF       | 4KB   | UART Registers            |
|  | 0x44E0_A000         | 0x44E0_AFFF       | 4KB   | Reserved                  |
| <u>I2C0</u>                                  | 0x44E0_B000         | 0x44E0_BFFF       | 4KB   | I2C Registers             |
|  | 0x44E0_C000         | 0x44E0_CFFF       | 4KB   | Reserved                  |
| <u>ADC_TSC</u>                               | 0x44E0_D000         | 0x44E0_EFFF       | 8KB   | ADC_TSC Registers         |
|  | 0x44E0_F000         | 0x44E0_FFFF       | 4KB   | Reserved                  |
| <u>Control Module</u>                        | 0x44E1_0000         | 0x44E1_1FFF       | 128KB | Control Module Registers  |
| <u>DDR2/3/mDDR PHY</u>                       | 0x44E1_2000         | 0x44E1_23FF       |       | DDR2/3/mDDR PHY Registers |
| Reserved                                     | 0x44E1_2400         | 0x44E3_0FFF       | 4KB   | Reserved                  |
| <u>DMTIMER1_1MS</u><br>(Accurate 1ms timer)  | 0x44E3_1000         | 0x44E3_1FFF       | 4KB   | DMTimer1 1ms Registers    |
|  | 0x44E3_2000         | 0x44E3_2FFF       | 4KB   | Reserved                  |
| Reserved                                     | 0x44E3_3000         | 0x44E3_3FFF       | 4KB   | Reserved                  |
|  | 0x44E3_4000         | 0x44E3_4FFF       | 4KB   | Reserved                  |
| <u>WDT1</u>                                  | 0x44E3_5000         | 0x44E3_5FFF       | 4KB   | Watchdog Timer Registers  |
|  | 0x44E3_6000         | 0x44E3_6FFF       | 4KB   | Reserved                  |
| SmartReflex0                                 | 0x44E3_7000         | 0x44E3_7FFF       | 4KB   | L3 Registers              |
|  | 0x44E3_8000         | 0x44E3_8FFF       | 4KB   | Reserved                  |
| SmartReflex1                                 | 0x44E3_9000         | 0x44E3_9FFF       | 4KB   | L3 Registers              |
|  | 0x44E3_A000         | 0x44E3_AFFF       | 4KB   | Reserved                  |
| Reserved                                     | 0x44E3_B000         | 0x44E3_DFFF       | 12KB  | Reserved                  |
| <u>RTCSS</u>                                 | 0x44E3_E000         | 0x44E3_EFFF       | 4KB   | RTC Registers             |
|  | 0x44E3_F000         | 0x44E3_FFFF       | 4KB   | Reserved                  |
| DebugSS<br>Instrumentation<br>HWMaster1 Port | 0x44E4_0000         | 0x44E7_FFFF       | 256KB | Debug Registers           |
|  | 0x44E8_0000         | 0x44E8_0FFF       | 4KB   | Reserved                  |
| Reserved                                     | 0x44E8_1000         | 0x44EF_FFFF       | 508KB | Reserved                  |
| Reserved                                     | 0x44F0_0000         | 0x44FF_FFFF       | 1MB   | Reserved                  |

**Table 2-3. L4\_PER Peripheral Memory Map**

| Device Name | Start_address (hex) | End_address (hex) | Size | Description |
|-------------|---------------------|-------------------|------|-------------|
| Reserved    | 0x4800_0000         | 0x4800_07FF       | 2KB  | Reserved    |
|             | 0x4800_0800         | 0x4800_0FFF       | 2KB  | Reserved    |
|             | 0x4800_1000         | 0x4800_13FF       | 1KB  | Reserved    |
|             | 0x4800_1400         | 0x4800_17FF       | 1KB  | Reserved    |
|             | 0x4800_1800         | 0x4800_1BFF       | 1KB  | Reserved    |
|             | 0x4800_1C00         | 0x4800_1FFF       | 1KB  | Reserved    |
| Reserved    | 0x4800_2000         | 0x4800_3FFF       | 8KB  | Reserved    |
| Reserved    | 0x4800_4000         | 0x4800_7FFF       | 16KB | Reserved    |
| Reserved    | 0x4800_8000         | 0x4800_8FFF       | 4KB  | Reserved    |
|             | 0x4800_9000         | 0x4800_9FFF       | 4KB  | Reserved    |
| Reserved    | 0x4800_A000         | 0x4800_FFFF       | 24KB | Reserved    |

**Table 2-3. L4\_PER Peripheral Memory Map (continued)**

| <b>Device Name</b> | <b>Start_address (hex)</b> | <b>End_address (hex)</b> | <b>Size</b> | <b>Description</b>   |
|--------------------|----------------------------|--------------------------|-------------|----------------------|
| Reserved           | 0x4801_0000                | 0x4801_0FFF              | 4KB         | Reserved             |
|                    | 0x4801_1000                | 0x4801_1FFF              | 4KB         | Reserved             |
| Reserved           | 0x4801_2000                | 0x4801_3FFF              | 8KB         | Reserved             |
| Reserved           | 0x4801_4000                | 0x4801_FFFF              | 48KB        | Reserved             |
| Reserved           | 0x4802_0000                | 0x4802_0FFF              | 4KB         | Reserved             |
|                    | 0x4802_1000                | 0x4802_1FFF              | 4KB         | Reserved             |
| <u>UART1</u>       | 0x4802_2000                | 0x4802_2FFF              | 4KB         | UART1 Registers      |
|                    | 0x4802_3000                | 0x4802_3FFF              | 4KB         | Reserved             |
| <u>UART2</u>       | 0x4802_4000                | 0x4802_4FFF              | 4KB         | UART2 Registers      |
|                    | 0x4802_5000                | 0x4802_5FFF              | 4KB         | Reserved             |
| Reserved           | 0x4802_6000                | 0x4802_7FFF              | 8KB         | Reserved             |
| Reserved           | 0x4802_8000                | 0x4802_8FFF              | 4KB         | Reserved             |
|                    | 0x4802_9000                | 0x4802_9FFF              | 4KB         | Reserved             |
| <u>I2C1</u>        | 0x4802_A000                | 0x4802_AFFF              | 4KB         | I2C1 Registers       |
|                    | 0x4802_B000                | 0x4802_BFFF              | 4KB         | Reserved             |
| Reserved           | 0x4802_C000                | 0x4802_CFFF              | 4KB         | Reserved             |
|                    | 0x4802_D000                | 0x4802_DFFF              | 4KB         | Reserved             |
| Reserved           | 0x4802_E000                | 0x4802_EFFF              | 4KB         | Reserved             |
|                    | 0x4802_F000                | 0x4802_FFFF              | 4KB         | Reserved             |
| <u>McSPI0</u>      | 0x4803_0000                | 0x4803_0FFF              | 4KB         | McSPI0 Registers     |
|                    | 0x4803_1000                | 0x4803_1FFF              | 4KB         | Reserved             |
| Reserved           | 0x4803_2000                | 0x4803_2FFF              | 4KB         | Reserved             |
|                    | 0x4803_3000                | 0x4803_3FFF              | 4KB         | Reserved             |
| Reserved           | 0x4803_4000                | 0x4803_4FFF              | 4KB         | Reserved             |
|                    | 0x4803_5000                | 0x4803_5FFF              | 4KB         | Reserved             |
| Reserved           | 0x4803_6000                | 0x4803_6FFF              | 4KB         | Reserved             |
|                    | 0x4803_7000                | 0x4803_7FFF              | 4KB         | Reserved             |
| <u>McASP0 CFG</u>  | 0x4803_8000                | 0x4803_9FFF              | 8KB         | McASP0 CFG Registers |
|                    | 0x4803_A000                | 0x4803_AFFF              | 4KB         | Reserved             |
| Reserved           | 0x4803_B000                | 0x4803_BFFF              | 4KB         | Reserved             |
| <u>McASP1 CFG</u>  | 0x4803_C000                | 0x4803_DFFF              | 8KB         | McASP1 CFG Registers |
|                    | 0x4803_E000                | 0x4803_EFFF              | 4KB         | Reserved             |
| Reserved           | 0x4803_F000                | 0x4803_FFFF              | 4KB         | Reserved             |
| <u>DMTIMER2</u>    | 0x4804_0000                | 0x4804_0FFF              | 4KB         | DMTimer2 Registers   |
|                    | 0x4804_1000                | 0x4804_1FFF              | 4KB         | Reserved             |
| <u>DMTIMER3</u>    | 0x4804_2000                | 0x4804_2FFF              | 4KB         | DMTimer3 Registers   |
|                    | 0x4804_3000                | 0x4804_3FFF              | 4KB         | Reserved             |
| <u>DMTIMER4</u>    | 0x4804_4000                | 0x4804_4FFF              | 4KB         | DMTimer4 Registers   |
|                    | 0x4804_5000                | 0x4804_5FFF              | 4KB         | Reserved             |
| <u>DMTIMER5</u>    | 0x4804_6000                | 0x4804_6FFF              | 4KB         | DMTimer5 Registers   |
|                    | 0x4804_7000                | 0x4804_7FFF              | 4KB         | Reserved             |
| <u>DMTIMER6</u>    | 0x4804_8000                | 0x4804_8FFF              | 4KB         | DMTimer6 Registers   |
|                    | 0x4804_9000                | 0x4804_9FFF              | 4KB         | L4 Interconnect      |
| <u>DMTIMER7</u>    | 0x4804_A000                | 0x4804_AFFF              | 4KB         | DMTimer7 Registers   |
|                    | 0x4804_B000                | 0x4804_BFFF              | 4KB         | Reserved             |
| <u>GPIO1</u>       | 0x4804_C000                | 0x4804_CFFF              | 4KB         | GPIO1 Registers      |
|                    | 0x4804_D000                | 0x4804_DFFF              | 4KB         | Reserved             |

**Table 2-3. L4\_PER Peripheral Memory Map (continued)**

| <b>Device Name</b> | <b>Start_address (hex)</b> | <b>End_address (hex)</b> | <b>Size</b> | <b>Description</b>       |
|--------------------|----------------------------|--------------------------|-------------|--------------------------|
| Reserved           | 0x4804_E000                | 0x4804_FFFF              | 8KB         | Reserved                 |
| Reserved           | 0x4805_0000                | 0x4805_FFFF              | 64KB        | Reserved                 |
| <u>MMCHS0</u>      | 0x4806_0000                | 0x4806_0FFF              | 4KB         | MMCHS0 Registers         |
|                    | 0x4806_1000                | 0x4806_1FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4806_2000                | 0x4807_FFFF              | 120KB       | Reserved                 |
| <u>ELM</u>         | 0x4808_0000                | 0x4808_FFFF              | 64KB        | ELM Registers            |
|                    | 0x4809_0000                | 0x4809_0FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4809_1000                | 0x4809_FFFF              | 60KB        | Reserved                 |
| Reserved           | 0x480A_0000                | 0x480A_FFFF              | 64KB        | Reserved                 |
|                    | 0x480B_0000                | 0x480B_0FFF              | 4KB         | Reserved                 |
| Reserved           | 0x480B_1000                | 0x480B_FFFF              | 60KB        | Reserved                 |
| Reserved           | 0x480C_0000                | 0x480C_0FFF              | 4KB         | Reserved                 |
|                    | 0x480C_1000                | 0x480C_1FFF              | 4KB         | Reserved                 |
| Reserved           | 0x480C_2000                | 0x480C_2FFF              | 4KB         | Reserved                 |
|                    | 0x480C_3000                | 0x480C_3FFF              | 4KB         | Reserved                 |
| Reserved           | 0x480C_4000                | 0x480C_7FFF              | 16KB        | Reserved                 |
| <u>Mailbox 0</u>   | 0x480C_8000                | 0x480C_8FFF              | 4KB         | Mailbox Registers        |
|                    | 0x480C_9000                | 0x480C_9FFF              | 4KB         | Reserved                 |
| <u>Spinlock</u>    | 0x480C_A000                | 0x480C_AFFF              | 4KB         | Spinlock Registers       |
|                    | 0x480C_B000                | 0x480C_BFFF              | 4KB         | Reserved                 |
| Reserved           | 0x480C_C000                | 0x480F_FFFF              | 208KB       | Reserved                 |
| Reserved           | 0x4810_0000                | 0x4811_FFFF              | 128KB       | Reserved                 |
|                    | 0x4812_0000                | 0x4812_0FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4812_1000                | 0x4812_1FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4812_2000                | 0x4812_2FFF              | 4KB         | Reserved                 |
|                    | 0x4812_3000                | 0x4812_3FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4812_4000                | 0x4813_FFFF              | 112KB       | Reserved                 |
| Reserved           | 0x4814_0000                | 0x4815_FFFF              | 128KB       | Reserved                 |
|                    | 0x4816_0000                | 0x4816_0FFF              | 4K          | Reserved                 |
| Reserved           | 0x4816_1000                | 0x4817_FFFF              | 124KB       | Reserved                 |
| Reserved           | 0x4818_0000                | 0x4818_2FFF              | 12KB        | Reserved                 |
|                    | 0x4818_3000                | 0x4818_3FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4818_4000                | 0x4818_7FFF              | 16KB        | Reserved                 |
| Reserved           | 0x4818_8000                | 0x4818_8FFF              | 4KB         | Reserved                 |
|                    | 0x4818_9000                | 0x4818_9FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4818_A000                | 0x4818_AFFF              | 4KB         | Reserved                 |
|                    | 0x4818_B000                | 0x4818_BFFF              | 4KB         | Reserved                 |
| OCP Watchpoint     | 0x4818_C000                | 0x4818_CFFF              | 4KB         | OCP Watchpoint Registers |
|                    | 0x4818_D000                | 0x4818_DFFF              | 4KB         | Reserved                 |
| Reserved           | 0x4818_E000                | 0x4818_EFFF              | 4KB         | Reserved                 |
|                    | 0x4818_F000                | 0x4818_FFFF              | 4KB         | Reserved                 |
| Reserved           | 0x4819_0000                | 0x4819_0FFF              | 4KB         | Reserved                 |
|                    | 0x4819_1000                | 0x4819_1FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4819_2000                | 0x4819_2FFF              | 4KB         | Reserved                 |
|                    | 0x4819_3000                | 0x4819_3FFF              | 4KB         | Reserved                 |
| Reserved           | 0x4819_4000                | 0x4819_BFFF              | 32KB        | Reserved                 |
| <u>I2C2</u>        | 0x4819_C000                | 0x4819_CFFF              | 4KB         | I2C2 Registers           |

**Table 2-3. L4\_PER Peripheral Memory Map (continued)**

| Device Name                          | Start_address (hex) | End_address (hex) | Size  | Description                        |
|--------------------------------------|---------------------|-------------------|-------|------------------------------------|
| Reserved                             | 0x4819_D000         | 0x4819_DFFF       | 4KB   | Reserved                           |
|                                      | 0x4819_E000         | 0x4819_EFFF       | 4KB   | Reserved                           |
|                                      | 0x4819_F000         | 0x4819_FFFF       | 4KB   | Reserved                           |
| <u>McSPI1</u>                        | 0x481A_0000         | 0x481A_0FFF       | 4KB   | McSPI1 Registers                   |
|                                      | 0x481A_1000         | 0x481A_1FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481A_2000         | 0x481A_5FFF       | 16KB  | Reserved                           |
| <u>UART3</u>                         | 0x481A_6000         | 0x481A_6FFF       | 4KB   | UART3 Registers                    |
|                                      | 0x481A_7000         | 0x481A_7FFF       | 4KB   | Reserved                           |
| <u>UART4</u>                         | 0x481A_8000         | 0x481A_8FFF       | 4KB   | UART4 Registers                    |
|                                      | 0x481A_9000         | 0x481A_9FFF       | 4KB   | Reserved                           |
| <u>UART5</u>                         | 0x481A_A000         | 0x481A_AFFF       | 4KB   | UART5 Registers                    |
|                                      | 0x481A_B000         | 0x481A_BFFF       | 4KB   | Reserved                           |
| <u>GPIO2</u>                         | 0x481A_C000         | 0x481A_CFFF       | 4KB   | GPIO2 Registers                    |
|                                      | 0x481A_D000         | 0x481A_DFFF       | 4KB   | Reserved                           |
| <u>GPIO3</u>                         | 0x481A_E000         | 0x481A_EFFF       | 4KB   | GPIO3 Registers                    |
|                                      | 0x481A_F000         | 0x481A_FFFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481B_0000         | 0x481B_FFFF       | 64KB  | Reserved                           |
|                                      | 0x481C_0000         | 0x481C_0FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481C_1000         | 0x481C_1FFF       | 4KB   | Reserved                           |
|                                      | 0x481C_2000         | 0x481C_2FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481C_3000         | 0x481C_9FFF       | 28KB  | Reserved                           |
| Reserved                             | 0x481C_A000         | 0x481C_AFFF       | 4KB   | Reserved                           |
|                                      | 0x481C_B000         | 0x481C_BFFF       | 4KB   | Reserved                           |
| <u>DCAN0</u>                         | 0x481C_C000         | 0x481C_DFFF       | 8KB   | DCAN0 Registers                    |
|                                      | 0x481C_E000         | 0x481C_FFFF       | 8KB   | Reserved                           |
| <u>DCAN1</u>                         | 0x481D_0000         | 0x481D_1FFF       | 8KB   | DCAN1 Registers                    |
|                                      | 0x481D_2000         | 0x481D_3FFF       | 8KB   | Reserved                           |
| Reserved                             | 0x481D_4000         | 0x481D_4FFF       | 4KB   | Reserved                           |
|                                      | 0x481D_5000         | 0x481D_5FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481D_6000         | 0x481D_6FFF       | 4KB   | Reserved                           |
|                                      | 0x481D_7000         | 0x481D_7FFF       | 4KB   | Reserved                           |
| <u>MMC1</u>                          | 0x481D_8000         | 0x481D_8FFF       | 4KB   | MMC1 Registers                     |
|                                      | 0x481D_9000         | 0x481D_9FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x481D_A000         | 0x481F_FFFF       | 152KB | Reserved                           |
| <u>Interrupt controller (INTCPs)</u> | 0x4820_0000         | 0x4820_0FFF       | 4KB   | Interrupt Controller Registers     |
| Reserved                             | 0x4820_1000         | 0x4823_FFFF       | 252KB | Reserved                           |
| MPUSS config register                | 0x4824_0000         | 0x4824_0FFF       | 4KB   | Host ARM non-shared device mapping |
| Reserved                             | 0x4824_1000         | 0x4827_FFFF       | 252KB | Reserved                           |
| Reserved                             | 0x4828_0000         | 0x4828_0FFF       | 4KB   | Reserved                           |
| Reserved                             | 0x4828_1000         | 0x482F_FFFF       | 508KB | Reserved                           |
| <u>PWM Subsystem 0</u>               | 0x4830_0000         | 0x4830_0FFF       | 4KB   | PWMSS0 Configuration Registers     |
| <u>eCAP0</u>                         | 0x4830_0100         | 0x4830_017F       |       | PWMSS eCAP0 Registers              |
| <u>eQEP0</u>                         | 0x4830_0180         | 0x4830_01FF       |       | PWMSS eQEP0 Registers              |
| <u>ePWM0</u>                         | 0x4830_0200         | 0x4830_025F       |       | PWMSS ePWM0 Registers              |
|                                      | 0x4830_0260         | 0x4830_1FFF       | 4KB   | Reserved                           |

**Table 2-3. L4\_PER Peripheral Memory Map (continued)**

| Device Name            | Start_address (hex) | End_address (hex) | Size       | Description                    |
|------------------------|---------------------|-------------------|------------|--------------------------------|
| <u>PWM Subsystem 1</u> | 0x4830_2000         | 0x4830_20FF       | 4KB        | PWMSS1 Configuration Registers |
| <u>eCAP1</u>           | 0x4830_2100         | 0x4830_217F       |            | PWMSS eCAP1 Registers          |
| <u>eQEP1</u>           | 0x4830_2180         | 0x4830_21FF       |            | PWMSS eQEP1 Registers          |
| <u>ePWM1</u>           | 0x4830_2200         | 0x4830_225F       |            | PWMSS ePWM1 Registers          |
|                        | 0x4830_2260         | 0x4830_3FFF       |            | Reserved                       |
| <u>PWM Subsystem 2</u> | 0x4830_4000         | 0x4830_40FF       | 4KB        | PWMSS2 Configuration Registers |
| <u>eCAP2</u>           | 0x4830_4100         | 0x4830_417F       |            | PWMSS eCAP2 Registers          |
| <u>eQEP2</u>           | 0x4830_4180         | 0x4830_41FF       |            | PWMSS eQEP2 Registers          |
| <u>ePWM2</u>           | 0x4830_4200         | 0x4830_425F       |            | PWMSS ePWM2 Registers          |
|                        | 0x4830_4260         | 0x4830_5FFF       | 4KB        | Reserved                       |
| Reserved               | 0x4830_6000         | 0x4830_DFFF       | 32KB       | Reserved                       |
| <u>LCD Controller</u>  | 0x4830_E000         | 0x4830_EFFF       | 4KB        | LCD Registers                  |
|                        | 0x4830_F000         | 0x4830_FFFF       | 4KB        | Reserved                       |
| Reserved               | 0x4831_0000         | 0x4831_1FFF       | 8KB        | Reserved                       |
|                        | 0x4831_2000         | 0x4831_2FFF       | 4KB        | Reserved                       |
| Reserved               | 0x4831_3000         | 0x4831_7FFF       | 20KB       | Reserved                       |
| Reserved               | 0x4831_8000         | 0x4831_BFFF       | 16KB       | Reserved                       |
|                        | 0x4831_C000         | 0x4831_CFFF       | 4KB        | Reserved                       |
| Reserved               | 0x4831_D000         | 0x4831_FFFF       | 12KB       | Reserved                       |
| Reserved               | 0x4832_0000         | 0x4832_5FFF       | 16KB       | Reserved                       |
| Reserved               | 0x4832_6000         | 0x48FF_FFFF       | 13MB-152KB | Reserved                       |

**Table 2-4. L4 Fast Peripheral Memory Map**

| Device Name           | Start_address (hex) | End_address (hex) | Size  | Description                    |
|-----------------------|---------------------|-------------------|-------|--------------------------------|
| L4_Fast configuration | 0x4A00_0000         | 0x4A00_07FF       | 2KB   | Address/Protection (AP)        |
|                       | 0x4A00_0800         | 0x4A00_0FFF       | 2KB   | Link Agent (LA)                |
|                       | 0x4A00_1000         | 0x4A00_13FF       | 1KB   | Initiator Port (IP0)           |
|                       | 0x4A00_1400         | 0x4A00_17FF       | 1KB   | Reserved                       |
|                       | 0x4A00_1800         | 0x4A00_1FFF       | 2KB   | Reserved (IP2 – IP3)           |
| Reserved              | 0x4A00_2000         | 0x4A07_FFFF       | 504KB | Reserved                       |
| Reserved              | 0x4A08_0000         | 0x4A09_FFFF       | 128KB | Reserved                       |
|                       | 0x4A0A_0000         | 0x4A0A_0FFF       | 4KB   | Reserved                       |
| Reserved              | 0x4A0A_1000         | 0x4A0F_FFFF       | 380KB | Reserved                       |
| <u>CPSW_SS</u>        | 0x4A10_0000         | 0x4A10_7FFF       | 32KB  | Ethernet Switch Subsystem      |
| <u>CPSW_PORT</u>      | 0x4A10_0100         | 0x4A10_07FF       |       | Ethernet Switch Port Control   |
| <u>CPSW_CPDMA</u>     | 0x4A10_0800         | 0x4A10_08FF       |       | CPPI DMA Controller Module     |
| <u>CPSW_STATS</u>     | 0x4A10_0900         | 0x4A10_09FF       |       | Ethernet Statistics            |
| <u>CPSW_STATERAM</u>  | 0x4A10_0A00         | 0x4A10_0BFF       |       | CPPI DMA State RAM             |
| <u>CPSW_CPTS</u>      | 0x4A10_0C00         | 0x4A10_0CFF       |       | Ethernet Time Sync Module      |
| <u>CPSW_ALE</u>       | 0x4A10_0D00         | 0x4A10_0D7F       |       | Ethernet Address Lookup Engine |
| <u>CPSW_SL1</u>       | 0x4A10_0D80         | 0x4A10_0DBF       |       | Ethernet Sliver for Port 1     |
| <u>CPSW_SL2</u>       | 0x4A10_0DC0         | 0x4A10_0DFF       |       | Ethernet Sliver for Port 2     |
| Reserved              | 0x4A10_0E00         | 0x4A10_0FFF       |       | Reserved                       |

**Table 2-4. L4 Fast Peripheral Memory Map (continued)**

| <b>Device Name</b> | <b>Start_address (hex)</b> | <b>End_address (hex)</b> | <b>Size</b> | <b>Description</b>                            |
|--------------------|----------------------------|--------------------------|-------------|---|
| <u>MDIO</u>        | 0x4A10_1000                | 0x4A10_10FF              |             | Ethernet MDIO Controller                      |
| Reserved           | 0x4A10_1100                | 0x4A10_11FF              |             | Reserved                                      |
| <u>CPSW_WR</u>     | 0x4A10_1200                | 0x4A10_1FFF              |             | Ethernet Subsystem Wrapper for RMII/RGMII     |
| CPPI_RAM           | 0x4A10_2000                | 0x4A10_3FFF              |             | Communications Port Programming Interface RAM |
| Reserved           | 0x4A10_9000                | 0x4A13_FFFF              | 220KB       | Reserved                                      |
| Reserved           | 0x4A14_0000                | 0x4A14_FFFF              | 64KB        | Reserved                                      |
|                    | 0x4A15_0000                | 0x4A15_0FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A15_1000                | 0x4A17_FFFF              | 188KB       | Reserved                                      |
| Reserved           | 0x4A18_0000                | 0x4A1A_1FFF              | 136KB       | Reserved                                      |
| Reserved           | 0x4A1A_2000                | 0x4A1A_3FFF              | 8KB         | Reserved                                      |
|                    | 0x4A1A_4000                | 0x4A1A_4FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_5000                | 0x4A1A_5FFF              | 4KB         | Reserved                                      |
|                    | 0x4A1A_6000                | 0x4A1A_6FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_7000                | 0x4A1A_7FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_8000                | 0x4A1A_9FFF              | 8KB         | Reserved                                      |
|                    | 0x4A1A_A000                | 0x4A1A_AFFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_B000                | 0x4A1A_BFFF              | 4KB         | Reserved                                      |
|                    | 0x4A1A_C000                | 0x4A1A_CFFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_D000                | 0x4A1A_DFFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1A_E000                | 0x4A1A_FFFF              | 8KB         | Reserved                                      |
|                    | 0x4A1B_0000                | 0x4A1B_0FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1B_1000                | 0x4A1B_1FFF              | 4KB         | Reserved                                      |
|                    | 0x4A1B_2000                | 0x4A1B_2FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1B_3000                | 0x4A1B_3FFF              | 4KB         | Reserved                                      |
|                    | 0x4A1B_4000                | 0x4A1B_4FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1B_5000                | 0x4A1B_5FFF              | 4KB         | Reserved                                      |
|                    | 0x4A1B_6000                | 0x4A1B_6FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A1B_4000                | 0x4A1F_FFFF              | 304KB       | Reserved                                      |
| Reserved           | 0x4A20_0000                | 0x4A2F_FFFF              | 1MB         | Reserved                                      |
| PRU_ICSS           | 0x4A30_0000                | 0x4A37_FFFF              | 512KB       | PRU-ICSS Instruction/Data/Control Space       |
|                    | 0x4A38_0000                | 0x4A38_0FFF              | 4KB         | Reserved                                      |
| Reserved           | 0x4A38_1000                | 0x4A3F_FFFF              | 508KB       | Reserved                                      |
| Reserved           | 0x4A40_0000                | 0x4AFF_FFFF              | 12MB        | Reserved                                      |

## 2.2 ARM Cortex-M3 Memory Map

**Table 2-5. M3 Processor Memory Map**

| Region Name                                 | Start Address (hex) | End Address (hex) | Size       | Description  |
|---|---------------------|-------------------|------------|--|
| M3 Unified Memory (UMEM)                    | 0x0000_0000         | 0x0000_3FFF       | 16 KB      | ICODE & DCODE unified code space (ICode, DCode Buses are muxed into single Memory: UMEM) |
| Reserved                                    | 0x0000_4000         | 0x0007_FFFF       | 512KB-16KB | Reserved   |
| M3 DMEM                                     | 0x0008_0000         | 0x0008_1FFF       | 8KB        | M3 Data Memory   |
| Reserved                                    | 0x0008_2000         | 0x000F_FFFF       | 512KB-8KB  | Reserved   |
| Reserved                                    | 0x0010_0000         | 0x1FFF_FFFF       | 511MB      | Reserved   |
| M3 Unified Memory <sup>(1)</sup> (UMEM)     | 0x2000_0000         | 0x2000_3FFF       | 16KB       | Through L4_WKUP (using System Bus) (SRAM Bit Band Region)                                |
| Reserved                                    | 0x2000_4000         | 0x2007_FFFF       | 512KB-16KB | Reserved (SRAM Bit Band Region)  |
| M3 DMEM <sup>(1)</sup>                      | 0x2008_0000         | 0x2008_1FFF       | 8KB        | Through L4_WKUP (using System Bus) (SRAM Bit Band Region)                                |
| Reserved <sup>(2)</sup>                     | 0x2008_2000         | 0x200F_FFFF       | 512KB-8KB  | Reserved (SRAM Bit Band Region)  |
| Reserved <sup>(2)</sup>                     | 0x2010_0000         | 0x21FF_FFFF       | 31MB       | Reserved (SRAM Region)   |
| Bit-Band Alias Region <sup>(3)</sup>        | 0x2200_0000         | 0x23FF_FFFF       | 32MB       | Bit-Band Alias Region  |
| Reserved <sup>(2)</sup>                     | 0x2400_0000         | 0x3FFF_FFFF       | 448MB      | Reserved (SRAM Region)   |
| Reserved <sup>(2)</sup>                     | 0x4000_0000         | 0x400F_FFFF       | 1MB        | Reserved (Periph Bit Band Region)  |
| Reserved <sup>(2)</sup>                     | 0x4010_0000         | 0x41FF_FFFF       | 31MB       | Reserved (Periph Region)   |
| Reserved <sup>(2)</sup> (3)                 | 0x4200_0000         | 0x43FF_FFFF       | 32MB       | Reserved (Periph Bit-Band Alias)   |
| Reserved <sup>(2)</sup>                     | 0x4400_0000         | 0x44BF_FFFF       | 12MB       | Reserved (Peripheral Region)   |
| L4_WKUP <sup>(4)</sup> (1)                  | 0x44C0_0000         | 0x44FF_FFFF       | 4MB        | L4 Wake Up Domain  |
| Reserved <sup>(2)</sup>                     | 0x4500_0000         | 0x5FFF_FFFF       | 432MB      | Reserved (Peripheral Region)   |
| Reserved <sup>(2)</sup>                     | 0x6000_0000         | 0xDFFF_FFFF       | 2GB        | Reserved (External RAM/Device)   |
| Reserved (ITM)                              | 0xE000_0000         | 0xE000_0FFF       | 4KB        | Instrumentation Trace Macrocell  |
| Reserved (DWT)                              | 0xE000_1000         | 0xE000_1FFF       | 4KB        | Data Watchpoint Trace  |
| Reserved (FPB)                              | 0xE000_2000         | 0xE000_2FFF       | 4KB        | Flashpatch and Breakpoint  |
| Reserved (Private Peripheral Bus -Internal) | 0xE000_3000         | 0xE000_DFFF       | 44KB       | Reserved (Private Peripheral Bus -Internal)  |
| NVIC  | 0xE000_E000         | 0xE000_EFFF       | 4KB        | Nested Vector Interrupt Controller   |
| Reserved                                    | 0xE000_F000         | 0xE003_FFFF       | 196KB      | Reserved   |

<sup>(1)</sup> The Wakeup M3 may access the UMEM and DMEM using the L4\_WKUP (OCP Bus) or it may use its direct buses, by accessing address 0x00000000 – 0x00081FFF.

<sup>(2)</sup> These addresses correspond to L3 regions. These are reserved regions with respect to Wakeup M3.

<sup>(3)</sup> Accessing these bit-band alias regions results in an access to the corresponding bit-band region.

<sup>(4)</sup> Peripherals located on L4 Wakeup Bus can be accessed by Wakeup M3.

**Table 2-5. M3 Processor Memory Map (continued)**

| Region Name           | Start Address (hex) | End Address (hex) | Size  | Description           |
|-----------------------|---------------------|-------------------|-------|-----------------------|
| Reserved              | 0xE004_0000         | 0xE004_0FFF       | 4KB   | Reserved              |
| Reserved              | 0xE004_1000         | 0xE004_1FFF       | 4KB   | Reserved              |
| Reserved              | 0xE004_2000         | 0xE00F_EFFF       | 756KB | Reserved              |
| ROM table (registers) | 0xE00F_F000         | 0xE00F_FFFF       | 4KB   | ROM table (registers) |
| Reserved              | 0xE010_0000         | 0xFFFF_FFFF       | 511MB | Reserved              |

## ARM MPU Subsystem

This chapter describes the MPU Subsystem for the device.

| Topic  | Page       |
|--|------------|
| <b>3.1 ARM Cortex-A8 MPU Subsystem .....</b> | <b>162</b> |

### 3.1 ARM Cortex-A8 MPU Subsystem

The Microprocessor Unit (MPU) subsystem of the device handles transactions between the ARM core (ARM® Cortex™-A8 Processor), the L3 interconnect, and the interrupt controller (INTC). The MPU subsystem is a hard macro that integrates the ARM® Cortex™-A8 Processor with additional logic for protocol conversion, emulation, interrupt handling, and debug enhancements.

Cortex™-A8 is an ARMv7 compatible, dual-issue, in-order execution engine with integrated L1 and L2 caches with NEON™ SIMD Media Processing Unit.

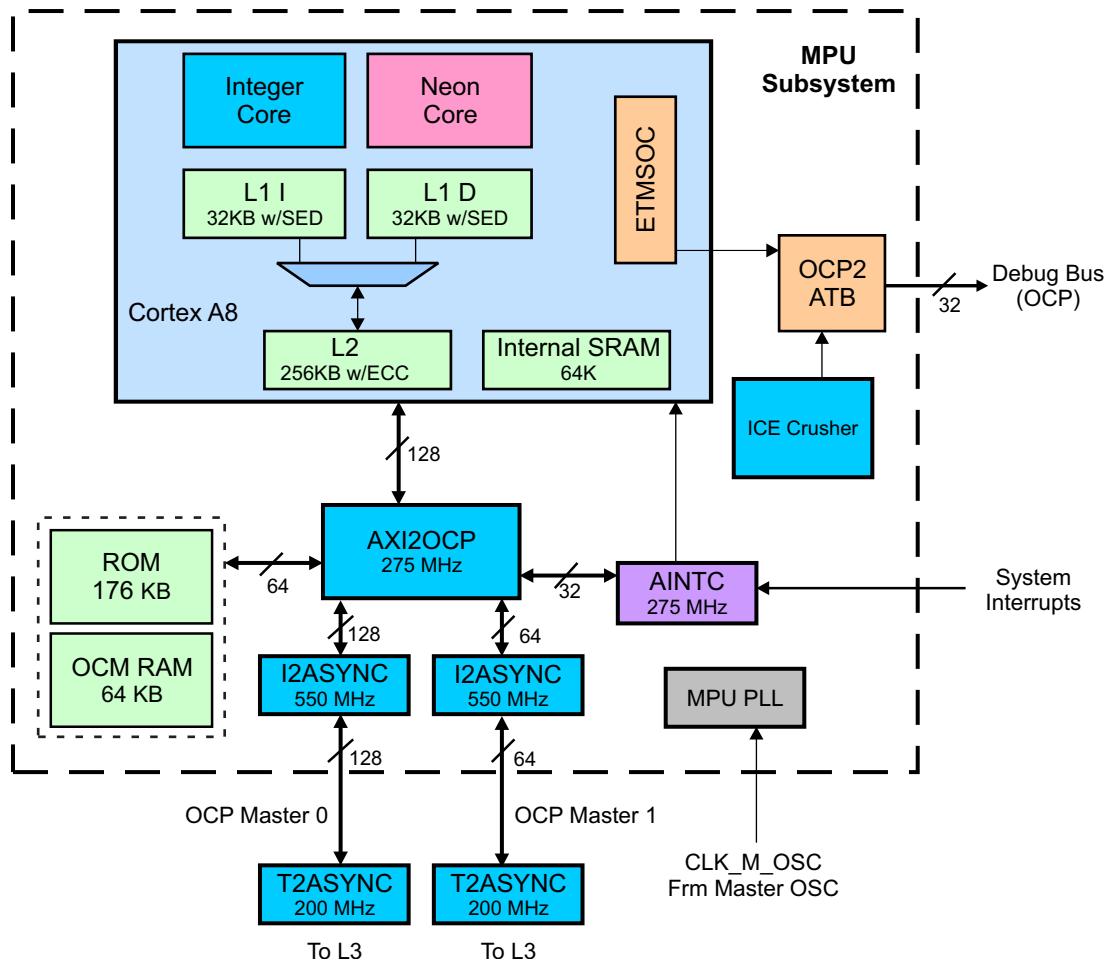
An Interrupt Controller is included in the MPU subsystem to handle host interrupt requests in the system.

The MPU subsystem includes CoreSight compliant logic to allow the Debug Sub-system access to the CortexA8 debug and emulation resources, including the Embedded Trace Macrocell.

The MPU subsystem has three functional clock domains, including a high-frequency clock domain used by the Cortex™-A8. The high-frequency domain is isolated from the rest of the system by asynchronous bridges.

Figure 3-1 shows the high-level block diagram of the MPU subsystem.

**Figure 3-1. Microprocessor Unit (MPU) Subsystem**



### 3.1.1 Features

This section outlines the key features of the MPU subsystem:

- ARM Microprocessor
  - CortexA8
  - ARM Architecture version 7 ISA.
  - 2-issue, in-order execution pipeline.
  - L1 and L2 Instruction and Data Cache of 32 KB , 4-way, 16 word line with 128 bit interface.
  - Integrated L2 cache of 256 KB, 8-way, 16 word line, 128 bit interface to L1 along with ECC/Parity supported.
  - Includes the Neon Media coprocessor (NEON™) which implements the Advanced SIMD media processing architecture.
  - Includes the VFP coprocessor which implements the VFPv3 architecture and is fully compliant with IEEE 754 standard.
  - The external interface uses the AXI protocol configured to 128-bit data width.
  - Includes the Embedded Trace Macrocell (ETM) support for non-invasive debugging.
  - Implements the ARMv7 debug with watch-point and breakpoint registers and 32-bit Advanced Peripheral Bus (APB) slave interface to CoreSight debug systems.
- AXI2OCP Bridge
  - Support OCP 2.2.
  - Single Request Multiple Data Protocol on two ports.
  - Multiple targets, including three OCP ports (128-bit, 64-bit and 32-bit).
- Interrupt Controller
  - Support up to 128 interrupt requests
- Emulation/Debug
  - Compatible with CoreSight Architecture.
- Clock Generation
  - Through PRCM
- DFT
  - Integrated PBIST controller to test L2 tag and data ram, L1I and L1D data ram and OCM RAM.

### 3.1.2 MPU Subsystem Integration

The MPU subsystem integrates the following group of submodules:

**ARM® Cortex™-A8 Processor:** Provides a high processing capability, including the NEON technology for mobile multimedia acceleration. The ARM communicates through an AXI bus with the AXI2OCP bridge and receives interrupts from the MPU subsystem interrupt controller (MPU INTC).

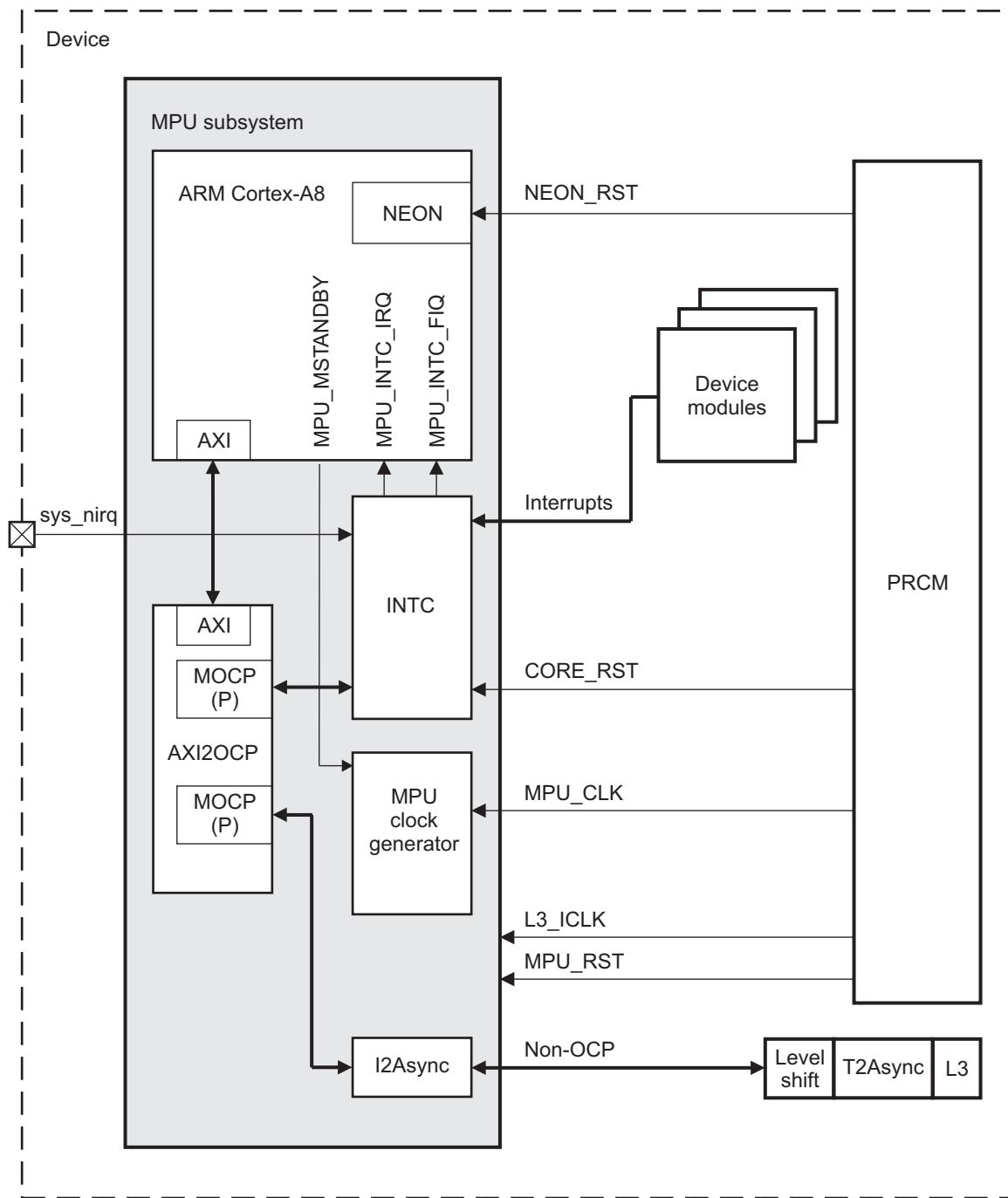
**Interrupt controller:** Handles the module interrupts (for details, see [Chapter 6, Interrupts](#)).

**AXI2OCP bridge:** Allows communication between the ARM (AXI), the INTC (OCP), and the modules (OCP L3).

**I2Async bridge:** This is an asynchronous bridge interface providing an asynchronous OCP to OCP interface. This interface is between the AXI2OCP bridge within the MPU subsystem and the T2Async bridge external to the MPU subsystem.

**Clock Divider:** Provides the required divided clocks to the internal modules of the MPU subsystem and has a clock input from SYSCLK2 which is fed by the power, reset, and clock management (PRCM) module of the device.

**In-Circuit Emulator:** It is fully Compatible with CoreSight Architecture and enables debugging capabilities.



**Figure 3-2. Microprocessor Unit (MPU) Subsystem Signal Interface**

### 3.1.3 MPU Subsystem Clock and Reset Distribution

#### 3.1.3.1 Clock Distribution

The MPU subsystem includes an embedded DPLL which sources the clock for the ARM Cortex-A8 processor. A clock divider within the subsystem is used for deriving the clocks for other internal modules.

All major modules inside the MPU subsystem are clocked at half the frequency of the ARM core. The divider of the output clock can be programmed with the PRCM.CM\_CLKSEL2\_PLL\_MPUMPU\_DPLL\_CLKOUT\_DIV register field, the frequency is relative to the ARM core. For details see the Power, Reset, and Clock Management (PRCM) chapter.

The clock generator generates the following functional clocks:

**ARM (ARM\_FCLK):** This is the core clock. It is the base fast clock that is routed internally to the ARM logic and internal RAMs, including NEON, L2 cache, the ETM core (emulation), and the ARM core.

**AXI2OCP Clock (AXI\_FCLK):** This clock is half the frequency of the ARM clock (ARM\_FCLK). The OCP interface thus performs at one half the frequency of ARM.

**Interrupt Controller Functional Clock (MPU\_INTC\_FCLK):** This clock, which is part of the INTC module, is half the frequency of the ARM clock (ARM\_FCLK).

**ICE-Crusher Functional Clock (ICECRUSHER\_FCLK):** ICE-Crusher clocking operates on the APB interface, using the ARM core clocking. This clock is half the frequency of the ARM clock (ARM\_FCLK).

**I2Async Clock (I2ASYNC\_FCLK):** This clock is half the frequency of the ARM clock (ARM\_FCLK). It matches the OCP interface of the AXI2OCP bridge.

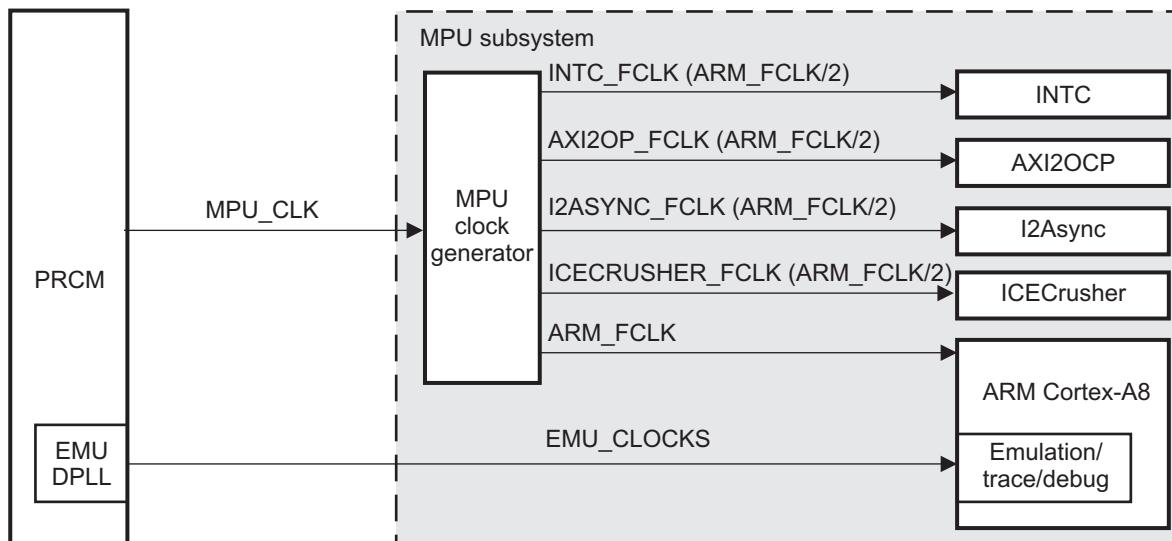
---

**NOTE:** The second half of the asynchronous bridge (T2ASYNC) is clocked directly by the PRCM with the core clock. T2ASYNC is not part of the MPU subsystem.

---

**Emulation Clocking:** Emulation clocks are distributed by the PRCM module and are asynchronous to the ARM core clock (ARM\_FCLK) and can run at a maximum of 1/3 the ARM core clock.

Table [Table 3-1](#) summarizes the clocks generated in the MPU subsystem by the MPU clock generator.



**Figure 3-3. MPU Subsystem Clocking Scheme**

**Table 3-1. MPU Subsystem Clock Frequencies**

| Clock signal                    | Frequency   |
|---------------------------------|-------------|
| Cortex A8 Core Functional Clock | MPU_CLK     |
| AXI2OCP Bridge Functional Clock | MPU_CLK / 2 |
| Device Clock                    | MPU_CLK / 2 |
| I2Async Bridge Functional Clock | MPU_CLK / 2 |

### 3.1.3.2 Reset Distribution

Resets to the MPU subsystem are provided by the PRCM and controlled by the clock generator module.

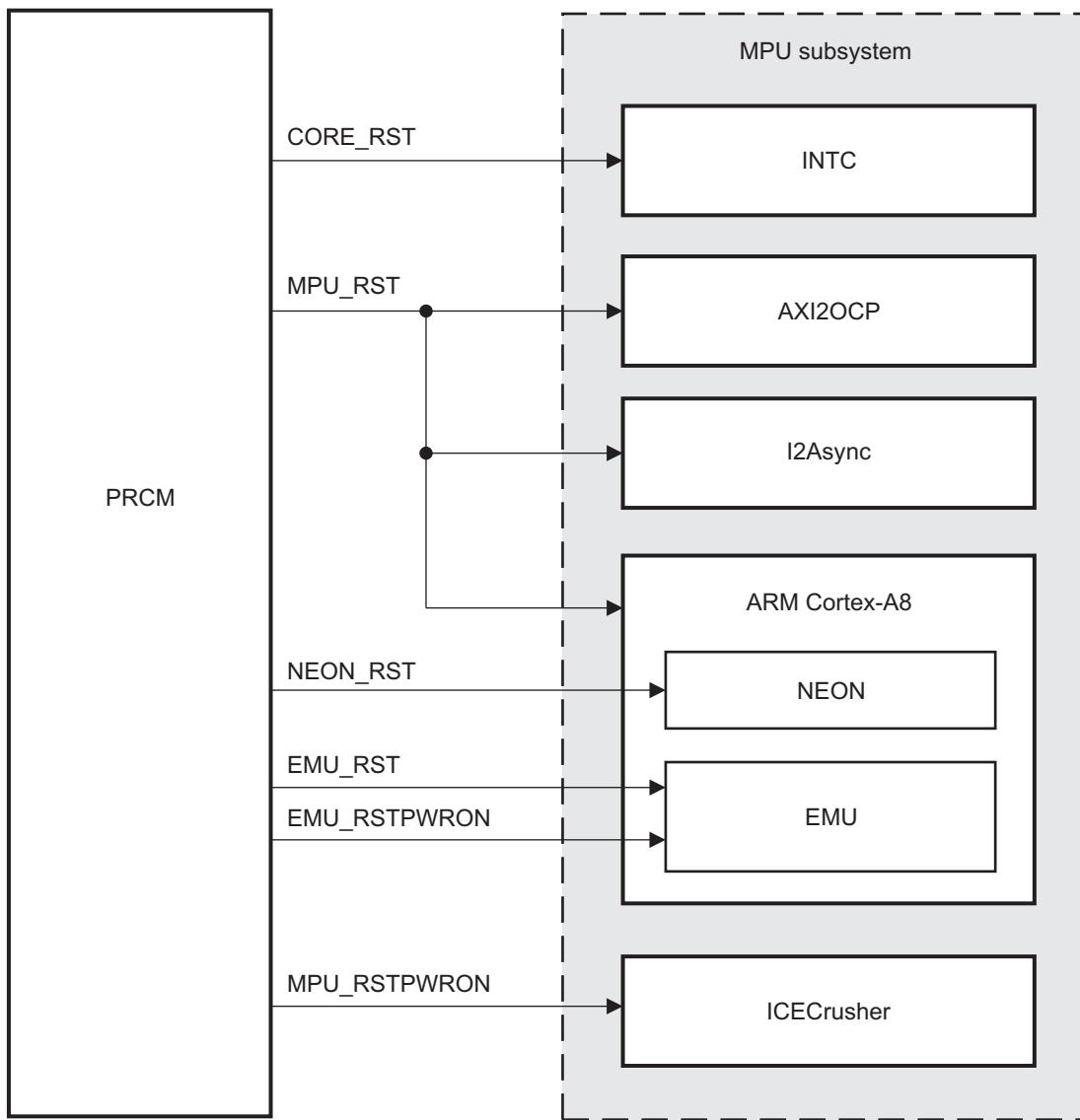


Figure 3-4. Reset Scheme of the MPU Subsystem

Table 3-2. Reset Scheme of the MPU Subsystem

| Signal Name  | I/O | Interface |
|--------------|-----|-----------|
| MPU_RST      | I   | PRCM      |
| NEON_RST     | I   | PRCM      |
| CORE_RST     | I   | PRCM      |
| MPU_RSTPWRON | I   | PRCM      |
| EMU_RST      | I   | PRCM      |
| EMU_RSTPWRON | I   | PRCM      |

For details about clocks, resets, and power domains, see [Chapter 8, Power, Reset, and Clock Management \(PRCM\)](#).

### 3.1.4 ARM Subchip

#### 3.1.4.1 ARM Overview

The ARM Cortex-A8 processor incorporates the technologies available in the ARM7™ architecture. These technologies include NEON™ for media and signal processing and Jazelle™ RCT for acceleration of realtime compilers, Thumb®-2 technology for code density and the VFPv3 floating point architecture.

#### 3.1.4.2 ARM Description

##### 3.1.4.2.1 ARM® Cortex™-A8 Instruction, Data, and Private Peripheral Port

The AXI bus interface is the main interface to the ARM system bus. It performs L2 cache fills and non-cacheable accesses for both instructions and data. The AXI interface supports 128bit and 64-bit wide input and output data buses. It supports multiple outstanding requests on the AXI bus and a wide range of bus clock to core clock ratios. The bus clock is synchronous with the core clock.

See the ARM ® Cortex™-A8 Technical Reference Manual for a complete programming model of the transaction rules (ordering, posting, and pipeline synchronization) that are applied depending on the memory region attribute associated with the transaction destination address.

##### 3.1.4.2.2 Secure Monitor Calls to Access CP15 Registers

The device supports special secure monitor functions that allows access to certain ARM core registers in privileged mode. Functions to write to the Auxiliary Control Register, Nonsecure Access Control Register, and the L2 Cache Auxiliary Control Register are provided (see the ARM Technical Reference Manual for a description of these registers).

| Service ID (R12) | Description  |
|------------------|--|
| 0x100            | Write value in R0 to Auxiliary Control Register          |
| 0x101            | Write value in R0 to Non Secure Access Control Register  |
| 0x102            | Write value in R0 to L2 Cache Auxiliary Control Register |

In general, the procedure to use these secure monitor call is as follows:

- Write the appropriate service ID to R12 .
- Load R0 with the value to write to the ARM core register.
- Perform barrier operations to ensure proper execution.
- Use SMC #1 (or SMI #1) to make the secure monitor call

Barrier instructions are also necessary to ensure a clean state before entering the secure monitor. Refer to the following example which provides the proper code sequence. This code provides an example of enabling ECC on L2 cache. Note: This function should be executed in an ARM privileged mode.

```
_enableL2ECC:
    STMFD sp!, {r0 - r4}          ; save context of R0-R4, which secure monitor call may use
    MRC p15, #1, r0, c9, c0, #2    ; Read L2 Cache Auxiliary Control Reg into R0
    MOV r1, #0                     ; Clear R1
    MOVT r1, #0x1020              ; enable mask for ECC (set bits 21 and 28 to enable ECC)
    ORR r0, r0, r1                ; OR with original register value
    MOV r12, #0x0102              ; setup service ID in R12
    MCR p15,#0x0,r1,c7,c5,#6     ; invalidate entire branch predictor array
    DSB                         ; data synchronization barrier operation
    ISB                          ; instruction synchronization barrier operation
    DMB                          ; data memory barrier operation
    SMC #1                       ; secure monitor call SMC (previously SMI)
    LDMFD sp!, {r0 - r4}          ; after returning from SMC, restore R0-R4
    MOV pc, lr                    ; return
```

### 3.1.4.2.3 ARM Core Supported Features

Table 3-3 provides a list of main functions of the Cortex™-A8 core supported inside the MPU Subsystem.

**Table 3-3. ARM Core Supported Features**

| Features                           | Comments   |
|------------------------------------|--|
| ARM version 7 ISA                  | Standard ARM instruction set + Thumb2™, JazelleX™ Java accelerator, and Media extensions.  |
|                                    | Backward compatible with previous ARM ISA versions.  |
| L1 Icache and Dcache               | 32 KB , 4-way, 16 word line, 128 bit interface.  |
| L2 Cache                           | 256 KB, 8-way, 16 word line, 128 bit interface to L1, ECC/Parity is supported. The L2 cache and cache controller are embedded within the ARM core.               |
|                                    | L2 valid bits cleared by software loop or by hardware.   |
| TLB                                | Fully associative and separate ITLB with 32 entries and DTLB with 32 entries.  |
| CoreSight ETM                      | The CoreSight ETM is embedded with the ARM core. The 32KB buffer (ETB) exists at the Chip Level (DebugSS)  |
| Branch Target Address Cache        | 512 entries  |
| Enhanced Memory Management Unit    | Mapping sizes are 4KB, 64KB, 1MB, and 16MB. (ARM MMU adds extended physical address ranges)  |
| Neon                               | Gives greatly enhanced throughput for media workloads and VFP-Lite support.  |
| Flat Memories                      | 176 Kbytes of ROM  |
|                                    | 64 Kbytes of RAM   |
| Buses                              | 128 bit AXI internal bus from CortexA8 routed by an AXI2OCP bridge to the interrupt controller, ROM, RAM, and 3 asynchronous OCP bridges (128 bits, and 64 bits) |
| Low interrupt latency              | Closely coupled INTC to the ARM core with 128 interrupt lines  |
| Vectored Interrupt Controller Port | Present.   |
| JTAG based debug                   | Supported via DAP  |
| Trace support                      | Supported via TPIU   |
| External Coprocessor               | Not supported  |

### 3.1.5 Interrupt Controller

The Host ARM Interrupt Controller (AINTC) is responsible for prioritizing all service requests from the system peripherals and generating either nIRQ or nFIQ to the host. The type of the interrupt (nIRQ or nFIQ) and the priority of the interrupt inputs are programmable. The AINTC interfaces to the ARM processor via the AXI port through an AXI2OCP bridge and runs at half the processor speed. It has the capability to handle up to 128 requests which can be steered/prioritized as A8 nFIQ or nIRQ interrupt requests.

The general features of the AINTC are:

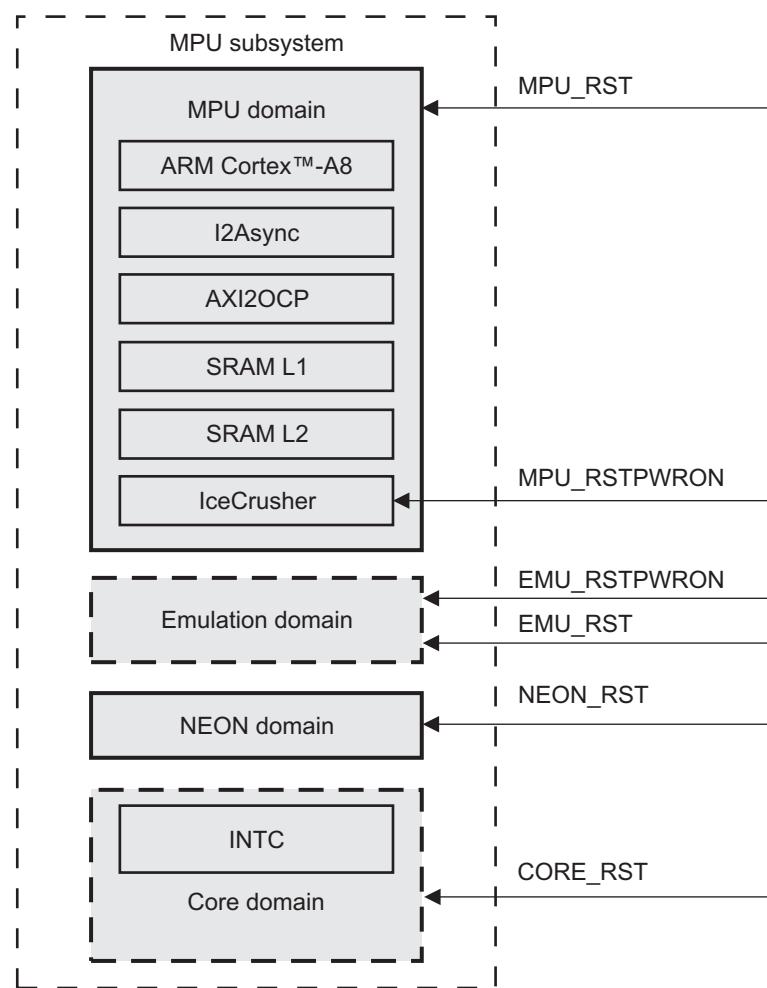
- Up to 128 level sensitive interrupts inputs
- Individual priority for each interrupt input
- Each interrupt can be steered to nFIQ or nIRQ
- Independent priority sorting for nFIQ and nIRQ

### 3.1.6 Power Management

#### 3.1.6.1 Power Domains

The MPU subsystem is divided into 5 power domains controlled by the PRCM, as shown in Figure 3-5.

**NOTE:** The emulation domain and the core domain are not fully embedded in MPU subsystem.



**Figure 3-5. MPU Subsystem Power Domain Overview**

Power management requirements at the device level govern power domains for the MPU subsystem. The device-level power domains are directly aligned with voltage domains and thus can be represented as a cross reference to the different voltage domains.

Table 3-4 shows the different power domains of the MPU subsystem and the modules inside.

**Table 3-4. Overview of the MPU Subsystem Power Domain**

| Functional Power Domain | Physical Power Domain per System/Module   |
|-------------------------|---|
| MPU subsystem domain    | ARM, AXI2OCP, I2Asynch Bridge, ARM L1 and L2 periphery logic and array, ICE-Crusher, ETM, APB modules |
| MPU NEON domain         | ARM NEON accelerator  |
| CORE domain             | MPU interrupt controller  |
| EMU domain              | EMU (ETB,DAP)   |

---

**NOTE:** L1 and L2 array memories have separate control signals into the in MPU Subsystem, thus directly controlled by PRCM. For details on the physical power domains and the voltage domains, see [Chapter 8, Power, Reset, and Clock Management \(PRCM\)](#).

---

### 3.1.6.2 Power States

Each power domain can be driven by the PRCM in 4 different states, depending on the functional mode required by the user.

For each power domain the PRCM manages all transitions by controlling domain clocks, domain resets, domain logic power switches and memory power switches.

**Table 3-5. MPU Power States**

| Power State | Logic Power | Memory Power | Clocks                  |
|-------------|-------------|--------------|-------------------------|
| Active      | On          | On or Off    | On (at least one clock) |
| Inactive    | On          | On or Off    | Off                     |
| Off         | Off         | Off          | Off (all clocks)        |

### 3.1.6.3 Power Modes

The major part of the MPU subsystem belongs to the MPU power domain. The modules inside this power domain can be off at a time when the ARM processor is in an OFF or standby mode. IDLE/WAKEUP control is managed by the clock generator block but initiated by the PRCM module.

The MPU Standby status can be checked with PRCM.CM\_IDLEST\_MP0[0] ST\_MP0 bit. For the MPU to be on, the core (referred here as the device core) power must be on. Device power management does not allow INTc to go to OFF state when MPU domain is on (active or one of retention modes).

The NEON core has independent power off mode when not in use. Enabling and disabling of NEON can be controlled by software.

**CAUTION**

The MPU L1 cache memory does not support retention mode, and its array switch is controlled together with the MPU logic. For compliance, the L1 retention control signals exist at the PRCM boundary, but are not used. The ARM L2 can be put into retention independently of the other domains.

[Table 3-6](#) outlines the supported operational power modes. All other combinations are illegal. The ARM L2, NEON, and ETM/Debug can be powered up/down independently. The APB/ATB ETM/Debug column refers to all three features: ARM emulation, trace, and debug.

The MPU subsystem must be in a power mode where the MPU power domain, NEON power domain, debug power domain, and INTc power domain are in standby, or off state.

**Table 3-6. MPU Subsystem Operation Power Modes**

| Mode | MPU and ARM Core Logic | ARM L2 RAM | NEON INTC | Device Core and ETM | APB/ATB Debug       |
|------|------------------------|------------|-----------|---------------------|---------------------|
| 1    | Active                 | Active     | Active    | Active              | Disabled or enabled |
| 2    | Active                 | Active     | OFF       | Active              | Disabled or enabled |
| 3    | Active                 | RET        | Active    | Active              | Disabled or enabled |
| 4    | Active                 | RET        | OFF       | Active              | Disabled or enabled |
| 5    | Active                 | OFF        | Active    | Active              | Disabled or enabled |
| 6    | Active                 | OFF        | OFF       | Active              | Disabled or enabled |
| 7    | OFF                    | RET        | OFF       | OFF                 | Disabled or enabled |
| 8    | Standby                | Active     | Standby   | Active              | Disabled or enabled |
| 9    | Standby                | Active     | OFF       | Active              | Disabled or enabled |
| 10   | Standby                | RET        | Standby   | Active              | Disabled or enabled |
| 11   | Standby                | RET        | OFF       | Active              | Disabled or enabled |
| 12   | Standby                | OFF        | Standby   | Active              | Disabled or enabled |
| 13   | Standby                | OFF        | OFF       | Active              | Disabled or enabled |
| 14   | OFF                    | OFF        | OFF       | OFF                 | Disabled or enabled |

### 3.1.7 ARM Programming Model

For detailed descriptions of registers used for MPU configuration, see [Chapter 8, Power, Reset, and Clock Management \(PRCM\)](#).

#### 3.1.7.1 Clock Control

For clock configuration settings, see [Chapter 8, Power, Reset, and Clock Management \(PRCM\)](#).

#### 3.1.7.2 MPU Power Mode Transitions

The following subsections describe transitions of different power modes for MPU power domain:

- Basic power on reset
- MPU into standby mode
- MPU out of standby mode
- MPU power on from a powered off state

##### 3.1.7.2.1 Basic Power-On Reset

The power-on reset follows the following sequence of operation and is applicable to initial power-up and wakeup from device off mode:

Reset the INTC (CORE\_RST) and the MPU subsystem modules (MPU\_RST). The clocks must be active during the MPU reset and CORE reset.

##### 3.1.7.2.2 MPU Into Standby Mode

The MPU into standby mode follows the following sequence of operation and is applicable to initial power-up and wakeup from device Off mode.

1. The ARM core initiates entering into standby via software only (CP15 - WFI).
2. MPU modules requested internally of MPU subsystem to enter idle, after ARM core standby detected.
3. MPU is in standby output asserted for PRCM (all outputs guaranteed to be at reset values).
4. PRCM can now request INTC to enter into idle mode. Acknowledge from INTC goes to PRCM.

---

**NOTE:** The INTC SWAKEUP output is a pure hardware signal to PRCM for the status of its IDLE request and IDLE acknowledge handshake.

---

---

**NOTE:** In debug mode, ICE-Crusher could prevent MPU subsystem from entering into IDLE mode.

---

### 3.1.7.2.3 MPU Out Of Standby Mode

The MPU out of standby mode follows the following sequence of operation and is applicable to initial power-up and wakeup from device Off mode.

1. PRCM must start clocks through DPLL programming.
2. Detect active clocking via status output of DPLL.
3. Initiate an interrupt through the INTC to wake up the ARM core from STANDBYWFI mode.

### 3.1.7.2.4 MPU Power On From a Powered-Off State

1. MPU Power On, NEON Power On, Core Power On (INTC) should follow the ordered sequence per power switch daisy chain to minimize the peaking of current during power-up.

---

**NOTE:** The core domain must be on, and reset, before the MPU can be reset.

---

2. Follow the reset sequence as described in the Basic Power-On Reset section.

## **Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS)**

---

---

---

This chapter describes the PRU-ICSS for the device.

| <b>Topic</b>                     | <b>Page</b> |
|----------------------------------|-------------|
| <b>4.1    Introduction .....</b> | <b>174</b>  |

## 4.1 Introduction

The Programmable Real-Time Unit and Industrial Communication Subsystem (PRU-ICSS) consists of dual 32-bit RISC cores (Programmable Real-Time Units, or PRUs), memories, interrupt controller, and internal peripherals that enable additional peripheral interfaces and protocols. The subsystem available on this device is the next-generation PRU (PRUSSv2) compared to the AM1x and OMAP-L13x.

Among the interfaces supported by the PRU-ICSS are the real-time industrial protocols used in master and slave mode, such as:

- EtherCAT®
- PROFINET
- EtherNet/IP™
- PROFIBUS
- POWERLINK
- SERCOS III

---

**NOTE:** For the availability of EtherCAT in the AM335x family of devices, see [Table 1-1, Device Features](#).

---

## **Graphics Accelerator (SGX)**

This chapter describes the graphics accelerator for the device.

| Topic                                   | Page       |
|---|------------|
| 5.1 <b>Introduction</b> .....           | <b>176</b> |
| 5.2 <b>Integration</b> .....            | <b>179</b> |
| 5.3 <b>Functional Description</b> ..... | <b>181</b> |

## 5.1 Introduction

This chapter describes the 2D/3D graphics accelerator (SGX) for the device.

---

**NOTE:** The SGX subsystem is a Texas Instruments instantiation of the POWERVR® SGX530 core from Imagination Technologies Ltd.

This document contains materials that are ©2003-2007 Imagination Technologies Ltd.

POWERVR® and USSE™ are trademarks or registered trademarks of Imagination Technologies Ltd.

---

The 2D/3D graphics accelerator (SGX) subsystem accelerates 2-dimensional (2D) and 3-dimensional (3D) graphics applications. The SGX subsystem is based on the POWERVR® SGX core from Imagination Technologies. SGX is a new generation of programmable POWERVR graphic cores. The POWERVR SGX530 v1.2.5 architecture is scalable and can target all market segments from mainstream mobile devices to high-end desktop graphics. Targeted applications include feature phone, PDA, and hand-held games.

### 5.1.1 POWERVR SGX Main Features

- 2D graphics, 3D graphics, vector graphics, and programming support for GP-GPU functions
- Tile-based architecture
- Universal scalable shader engine ( USSE™) – multithreaded engine incorporating pixel and vertex shader functionality
- Advanced shader feature set – in excess of Microsoft VS3.0, PS3.0, and OpenGL2.0
- Industry-standard API support – Direct3D Mobile, OpenGL ES 1.1 and 2.0, OpenVG v1.0.1
- Fine-grained task switching, load balancing, and power management
- Advanced geometry direct memory access (DMA) driven operation for minimum CPU interaction
- Programmable high-quality image anti-aliasing
- POWERVR SGX core MMU for address translation from the core virtual address to the external physical address (up to 4GB address range)
- Fully virtualized memory addressing for OS operation in a unified memory architecture
- Advanced and standard 2D operations [e.g., vector graphics, BLTs (block level transfers), ROPs (raster operations)]
- 32K stride support

### 5.1.2 SGX 3D Features

- Deferred pixel shading
- On-chip tile floating point depth buffer
- 8-bit stencil with on-chip tile stencil buffer
- 8 parallel depth/stencil tests per clock
- Scissor test
- Texture support:
  - Cube map
  - Projected textures
  - 2D textures
  - Nonsquare textures
- Texture formats:
  - RGBA 8888, 565, 1555
  - Monochromatic 8, 16, 16f, 32f, 32int
  - Dual channel, 8:8, 16:16, 16f:16f

- Compressed textures PVR-TC1, PVR-TC2, ETC1
  - Programmable support for all YUV formats
  - Resolution support:
    - Frame buffer maximum size = 2048 x 2048
    - Texture maximum size = 2048 x 2048
  - Texture filtering:
    - Bilinear, trilinear, anisotropic
    - Independent minimum and maximum control
  - Antialiasing:
    - 4x multisampling
    - Up to 16x full scene anti-aliasing
    - Programmable sample positions
  - Indexed primitive list support
    - Bus mastered
  - Programmable vertex DMA
  - Render to texture:
    - Including twiddled formats
    - Auto MipMap generation
  - Multiple on-chip render targets (MRT).
- Note:** Performance is limited when the on-chip memory is not available.

### 5.1.3 Universal Scalable Shader Engine (USSE) – Key Features

The USSE is the engine core of the POWERVR SGX architecture and supports a broad range of instructions.

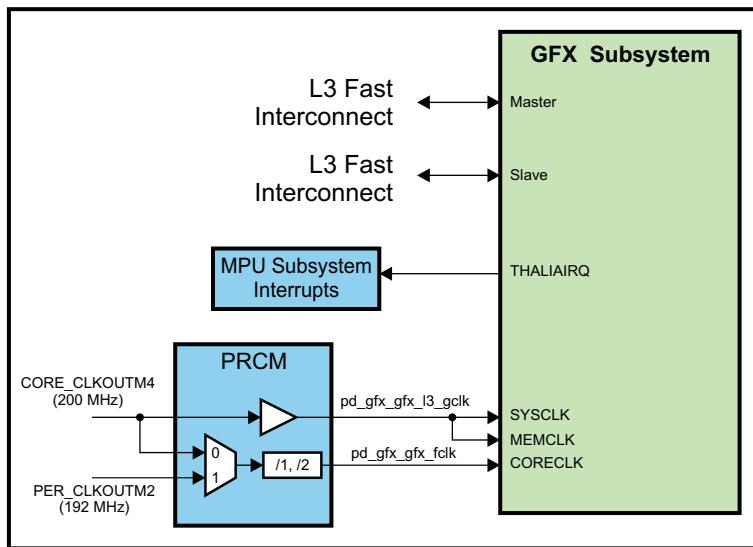
- Single programming model:
  - Multithreaded with 16 simultaneous execution threads and up to 64 simultaneous data instances
  - Zero-cost swapping in, and out, of threads
  - Cached program execution model
  - Dedicated pixel processing instructions
  - Dedicated video encode/decode instructions
- SIMD execution unit supporting operations in:
  - 32-bit IEEE float
  - 2-way 16-bit fixed point
  - 4-way 8-bit integer
  - 32-bit bit-wise (logical only)
- Static and dynamic flow control:
  - Subroutine calls
  - Loops
  - Conditional branches
  - Zero-cost instruction predication
- Procedural geometry:
  - Allows generation of primitives
  - Effective geometry compression
  - High-order surface support
- External data access:
  - Permits reads from main memory using cache

- Permits writes to main memory
- Data fence facility
- Dependent texture reads

#### **5.1.4 Unsupported Features**

There are no unsupported SGX530 features for this device.

## 5.2 Integration



**Figure 5-1. SGX530 Integration**

### 5.2.1 SGX530 Connectivity Attributes

The general connectivity attributes of the SGX530 are shown in the following table.

**Table 5-1. SGX530 Connectivity Attributes**

| Attributes          | Type                                |
|---------------------|-------------------------------------|
| Power domain        | GFX Domain                          |
| Clock domain        | SGX_CLK                             |
| Reset signals       | SGX_RST                             |
| Idle/Wakeup signals | Smart Idle Initiator Standby        |
| Interrupt request   | THALIAIRQ (GFXINT) to MPU Subsystem |
| DMA request         | None                                |
| Physical address    | L3 Fast slave port                  |

### 5.2.2 SGX530 Clock and Reset Management

The SGX530 uses separate functional and interface clocks. The SYSCLK is the clock for the slave interface and runs at the L3F frequency. The MEMCLK is the clock for the memories and master interface and also runs at the L3F frequency. The CORECLK is the functional clock. It can be sourced from either the L3F clock (CORE\_CLKOUTM4) or from the 192 MHz PER\_CLKOUTM2 and can optionally be divided by 2.

**Table 5-2. SGX530 Clock Signals**

| Clock signal                | Max Freq | Reference / Source               | Comments                        |
|-----------------------------|----------|----------------------------------|---------------------------------|
| SYSCLK<br>Interface clock   | 200 MHz  | CORE_CLKOUTM4                    | pd_gfx_gfx_l3_gclk<br>From PRCM |
| MEMCLK<br>Memory Clock      | 200 MHz  | CORE_CLKOUTM4                    | pd_gfx_gfx_l3_gclk<br>From PRCM |
| CORECLK<br>Functional clock | 200 MHz  | PER_CLKOUTM2 or<br>CORE_CLKOUTM4 | pd_gfx_gfx_fclk<br>From PRCM    |

### 5.2.3 SGX530 Pin List

The SGX530 module does not include any external interface pins.

## 5.3 Functional Description

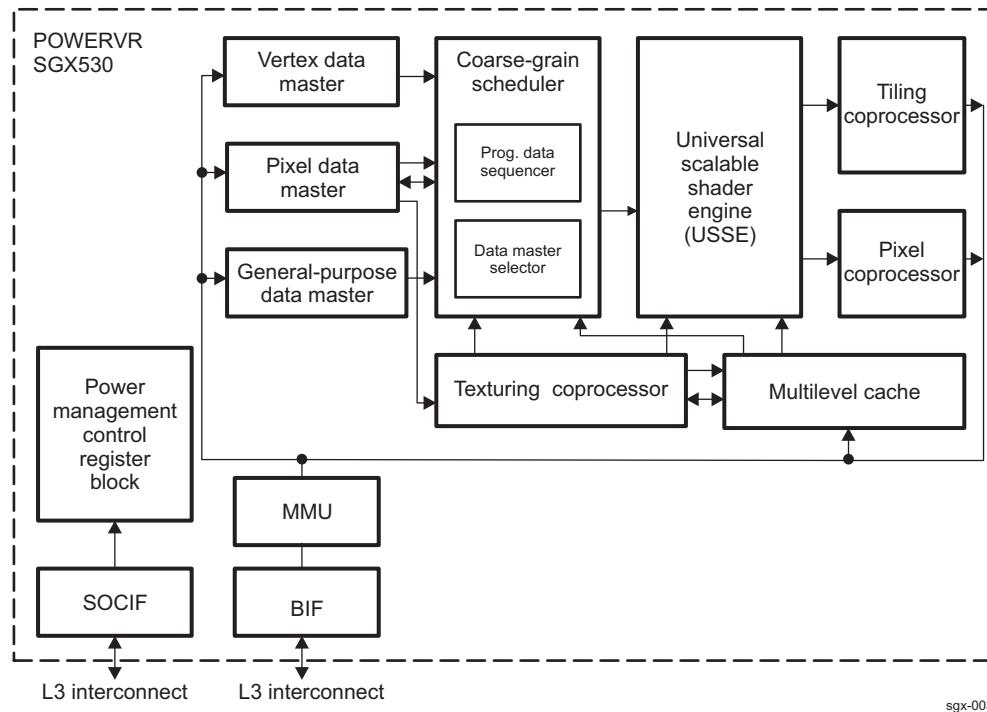
### 5.3.1 SGX Block Diagram

The SGX subsystem is based on the POWERVR® SGX530 core from Imagination Technologies. The architecture uses programmable and hard coded pipelines to perform various processing tasks required in 2D, 3D, and video processing. The SGX architecture comprises the following elements:

- Coarse grain scheduler
  - Programmable data sequencer (PDS)
  - Data master selector (DMS)
- Vertex data master (VDM)
- Pixel data master (PDM)
- General-purpose data master
- USSE
- Tiling coprocessor
- Pixel coprocessor
- Texturing coprocessor
- Multilevel cache

Figure 5-2 shows a block diagram of the SGX cores.

**Figure 5-2. SGX Block Diagram**



sgx-003

### 5.3.2 SGX Elements Description

The coarse grain scheduler (CGS) is the main system controller for the POWERVR SGX architecture. It consists of two stages, the DMS and the PDS. The DMS processes requests from the data masters and determines which tasks can be executed given the resource requirements. The PDS then controls the loading and processing of data on the USSE.

There are three data masters in the SGX core:

- The VDM is the initiator of transform and lighting processing within the system. The VDM reads an

input control stream, which contains triangle index data and state data. The state data indicates the PDS program, size of the vertices, and the amount of USSE output buffer resource available to the VDM. The triangle data is parsed to determine unique indices that must be processed by the USSE. These are grouped together according to the configuration provided by the driver and presented to the DMS.

- The PDM is the initiator of rasterization processing within the system. Each pixel pipeline processes pixels for a different half of a given tile, which allows for optimum efficiency within each pipe due to locality of data. It determines the amount of resource required within the USSE for each task. It merges this with the state address and issues a request to the DMS for execution on the USSE.
- The general-purpose data master responds to events within the system (such as end of a pass of triangles from the ISP, end of a tile from the ISP, end of render, or parameter stream breakpoint event). Each event causes either an interrupt to the host or synchronized execution of a program on the PDS. The program may, or may not cause a subsequent task to be executed on the USSE.

The USSE is a user-programmable processing unit. Although general in nature, its instructions and features are optimized for three types of task: processing vertices (vertex shading), processing pixels (pixel shading), and video/imaging processing.

The multilevel cache is a 2-level cache consisting of two modules: the main cache and the mux/arbitrator/demux/decompression unit (MADD). The MADD is a wrapper around the main cache module designed to manage and format requests to and from the cache, as well as providing Level 0 caching for texture and USSE requests. The MADD can accept requests from the PDS, USSE, and texture address generator modules. Arbitration, as well as any required texture decompression, are performed between the three data streams.

The texturing coprocessor performs texture address generation and formatting of texture data. It receives requests from either the iterators or USSE modules and translates these into requests in the multilevel cache. Data returned from the cache are then formatted according to the texture format selected, and sent to the USSE for pixel-shading operations.

To process pixels in a tiled manner, the screen is divided into tiles and arranged as groups of tiles by the tiling coprocessor. An inherent advantage of tiling architecture is that a large amount of vertex data can be rejected at this stage, thus reducing the memory storage requirements and the amount of pixel processing to be performed.

The pixel coprocessor is the final stage of the pixel-processing pipeline and controls the format of the final pixel data sent to the memory. It supplies the USSE with an address into the output buffer and then USSE returns the relevant pixel data. The address order is determined by the frame buffer mode. The pixel coprocessor contains a dithering and packing function.

## **Interrupts**

This section describes the interrupts for the device.

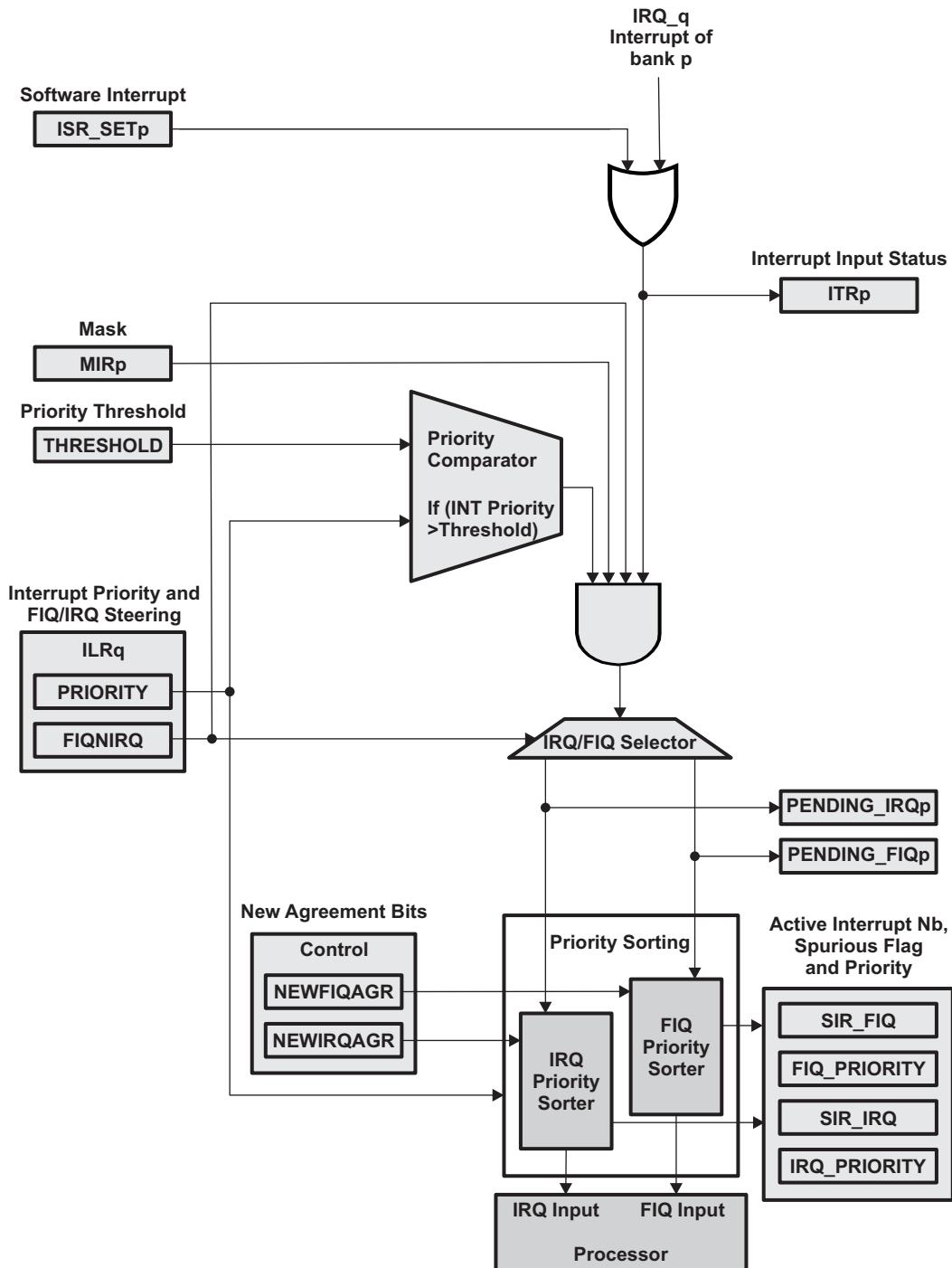
| Topic                                    | Page |
|--|------|
| 6.1 Functional Description .....         | 184  |
| 6.2 Basic Programming Model .....        | 187  |
| 6.3 ARM Cortex-A8 Interrupts .....       | 196  |
| 6.4 ARM Cortex-M3 Interrupts .....       | 200  |
| 6.5 PWM Events .....                     | 202  |
| 6.6 Interrupt Controller Registers ..... | 203  |

## 6.1 Functional Description

The interrupt controller processes incoming interrupts by masking and priority sorting to produce the interrupt signals for the processor to which it is attached. [Figure 6-1](#) shows the top-level view of interrupt processing.

**NOTE:** FIQ is not available on general-purpose (GP) devices.

**Figure 6-1. Interrupt Controller Block Diagram**



## 6.1.1 Interrupt Processing

### 6.1.1.1 Input Selection

The INTC supports only level-sensitive incoming interrupt detection. A peripheral asserting an interrupt maintains it until software has handled the interrupt and instructed the peripheral to deassert the interrupt. A software interrupt is generated if the corresponding bit in the MPU\_INTC.INTC\_ISR\_SETn register is set (register bank number: n = [0,1,2,3] for the MPU subsystem INTC, 128 incoming interrupt lines are supported). The software interrupt clears when the corresponding bit in the MPU\_INTC.INTC\_ISR\_CLEARn register is written. Typical use of this feature is software debugging.

### 6.1.1.2 Masking

#### 6.1.1.2.1 Individual Masking

Detection of interrupts on each incoming interrupt line can be enabled or disabled independently by the MPU\_INTC.INTC\_MIRn interrupt mask register. In response to an unmasked incoming interrupt, the INTC can generate one of two types of interrupt requests to the processor:

- IRQ: low-priority interrupt request
- FIQ: fast interrupt request (Not available on General Purpose (GP) devices)

The type of interrupt request is determined by the MPU\_INTC.INTC\_ILRm[0] FIQNIRQ bit (m= [0,127]). The current incoming interrupt status before masking is readable from the MPU\_INTC.INTC\_ITRn register. After masking and IRQ/FIQ selection, and before priority sorting is done, the interrupt status is readable from the MPU\_INTC.INTC\_PENDING\_IRQn and MPU\_INTC.INTC\_PENDING\_FIQn registers.

#### 6.1.1.2.2 Priority Masking

To enable faster processing of high-priority interrupts, a programmable priority masking threshold is provided (the MPU\_INTC.INTC\_THRESHOLD[7:0] PRIORITYTHRESHOLD field). This priority threshold allows preemption by higher priority interrupts; all interrupts of lower or equal priority than the threshold are masked. However, priority 0 can never be masked by this threshold; a priority threshold of 0 is treated the same way as priority 1. PRIORITY and PRIORITYTHRESHOLD fields values can be set between 0x0 and 0x7F; 0x0 is the highest priority and 0x7F is the lowest priority. When priority masking is not necessary, a priority threshold value of 0xFF disables the priority threshold mechanism. This value is also the reset default for backward compatibility with previous versions of the INTC.

### 6.1.1.3 Priority Sorting

A priority level (0 being the highest) is assigned to each incoming interrupt line. Both the priority level and the interrupt request type are configured by the MPU\_INTC.INTC\_ILRm register. If more than one incoming interrupt with the same priority level and interrupt request type occur simultaneously, the highest-numbered interrupt is serviced first. When one or more unmasked incoming interrupts are detected, the INTC separates between IRQ and FIQ using the corresponding MPU\_INTC.INTC\_ILRm[0] FIQNIRQ bit.

The result is placed in INTC\_PENDING\_IRQn or INTC\_PENDING\_FIQn. If no other interrupts are currently being processed, INTC asserts IRQ/FIQ and starts the priority computation. Priority sorting for IRQ and FIQ can execute in parallel. Each IRQ/FIQ priority sorter determines the highest priority interrupt number. Each priority number is placed in the corresponding MPU\_INTC.INTC\_SIR\_IRQ[6:0] ACTIVEIRQ field or MPU\_INTC.INTC\_SIR\_FIQ[6:0] ACTIVEFIQ field. The value is preserved until the corresponding MPU\_INTC.INTC\_CONTROL NEWIRQAGR or NEWFIQAGR bit is set. Once the interrupting peripheral device has been serviced and the incoming interrupt deasserted, the user must write to the appropriate NEWIRQAGR or NEWFIQAGR bit to indicate to the INTC the interrupt has been handled. If there are any pending unmasked incoming interrupts for this interrupt request type, the INTC restarts the appropriate priority sorter; otherwise, the IRQ or FIQ interrupt line is deasserted.

### 6.1.2 Register Protection

If the MPU\_INTC.INTC\_PROTECTION[0] PROTECTION bit is set, access to the INTC registers is restricted to the supervisor mode. Access to the MPU\_INTC.INTC\_PROTECTION register is always restricted to privileged mode. For more information, see [Section 6.6.1.7, INTC\\_PROTECTION Register \(offset = 4Ch\) \[reset = 0h\]](#).

### 6.1.3 Module Power Saving

The INTC provides an auto-idle function in its three clock domains:

- Interface clock
- Functional clock
- Synchronizer clock

The interface clock auto-idle power-saving mode is enabled if the MPU\_INTC.INTC\_SYSCONFIG[0] AUTOIDLE bit is set to 1. When this mode is enabled and there is no activity on the bus interface, the interface clock is disabled internally to the module, thus reducing power consumption. When there is new activity on the bus interface, the interface clock restarts without any latency penalty. After reset, this mode is disabled, by default. The functional clock auto-idle power-saving mode is enabled if the MPU\_INTC.INTC\_IDLE[0] FUNCIDLE bit is set to 0. When this mode is enabled and there is no active interrupt (IRQ or FIQ interrupt being processed or generated) or no pending incoming interrupt, the functional clock is disabled internally to the module, thus reducing power consumption.

When a new unmasked incoming interrupt is detected, the functional clock restarts and the INTC processes the interrupt. If this mode is disabled, the interrupt latency is reduced by one cycle. After reset, this mode is enabled, by default. The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked. The synchronizer input clock has an auto-idle power-saving mode enabled if the MPU\_INTC.INTC\_IDLE[1] TURBO bit is set to 1. If the auto-idle mode is enabled, the standby power is reduced, but the IRQ or FIQ interrupt latency increases from four to six functional clock cycles. This feature can be enabled dynamically according to the requirements of the device. After reset, this mode is disabled, by default.

### 6.1.4 Error Handling

The following accesses will cause an error:

- Privilege violation (attempt to access PROTECTION register in user mode or any register in user mode if Protection bit is set)
- Unsupported commands

The following accesses will not cause any error response:

- Access to a non-decoded address
- Write to a read-only register

### 6.1.5 Interrupt Handling

The IRQ/FIQ interrupt generation takes four INTC functional clock cycles (plus or minus one cycle) if the MPU\_INTC.INTC\_IDLE[1] TURBO bit is set to 0. If the TURBO bit is set to 1, the interrupt generation takes six cycles, but power consumption is reduced while waiting for an interrupt. These latencies can be reduced by one cycle by disabling functional clock auto-idle (MPU\_INTC.INTC\_IDLE[0] FUNCIDLE bit set to 1), but power consumption is increased, so the benefit is minimal.

To minimize interrupt latency when an unmasked interrupt occurs, the IRQ or FIQ interrupt is generated before priority sorting completion. The priority sorting takes 10 functional clock cycles, which is less than the minimum number of cycles required for the MPU to switch to the interrupt context after reception of the IRQ or FIQ event.

Any read of the MPU\_INTC.INTC\_SIR\_IRQ or MPU\_INTC.INTC\_SIR\_FIQ register during the priority sorting process stalls until priority sorting is complete and the relevant register is updated. However, the delay between the interrupt request being generated and the interrupt service routine being executed is such that priority sorting always completes before the MPU\_INTC.INTC\_SIR\_IRQ or MPU\_INTC.INTC\_SIR\_FIQ register is read.

## 6.2 Basic Programming Model

### 6.2.1 Initialization Sequence

1. Program the MPU\_INTC.INTC\_SYSCONFIG register: If necessary, enable the interface clock autogating by setting the AUTOIDLE bit.
2. Program the MPU\_INTC.INTC\_IDLE register: If necessary, disable functional clock autogating or enable synchronizer autogating by setting the FUNCIDLE bit or TURBO bit accordingly.
3. Program the MPU\_INTC.INTC\_ILRm register for each interrupt line: Assign a priority level and set the FIQNIRQ bit for an FIQ interrupt (by default, interrupts are mapped to IRQ and priority is 0x0 [highest]).
4. Program the MPU\_INTC.INTC\_MIRn register: Enable interrupts (by default, all interrupt lines are masked). NOTE: To program the MPU\_INTC.INTC\_MIRn register, the MPU\_INTC.INTC\_MIR\_SETn and MPU\_INTC.INTC\_MIR\_CLEARn registers are provided to facilitate the masking, even if it is possible for backward-compatibility to write directly to the MPU\_INTC.INTC\_MIRn register.

### 6.2.2 INTC Processing Sequence

After the INTC\_MIRn and INTC\_ILRm registers are configured to enable and assign priorities to incoming interrupts, the interrupt is processed as explained in the following subsections. IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in the code below.

1. One or more unmasked incoming interrupts (M\_IRQ\_n signals) are received and IRQ or FIQ outputs (IRQ/FIQ) are not currently asserted.
2. If the INTC\_ILRm[0] FIQNIRQ bit is cleared to 0, the MPU\_INTC\_IRQ output signal is generated. If the FIQNIRQ bit is set to 1, the MPU\_INTC\_FIQ output signal is generated.
3. The INTC performs the priority sorting and updates the INTC\_SIR\_IRQ[6:0] ACTIVEIRQ /INTC\_SIR\_FIQ[6:0] ACTIVEFIQ field with the current interrupt number.
4. During priority sorting, if the IRQ/FIQ is enabled at the host processor side, the host processor automatically saves the current context and executes the ISR as follows.

The ARM host processor automatically performs the following actions in pseudo code:

```

LR = PC + 4          /* return link */
SPSR = CPSR          /* Save CPSR before execution */
CPSR[5] = 0           /* Execute in ARM state */
CPSR[7] = 1           /* Disable IRQ */
CPSR[8] = 1           /* Disable Imprecise Data Aborts */
CPSR[9] = CP15_reg1_EEbit /* Endianness on exception entry */

if interrupt == IRQ then
  CPSR[4:0] = 0b10010 /* Enter IRQ mode */
  if high vectors configured then
    PC = 0xFFFF0018
  else
    PC = 0x00000018 /* execute interrupt vector */
  else if interrupt == FIQ then
    CPSR[4:0] = 0b10001 /* Enter FIQ mode */
    CPSR[6] = 1           /* Disable FIQ */
    if high vectors configured then
      PC = 0xFFFF001C
    else
      PC = 0x0000001C /* execute interrupt vector */
  endif
endif

```

5. The ISR saves the remaining context, identifies the interrupt source by reading the ACTIVEIRQ/ACTIVEFIQ field, and jumps to the relevant subroutine handler as follows:

**CAUTION**

The code in steps 5 and 7 is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
; INTC_SIR_IRQ/INTC_SIR_FIQ register address
INTC_SIR_IRQ_ADDR/INTC_SIR_FIQ_ADDR .word 0x48200040/0x48200044
; ACTIVEIRQ bit field mask to get only the bit field
ACTIVEIRQ_MASK .equ 0x7F
 IRQ_ISR/_FIQ_ISR:
; Save the critical context
STMFD SP!, {R0-R12, LR} ; Save working registers and the Link register
MRS R11, SPSR ; Save the SPSR into R11
; Get the number of the highest priority active IRQ/FIQ
LDR R10, INTC_SIR_IRQ_ADDR/INTC_SIR_FIQ_ADDR
LDR R10, [R10] ; Get the INTC_SIR_IRQ/INTC_SIR_FIQ register
AND R10, R10, #ACTIVEIRQ_MASK ; Apply the mask to get the active IRQ number
; Jump to relevant subroutine handler
LDR PC, [PC, R10, lsl #2] ; PC base address points this instruction + 8
NOP ; To index the table by the PC
; Table of handler start addresses
.word IRQ0handler ;For IRQ0 of BANK0
.word IRQ1handler
.word IRQ2handler
```

6. The subroutine handler executes code specific to the peripheral generating the interrupt by handling the event and deasserting the interrupt condition at the peripheral side.

```
; IRQ0 subroutine
IRQ0handler:
; Save working registers
STMFD SP!, {R0-R1}
; Now read-modify-write the peripheral module status register
; to de-assert the M_IRQ_0 interrupt signal
; De-Assert the peripheral interrupt
MOV R0, #0x7 ; Mask for 3 flags
LDR R1, MODULE0_STATUS_REG_ADDR ; Get the address of the module Status Register
STR R0, [R1] ; Clear the 3 flags
; Restore working registers LDMFD SP!, {R0-R1}
; Jump to the end part of the ISR
B IRQ_ISR_end/FIQ_ISR_end
```

7. After the return of the subroutine, the ISR sets the NEWIRQAGR/NEWFIQAGR bit to enable the processing of subsequent pending IRQs/FIQs and to restore ARM context in the following code. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/FIQs, a Data Synchronization Barrier is used. This operation ensure that the IRQ/FIQ line is de-asserted before IRQ/FIQ enabling. After that, the INTC processes any other pending interrupts or deasserts the IRQ/FIQ signal if there is no interrupt.

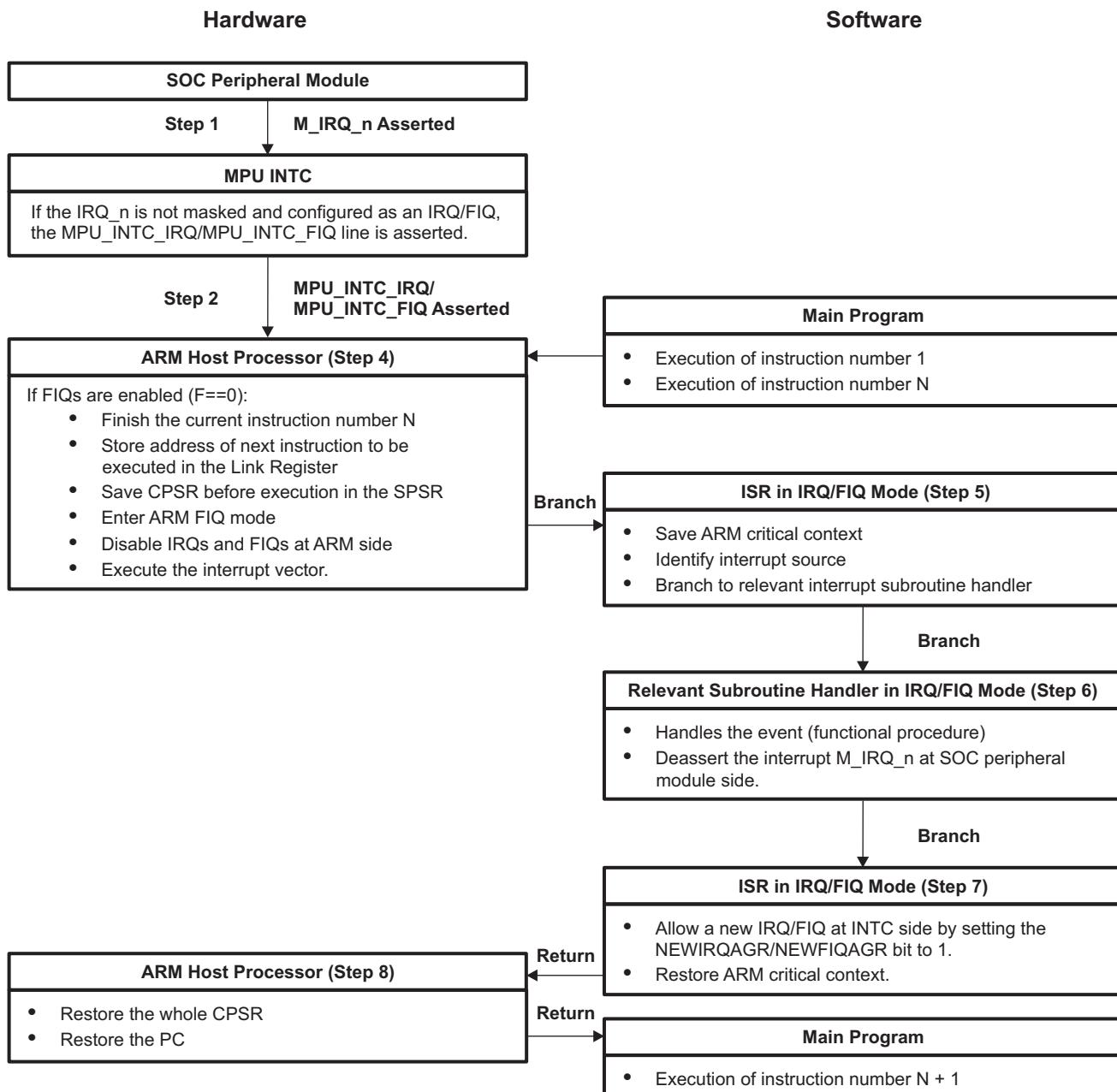
```

; INTC_CONTROL register address
INTC_CONTROL_ADDR .word 0x48200048;
NEWIRQAGR/NEWFIQAGR bit mask to set only the NEWIRQAGR/NEWFIQAGR bit
NEWIRQAGR_MASK/NEWFIQAGR_MASK .equ 0x01/0x02
IRQ_ISR_end/FIQ_ISR_end:
    ; Allow new IRQs/FIQs at INTC side
    ; The INTC_CONTROL register is a write only register so no need to write back others bits
    MOV R0, #NEWIRQAGR_MASK/NEWFIQAGR_MASK ; Get the NEWIRQAGR/NEWFIQAGR bit position
    LDR R1, INTC_CONTROL_ADDR
    STR R0, [R1] ; Write the NEWIRQAGR/NEWFIQAGR bit to allow new IRQs/FIQ
    ; Data Synchronization Barrier MOV R0, #0
    MCR P15, #0, R0, C7, C10, #4
    ; restore critical context
    MSR SPSR, R11 ; Restore the SPSR from R11
    LDMFD SP!, {R0-R12, LR} ; Restore working registers and Link register
    ; Return after handling the interrupt
    SUBS PC, LR, #4
8. After the ISR return, the ARM automatically restores its context as follows:
CPSR = SPSR
PC = LR

```

[Figure 6-2](#) shows the IRQ/FIQ processing sequence from the originating device peripheral module to the main program interruption.

The priority sorting mechanism is frozen during an interrupt processing sequence. If an interrupt condition occurs during this time, the interrupt is not lost. It is sorted when the NEWIRQAGR/NEWFIQAGR bit is set (priority sorting is reactivated).

**Figure 6-2. IRQ/FIQ Processing Sequence**


### 6.2.3 INTC Preemptive Processing Sequence

Preemptive interrupts, also called nested interrupts, can reduce the latencies for higher priority interrupts. A preemptive ISR can be suspended by a higher priority interrupt. Thus, the higher priority interrupt can be served immediately. Nested interrupts must be used carefully to avoid using corrupted data. Programmers must save corruptible registers and enable IRQ or FIQ at ARM side. IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in the code below.

To enable IRQ/FIQ preemption by higher priority IRQs/FIQs, programmers can follow this procedure to write the ISR.

At the beginning of an IRQ/FIQ ISR:

1. Save the ARM critical context registers.
2. Save the INTC\_THRESHOLD PRIORITYTHRESHOLD field before modifying it.
3. Read the active interrupt priority in the INTC\_IRQ\_PRIORITY IRQPRIORITY/INTC\_FIQ\_PRIORITY FIQPRIORITY field and write it to the PRIORITYTHRESHOLD(1) field.
4. Read the active interrupt number in the INTC\_SIR\_IRQ[6:0] ACTIVEIRQ/INTC\_SIR\_FIQ[6:0] ACTIVEFIQ field to identify the interrupt source.
5. Write 1 to the appropriate INTC\_CONTROL NEWIRQAGR and (2) NEWFIQAGR bit while an interrupt is still processing to allow only higher priority interrupts to preempt.
6. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/FIQs, a Data Synchronization Barrier is used. This operation ensure that the IRQ line is de-asserted before IRQ/FIQ enabling.
7. Enable IRQ/FIQ at ARM side.
8. Jump to the relevant subroutine handler.

The following sample code shows the previous steps:

**CAUTION**

The following code is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
; bit field mask to get only the bit field
ACTIVEPRI0_MASK .equ 0x7F
_IRQ_ISR:
; Step 1 : Save the critical context
STMFD SP!, {R0-R12, LR} ; Save working registers
MRS R11, SPSR ; Save the SPSR into R11
; Step 2 : Save the INTC_THRESHOLD register into R12
LDR R0, INTC_THRESHOLD_ADDR
LDR R12, [R0]
(1) The priority-
threshold mechanism is enabled automatically when writing a priority in the range of 0x00 to 0x7F. Writing a value of 0xFF (reset default) disables the priority-threshold mechanism. Values between 0x3F and 0xFF must not be used. When the hardware-priority threshold is in use, the priorities of interrupts selected as FIQ or IRQ become linked otherwise, they are independent. When they are linked, all FIQ priorities must be set higher than all IRQ priorities to maintain the relative priority of FIQ over IRQ.
(2) When handling FIQs using the priority-
threshold mechanism, both NEWFIQAGR and NEWIRQAGR bits must be written at the same time to ensure that the new priority threshold is applied while an IRQ sort is in progress. This IRQ will not have been seen by the ARM, as it will have been masked on entry to the FIQ ISR. However, the source of the IRQ remains active and it is finally processed when the priority threshold falls to a priority sufficiently low to allow it to be processed. The precaution of writing to New FIQ Agreement is not required during an IRQ ISR, as FIQ sorting is not affected (provided all FIQ priorities are higher than all IRQ priorities).
; Step 3 : Get the priority of the highest priority active IRQ
LDR R1, INTC_IRQ_PRIORITY_ADDR/INTC_FIQ_PRIORITY_ADDR
LDR R1, [R1] ; Get the INTC_IRQ_PRIORITY/INTC_FIQ_PRIORITY register
AND R1, R1, #ACTIVEPRI0_MASK ; Apply the mask to get the priority of the IRQ
STR R1, [R0] ; Write it to the INTC_THRESHOLD register
; Step 4 : Get the number of the highest priority active IRQ
LDR R10, INTC_SIR_IRQ_ADDR/INTC_SIR_FIQ_ADDR
LDR R10, [R10] ; Get the INTC_SIR_IRQ/INTC_SIR_FIQ register
AND R10, R10, #ACTIVEIRQ_MASK ; Apply the mask to get the active IRQ number
; Step 5 : Allow new IRQs and FIQs at INTC side
MOV R0, #0x1/0x3 ; Get the NEWIRQAGR and NEWFIQAGR bit position
LDR R1, INTC_CONTROL_ADDR
STR R0, [R1] ; Write the NEWIRQAGR and NEWFIQAGR bit
; Step 6 : Data Synchronization Barrier
MOV R0, #0 MCR P15, #0, R0, C7, C10, #4
; Step 7 : Read-modify-write the CPSR to enable IRQs/FIQs at ARM side
MRS R0, CPSR ; Read the status register
BIC R0, R0, #0x80/0x40 ; Clear the I/F bit
MSR CPSR, R0 ; Write it back to enable IRQs
; Step 8 : Jump to relevant subroutine handler
LDR PC, [PC, R10, lsl #2] ; PC base address points this instruction + 8
NOP ; To index the table by the PC
; Table of handler start addresses
.word IRQ0handler ;IRQ0 BANK0
.word IRQ1handler
.word IRQ2handler
```

After the return of the relevant IRQ/FIQ subroutine handle:

1. Disable IRQs/FIQs at ARM side.
2. Restore the INTC\_THRESHOLD PRIORITYTHRESHOLD field.
3. Restore the ARM critical context registers.

The following sample code shows the three previous steps:

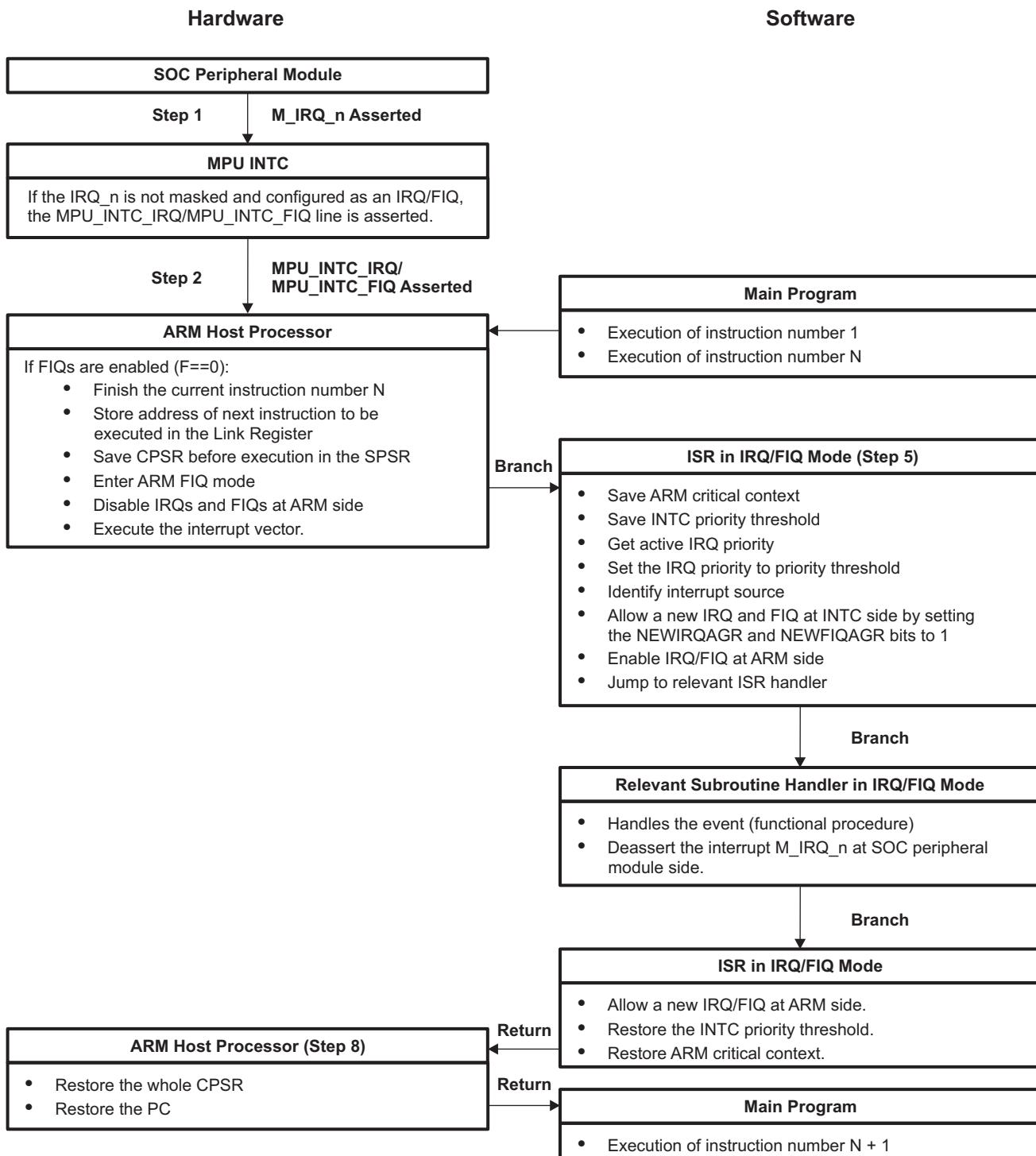
**CAUTION**

The following code is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
IRQ_ISR_end:  
    ; Step 1 : Read-modify-write the CPSR to disable IRQs/FIQs at ARM side  
    MRS R0, CPSR ; Read the CPSR  
    ORR R0, R0, #0x80/0x40 ; Set the I/F bit  
    MSR CPSR, R0 ; Write it back to disable IRQs  
    ; Step 2 : Restore the INTC_THRESHOLD register from R12  
    LDR R0, INTC_THRESHOLD_ADDR  
    STR R12, [R0]  
    ; Step 3 : Restore critical context  
    MSR SPSR, R11 ; Restore the SPSR from R11  
    LDMFD SP!, {R0-R12, LR} ; Restore working registers and Link register  
    ; Return after handling the interrupt  
    SUBS PC, LR, #4
```

Figure 6-3 shows the nested IRQ/FIQ processing sequence from the originating device peripheral module to the main program interruption.

Figure 6-3. Nested IRQ/FIQ Processing Sequence



### **6.2.4 Interrupt Preemption**

If wanting to enable pre-emption by higher priority interrupts, the ISR should read the active interrupt priority and write it to the priority threshold register. Writing a '1' to the appropriate NEW\_IRQ\_AGR or NEW\_FIQ\_AGR bits of the CONTROL register while still processing the interrupt will now allow only higher priority interrupts to pre-empt.

For each level of pre-emption, the programmer must save the threshold value before modifying it and restore it at the end of that ISR level.

The priority threshold mechanism is enabled automatically when writing a priority in the range of 0 to 7Fh as will be read from the IRQ\_PRIORITY and FIQ\_PRIORITY registers. Writing a value of FFh (reset default) disables the priority threshold mechanism.

When the hardware priority threshold is in use the priorities of interrupts selected as FIQ or IRQ become linked, otherwise they are independent. When linked, it is required that all FIQ priorities be set higher than all IRQ priorities to maintain the relative priority of FIQ over IRQ.

When handling FIQs using the priority threshold mechanism, it is required to write both New FIQ Agreement and New IRQ Agreement bits at the same time to cover the case that the new priority threshold is applied while an IRQ sorting is in progress. This IRQ will not have been seen by the ARM as it will have been masked on entry to the FIQ ISR. However, the source of the IRQ will remain active and it will be finally processed when the priority threshold falls to a low enough priority. The precaution of writing to New FIQ Agreement (as well as New IRQ Agreement) is not required during an IRQ ISR as FIQ sorting will not be affected (provided all FIQ priorities are higher than all IRQ priorities).

### **6.2.5 ARM A8 INTC Spurious Interrupt Handling**

The spurious flag indicates whether the result of the sorting (a window of 10 INTC functional clock cycles after the interrupt assertion) is invalid. The sorting is invalid if:

- The interrupt that triggered the sorting is no longer active during the sorting.
- A change in the mask has affected the result during the sorting time.

As a result, the values in the INTC\_MIRn, INTC\_ILRm, or INTC\_MIR\_SETn registers must not be changed while the corresponding interrupt is asserted. Only the active interrupt input that triggered the sort can be masked before it turns on the sort. If these registers are changed within the 10-cycle window after the interrupt assertion. The resulting values of the following registers become invalid:

- INTC\_SIR\_IRQ
- INTC\_SIR\_FIQ
- INTC\_IRQ\_PRIORITY
- INTC\_FIQ\_PRIORITY

This condition is detected for both IRQ and FIQ, and the invalid status is flagged across the SPURIOUSIRQFLAG (see NOTE 1) and SPURIOUSFIQFLAG (see NOTE 2) bit fields in the SIR and PRIORITY registers. A 0 indicates valid and a 1 indicates invalid interrupt number and priority. The invalid indication can be tested in software as a false register value.

**NOTE:**

1. The INTC\_SIR\_IRQ[31:7] SPURIOUSIRQFLAG bit field is a copy of the INTC\_IRQ\_PRIORITY[31:7] SPURIOUSIRQFLAG bit field.
2. The INTC\_SIR\_FIQ[31:7] SPURIOUSFIQFLAG bit field is a copy of the INTC\_FIQ\_PRIORITY[31:7] SPURIOUSFIQFLAG bit field.

## 6.3 ARM Cortex-A8 Interrupts

**Table 6-1. ARM Cortex-A8 Interrupts**

| Int Number | Acronym/name     | Source  | Signal Name                                    |
|------------|------------------|---|--|
| 0          | EMUINT           | MPU Subsystem Internal  | Emulation interrupt (EMUICINTR)                |
| 1          | COMMTX           | MPU Subsystem Internal  | CortexA8 COMMTX                                |
| 2          | COMMRX           | MPU Subsystem Internal  | CortexA8 COMMRX                                |
| 3          | BENCH            | MPU Subsystem Internal  | CortexA8 NPMUIRQ                               |
| 4          | ELM_IRQ          | ELM   | Sinterrupt (Error location process completion) |
| 5          | Reserved         |   |  |
| 6          | Reserved         |   |  |
| 7          | NMI              | External Pin (Primary Input)                                    | nmi_int  |
| 8          | Reserved         |   |  |
| 9          | L3DEBUG          | L3  | l3_FlagMux_top_FlagOut1                        |
| 10         | L3APPINT         | L3  | l3_FlagMux_top_FlagOut0                        |
| 11         | PRCMINT          | PRCM  | irq_mpu  |
| 12         | EDMACOMPINT      | TPCC (EDMA)   | tpcc_int_pend_p0                               |
| 13         | EDMAMPERR        | TPCC (EDMA)   | tpcc_mpint_pend_po                             |
| 14         | EDMAERRINT       | TPCC (EDMA)   | tpcc_errint_pend_po                            |
| 15         | Reserved         |   |  |
| 16         | ADC_TSC_GENINT   | ADC_TSC (Touchscreen Controller)                                | gen_intr_pend                                  |
| 17         | USBSSINT         | USBSS   | usbss_intr_pend                                |
| 18         | USBINT0          | USBSS   | usb0_intr_pend                                 |
| 19         | USBINT1          | USBSS   | usb1_intr_pend                                 |
| 20         | PRU_ICSS_EVTOUT0 | pr1_host[0] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr0_intr_pend                       |
| 21         | PRU_ICSS_EVTOUT1 | pr1_host[1] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr1_intr_pend                       |
| 22         | PRU_ICSS_EVTOUT2 | pr1_host[2] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr2_intr_pend                       |
| 23         | PRU_ICSS_EVTOUT3 | pr1_host[3] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr3_intr_pend                       |
| 24         | PRU_ICSS_EVTOUT4 | pr1_host[4] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr4_intr_pend                       |
| 25         | PRU_ICSS_EVTOUT5 | pr1_host[5] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr5_intr_pend                       |
| 26         | PRU_ICSS_EVTOUT6 | pr1_host[6] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr6_intr_pend                       |
| 27         | PRU_ICSS_EVTOUT7 | pr1_host[7] output/events exported from PRU-ICSS <sup>(1)</sup> | pr1_host_intr7_intr_pend                       |
| 28         | MMCSD1INT        | MMCSD1  | SINTERRUPTN                                    |
| 29         | MMCSD2INT        | MMCSD2  | SINTERRUPTN                                    |
| 30         | I2C2INT          | I2C2  | POINTRPEND                                     |
| 31         | eCAP0INT         | eCAP0 event/interrupt   | ecap_intr_intr_pend                            |
| 32         | GPIOINT2A        | GPIO 2  | POINTRPEND1                                    |
| 33         | GPIOINT2B        | GPIO 2  | POINTRPEND2                                    |
| 34         | USBWAKEUP        | USBSS   | slv0p_Swakeup                                  |
| 35         | Reserved         |   |  |
| 36         | LCDCINT          | LCDC  | lcd_irq  |

<sup>(1)</sup> pr1\_host\_intr[0:7] corresponds to Host-2 to Host-9 of the PRU-ICSS interrupt controller.

**Table 6-1. ARM Cortex-A8 Interrupts (continued)**

| Int Number | Acronym/name                  | Source                               | Signal Name               |
|------------|-------------------------------|--------------------------------------|---------------------------|
| 37         | GFXINT                        | SGX530                               | THALIAIRQ                 |
| 38         | Reserved                      |                                      |                           |
| 39         | ePWM2INT                      | eHRPWM2 (PWM Subsystem)              | epwm_intr_intr_pend       |
| 40         | 3PGSWRXTHR0 (RX_THRESH_PULSE) | CPSW (Ethernet)                      | c0_rx_thresh_pend         |
| 41         | 3PGSWRXINT0 (RX_PULSE)        | CPSW (Ethernet)                      | c0_rx_pend                |
| 42         | 3PGSWTXINT0 (TX_PULSE)        | CPSW (Ethernet)                      | c0_tx_pend                |
| 43         | 3PGSWMISC0 (MISC_PULSE)       | CPSW (Ethernet)                      | c0_misc_pend              |
| 44         | UART3INT                      | UART3                                | niq                       |
| 45         | UART4INT                      | UART4                                | niq                       |
| 46         | UART5INT                      | UART5                                | niq                       |
| 47         | eCAP1INT                      | eCAP1 (PWM Subsystem)                | ecap_intr_intr_pend       |
| 48         | Reserved                      |                                      |                           |
| 49         | Reserved                      |                                      |                           |
| 50         | Reserved                      |                                      |                           |
| 51         | Reserved                      |                                      |                           |
| 52         | DCAN0_INT0                    | DCAN0                                | dcan_intr0_intr_pend      |
| 53         | DCAN0_INT1                    | DCAN0                                | dcan_intr1_intr_pend      |
| 54         | DCAN0_PARITY                  | DCAN0                                | dcan_uerr_intr_pend       |
| 55         | DCAN1_INT0                    | DCAN1                                | dcan_intr0_intr_pend      |
| 56         | DCAN1_INT1                    | DCAN1                                | dcan_intr1_intr_pend      |
| 57         | DCAN1_PARITY                  | DCAN1                                | dcan_uerr_intr_pend       |
| 58         | ePWM0_TZINT                   | eHRPWM0 TZ interrupt (PWM Subsystem) | epwm_tz_intr_pend         |
| 59         | ePWM1_TZINT                   | eHRPWM1 TZ interrupt (PWM Subsystem) | epwm_tz_intr_pend         |
| 60         | ePWM2_TZINT                   | eHRPWM2 TZ interrupt (PWM Subsystem) | epwm_tz_intr_pend         |
| 61         | eCAP2INT                      | eCAP2 (PWM Subsystem)                | ecap_intr_intr_pend       |
| 62         | GPIOINT3A                     | GPIO 3                               | POINTRPEND1               |
| 63         | GPIOINT3B                     | GPIO 3                               | POINTRPEND2               |
| 64         | MMCSD0INT                     | MMCSD0                               | SINTERRUPTN               |
| 65         | McSPI0INT                     | McSPI0                               | SINTERRUPTN               |
| 66         | TINT0                         | Timer0                               | POINTR_PEND               |
| 67         | TINT1_1MS                     | DMTIMER_1ms                          | POINTR_PEND               |
| 68         | TINT2                         | DMTIMER2                             | POINTR_PEND               |
| 69         | TINT3                         | DMTIMER3                             | POINTR_PEND               |
| 70         | I2C0INT                       | I2C0                                 | POINTRPEND                |
| 71         | I2C1INT                       | I2C1                                 | POINTRPEND                |
| 72         | UART0INT                      | UART0                                | niq                       |
| 73         | UART1INT                      | UART1                                | niq                       |
| 74         | UART2INT                      | UART2                                | niq                       |
| 75         | RTCINT                        | RTC                                  | timer_intr_pend           |
| 76         | RTCALARMINT                   | RTC                                  | alarm_intr_pend           |
| 77         | MBINT0                        | Mailbox0 (mail_u0_irq)               | initiator_sinterrupt_q_n0 |
| 78         | M3_TXEV                       | Wake M3 Subsystem                    | TXEV                      |
| 79         | eQEP0INT                      | eQEP0 (PWM Subsystem)                | eqep_intr_intr_pend       |
| 80         | MCATXINT0                     | McASP0                               | mcasp_x_intr_pend         |
| 81         | MCARXINT0                     | McASP0                               | mcasp_r_intr_pend         |

**Table 6-1. ARM Cortex-A8 Interrupts (continued)**

| Int Number | Acronym/name              | Source   | Signal Name          |
|------------|---------------------------|--|----------------------|
| 82         | MCATXINT1                 | McASP1   | mcasp_x_intr_pend    |
| 83         | MCARXINT1                 | McASP1   | mcasp_r_intr_pend    |
| 84         | Reserved                  |  |                      |
| 85         | Reserved                  |  |                      |
| 86         | ePWM0INT                  | eHRPWM0 (PWM Subsystem)                        | epwm_intr_intr_pend  |
| 87         | ePWM1INT                  | eHRPWM1 (PWM Subsystem)                        | epwm_intr_intr_pend  |
| 88         | eQEP1INT                  | eQEP1 (PWM Subsystem)                          | eqep_intr_intr_pend  |
| 89         | eQEP2INT                  | eQEP2 (PWM Subsystem)                          | eqep_intr_intr_pend  |
| 90         | DMA_INTR_PIN2             | External DMA/Interrupt Pin2 (xdma_event_intr2) | pi_x_dma_event_intr2 |
| 91         | WDT1INT (Public Watchdog) | WDTIMER1                                       | PO_INT_PEND          |
| 92         | TINT4                     | DMTIMER4                                       | POINTR_PEND          |
| 93         | TINT5                     | DMTIMER5                                       | POINTR_PEND          |
| 94         | TINT6                     | DMTIMER6                                       | POINTR_PEND          |
| 95         | TINT7                     | DMTIMER7                                       | POINTR_PEND          |
| 96         | GPIOINT0A                 | GPIO 0   | POINTRPEND1          |
| 97         | GPIOINT0B                 | GPIO 0   | POINTRPEND2          |
| 98         | GPIOINT1A                 | GPIO 1   | POINTRPEND1          |
| 99         | GPIOINT1B                 | GPIO 1   | POINTRPEND2          |
| 100        | GPMCINT                   | GPMC   | gpmc_sinterrupt      |
| 101        | DDRERR0                   | EMIF   | sys_err_intr_pend    |
| 102        | Reserved                  |  |                      |
| 103        | Reserved                  |  |                      |
| 104        | Reserved                  |  |                      |
| 105        | Reserved                  |  |                      |
| 106        | Reserved                  |  |                      |
| 107        | Reserved                  |  |                      |
| 108        | Reserved                  |  |                      |
| 109        | Reserved                  |  |                      |
| 110        | Reserved                  |  |                      |
| 111        | Reserved                  |  |                      |
| 112        | TCERRINT0                 | TPTC0  | tptc_erint_pend_po   |
| 113        | TCERRINT1                 | TPTC1  | tptc_erint_pend_po   |
| 114        | TCERRINT2                 | TPTC2  | tptc_erint_pend_po   |
| 115        | ADC_TSC_PENINT            | ADC_TSC  | pen_intr_pend        |
| 116        | Reserved                  |  |                      |
| 117        | Reserved                  |  |                      |
| 118        | Reserved                  |  |                      |
| 119        | Reserved                  |  |                      |
| 120        | SMRFLX_MPU subsystem      | Smart Reflex 0                                 | intrpend             |
| 121        | SMRFLX_Core               | Smart Reflex 1                                 | intrpend             |
| 122        | Reserved                  |  |                      |
| 123        | DMA_INTR_PIN0             | External DMA/Interrupt Pin0 (xdma_event_intr0) | pi_x_dma_event_intr0 |
| 124        | DMA_INTR_PIN1             | External DMA/Interrupt Pin1 (xdma_event_intr1) | pi_x_dma_event_intr1 |
| 125        | McSPI1INT                 | McSPI1   | SINTERRUPTN          |
| 126        | Reserved                  |  |                      |

**Table 6-1. ARM Cortex-A8 Interrupts (continued)**

| Int Number | Acronym/name | Source | Signal Name |
|------------|--------------|--------|-------------|
| 127        | Reserved     |        |             |

## 6.4 ARM Cortex-M3 Interrupts

**Table 6-2. ARM Cortex-M3 Wakeup Processor Interrupts**

| Int Number | Acronym/Name        | Source                      | Signal Name               |
|------------|---------------------|-----------------------------|---------------------------|
| 0          | Reset               | M3                          |                           |
| 1          | NMI                 | M3                          |                           |
| 2          | Hard Fault          | M3                          |                           |
| 3          | Mem Manager Fault   | M3                          |                           |
| 4          | Bus Fault           | M3                          |                           |
| 5          | Usage Fault         | M3                          |                           |
| 6          | Reserved            |                             |                           |
| 7          | Reserved            |                             | M3 Internal Interrupts    |
| 8          | Reserved            |                             |                           |
| 9          | Reserved            |                             |                           |
| 10         | SVCall              | M3                          |                           |
| 11         | Dbug Mon            | M3                          |                           |
| 12         | Reserved            |                             |                           |
| 13         | PendSV              | M3                          |                           |
| 14         | SYSTICK             | M3                          |                           |
| 15         | Reserved            |                             |                           |
| 16         | PRCM_M3_IRQ1        | PRCM                        | irq_wkup_m3               |
| 17         | UART0INT            | UART0 (L4-WKUP)             | nirq                      |
| 18         | TINT0               | TIMER0 (L4_WKUP)            | POINTR_PEND               |
| 19         | TINT1_1MS           | TIMER1 (L4_WKUP)            | POINTR_PEND               |
| 20         | Reserved            |                             |                           |
| 21         | WDT1INT             | WDTimer1 (L4_WKUP)          | PO_INT_PEND               |
| 22         | I2C10NT             | I2C0 (L4_WKUP)              | POINTRPEND                |
| 23         | GPIOINT0A           | GPIO0 (L4_WKUP)             | POINTRPEND1               |
| 24         | GPIOINT0A           | GPIO0 (L4_WKUP)             | POINTRPEND2               |
| 25         | Reserved            |                             |                           |
| 26         | ADC_TSC_INT1        | ADC_TSC (L4_WKUP)           | gen_intr_pend             |
| 27         | RTCINT              | RTC                         | timer_intr_pend           |
| 28         | RTCALARMINT         | RTC                         | alarm_intr_pend           |
| 29         | SMRFLX_MP subsystem | SR0                         | intrpend                  |
| 30         | SMRFLX_Core         | SR1                         | intrpend                  |
| 31         | MBINT0              | Mailbox0 (mail_u3_irq)      | initiator_sinterrupt_q_n3 |
| 32         | Reserved            |                             |                           |
| 33         | USBSWAKEUP          | USBSS                       | slv0p_Swakeup             |
| 34         | PRCM_M3_IRQ2        | PRCM                        | irq_mpuisgated_wkup_m3    |
| 35         | USB0WUOUT           | USBSS                       | usb0_wuout                |
| 36         | USB1WUOUT           | USBSS                       | usb1_wuout                |
| 37         | DMA_INTR_PIN1       | External DMA/Interrupt Pin1 | pi_x_dma_event_intr1      |
| 38         | DMA_INTR_PIN2       | External DMA/Interrupt Pin2 | pi_x_dma_event_intr2      |
| 39         | Reserved            |                             |                           |
| 40         | I2C0_WAKE           | I2C0 (L4_WKUP)              | POINTRSWAKEUP             |
| 41         | RTCTIMER_WAKE       | RTC Timer Wake              | slvp_timer_SWakeup        |
| 42         | RTCALARM_WAKE       | RTC Alarm Wake              | slvp_alarm_SWakeup        |
| 43         | TIMER0_WAKE         | TIMER0 (L4_WKUP)            | POINTR_SWAKEUP            |
| 44         | TIMER1_WAKE         | TIMER1 (L4_WKUP)            | PO_SWAKEUP_OUT            |

**Table 6-2. ARM Cortex-M3 Wakeup Processor Interrupts (continued)**

| Int Number | Acronym/Name | Source             | Signal Name          |
|------------|--------------|--------------------|----------------------|
| 45         | UART0_WAKE   | UART0 (L4-WKUP)    | swakeup              |
| 46         | GPIO0_WAKE0  | GPIO0 (L4_WKUP)    | POINTRSWAKEUP1       |
| 47         | GPIO0_WAKE1  | GPIO0 (L4_WKUP)    | POINTRSWAKEUP2       |
| 48         | MPU_WAKE     | MPU subsystem      | POMWAKEUP            |
| 49         | Reserved     |                    |                      |
| 50         | WDT1_WAKE    | WDTimer1 (L4_WKUP) | PO_INT_WAKEUP        |
| 51         | ADCTSC_WAKE  | ADC_TSC (L4_WKUP)  | pwr_idle_prm_Swakeup |

## 6.5 PWM Events

**Table 6-3. Timer and eCAP Event Capture**

| Event # | IP   | Interrupt Name/Pin |
|---------|--|--------------------|
| 0       | For Timer 5 MUX input from IO signal<br>TIMER5 | TIMER5 IO pin      |
|         | For Timer 6 MUX input from IO signal<br>TIMER6 | TIMER6 IO pin      |
|         | For Timer 7 MUX input from IO signal<br>TIMER7 | TIMER7 IO pin      |
|         | For eCAP 0 MUX input from IO signal<br>eCAP0   | eCAP0 IO pin       |
|         | For eCAP 1 MUX input from IO signal<br>eCAP1   | eCAP1 IO pin       |
|         | For eCAP 2 MUX input from IO signal<br>eCAP2   | eCAP2 IO pin       |
| 1       | UART0  | UART0INT           |
| 2       | UART1  | UART1INT           |
| 3       | UART2  | UART2INT           |
| 4       | UART3  | UART3INT           |
| 5       | UART4  | UART4INT           |
| 6       | UART5  | UART5INT           |
| 7       | 3PGSW  | 3PGSWRXTHR0        |
| 8       | 3PGSW  | 3PGSWRXINT0        |
| 9       | 3PGSW  | 3PGSWTXINT0        |
| 10      | 3PGSW  | 3PGSWMISC0         |
| 11      | McASP0   | MCATXINT0          |
| 12      | McASP0   | MCARXINT0          |
| 13      | McASP1   | MCATXINT1          |
| 14      | McASP1   | MCARXINT1          |
| 15      | Reserved                                       | Reserved           |
| 16      | Reserved                                       | Reserved           |
| 17      | GPIO 0   | GPIOINT0A          |
| 18      | GPIO 0   | GPIOINT0B          |
| 19      | GPIO 1   | GPIOINT1A          |
| 20      | GPIO 1   | GPIOINT1B          |
| 21      | GPIO 2   | GPIOINT2A          |
| 22      | GPIO 2   | GPIOINT2B          |
| 23      | GPIO 3   | GPIOINT3A          |
| 24      | GPIO 3   | GPIOINT3B          |
| 25      | DCAN0  | DCAN0_INT0         |
| 26      | DCAN0  | DCAN0_INT1         |
| 27      | DCAN0  | DCAN0_PARITY       |
| 28      | DCAN1  | DCAN1_INT0         |
| 29      | DCAN1  | DCAN1_INT1         |
| 30      | DCAN1  | DCAN1_PARITY       |

## 6.6 Interrupt Controller Registers

**NOTE:** FIQ is not available on general-purpose (GP) devices.

### 6.6.1 INTC Registers

Table 6-4 lists the memory-mapped registers for the INTC. All register offset addresses not listed in Table 6-4 should be considered as reserved locations and the register contents should not be modified.

**Table 6-4. INTC REGISTERS**

| Offset | Acronym           | Register Name | Section          |
|--------|-------------------|---------------|------------------|
| 0h     | INTC_REVISION     |               | Section 6.6.1.1  |
| 10h    | INTC_SYSCONFIG    |               | Section 6.6.1.2  |
| 14h    | INTC_SYSSTATUS    |               | Section 6.6.1.3  |
| 40h    | INTC_SIR_IRQ      |               | Section 6.6.1.4  |
| 44h    | INTC_SIR_FIQ      |               | Section 6.6.1.5  |
| 48h    | INTC_CONTROL      |               | Section 6.6.1.6  |
| 4Ch    | INTC_PROTECTION   |               | Section 6.6.1.7  |
| 50h    | INTC_IDLE         |               | Section 6.6.1.8  |
| 60h    | INTC_IRQ_PRIORITY |               | Section 6.6.1.9  |
| 64h    | INTC_FIQ_PRIORITY |               | Section 6.6.1.10 |
| 68h    | INTC_THRESHOLD    |               | Section 6.6.1.11 |
| 80h    | INTC_ITR0         |               | Section 6.6.1.12 |
| 84h    | INTC_MIRO         |               | Section 6.6.1.13 |
| 88h    | INTC_MIR_CLEAR0   |               | Section 6.6.1.14 |
| 8Ch    | INTC_MIR_SET0     |               | Section 6.6.1.15 |
| 90h    | INTC_ISR_SET0     |               | Section 6.6.1.16 |
| 94h    | INTC_ISR_CLEAR0   |               | Section 6.6.1.17 |
| 98h    | INTC_PENDING_IRQ0 |               | Section 6.6.1.18 |
| 9Ch    | INTC_PENDING_FIQ0 |               | Section 6.6.1.19 |
| A0h    | INTC_ITR1         |               | Section 6.6.1.20 |
| A4h    | INTC_MIR1         |               | Section 6.6.1.21 |
| A8h    | INTC_MIR_CLEAR1   |               | Section 6.6.1.22 |
| ACh    | INTC_MIR_SET1     |               | Section 6.6.1.23 |
| B0h    | INTC_ISR_SET1     |               | Section 6.6.1.24 |
| B4h    | INTC_ISR_CLEAR1   |               | Section 6.6.1.25 |
| B8h    | INTC_PENDING_IRQ1 |               | Section 6.6.1.26 |
| BCh    | INTC_PENDING_FIQ1 |               | Section 6.6.1.27 |
| C0h    | INTC_ITR2         |               | Section 6.6.1.28 |
| C4h    | INTC_MIR2         |               | Section 6.6.1.29 |
| C8h    | INTC_MIR_CLEAR2   |               | Section 6.6.1.30 |
| CCh    | INTC_MIR_SET2     |               | Section 6.6.1.31 |
| D0h    | INTC_ISR_SET2     |               | Section 6.6.1.32 |
| D4h    | INTC_ISR_CLEAR2   |               | Section 6.6.1.33 |
| D8h    | INTC_PENDING_IRQ2 |               | Section 6.6.1.34 |
| DCh    | INTC_PENDING_FIQ2 |               | Section 6.6.1.35 |
| E0h    | INTC_ITR3         |               | Section 6.6.1.36 |
| E4h    | INTC_MIR3         |               | Section 6.6.1.37 |
| E8h    | INTC_MIR_CLEAR3   |               | Section 6.6.1.38 |
| ECh    | INTC_MIR_SET3     |               | Section 6.6.1.39 |

**Table 6-4. INTC REGISTERS (continued)**

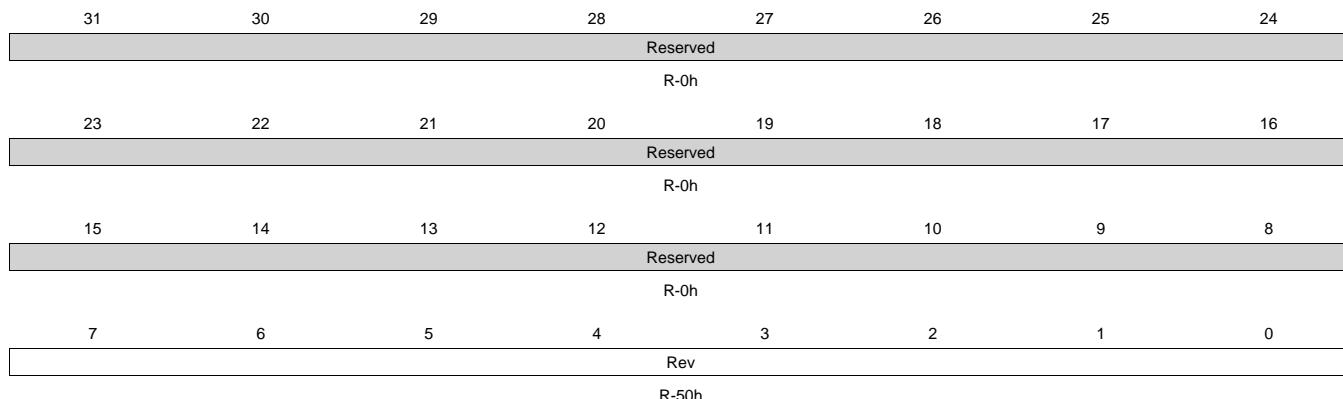
| Offset          | Acronym                  | Register Name | Section                          |
|-----------------|--------------------------|---------------|----------------------------------|
| F0h             | INTC_ISR_SET3            |               | <a href="#">Section 6.6.1.40</a> |
| F4h             | INTC_ISR_CLEAR3          |               | <a href="#">Section 6.6.1.41</a> |
| F8h             | INTC_PENDING_IRQ3        |               | <a href="#">Section 6.6.1.42</a> |
| FC <sub>h</sub> | INTC_PENDING_FIQ3        |               | <a href="#">Section 6.6.1.43</a> |
| 100h to 2FCh    | INTC_ILR0 to INTC_ILR127 |               | <a href="#">Section 6.6.1.44</a> |

### 6.6.1.1 INTC\_REVISION Register (offset = 0h) [reset = 50h]

INTC\_REVISION is shown in [Figure 6-4](#) and described in [Table 6-5](#).

This register contains the IP revision code

**Figure 6-4. INTC\_REVISION Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-5. INTC\_REVISION Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    | Reads returns 0  |
| 7-0  | Rev      | R    | 50h   | IP revision<br>[7:4] Major revision<br>[3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1 |

### 6.6.1.2 INTC\_SYSCONFIG Register (offset = 10h) [reset = 0h]

INTC\_SYSCONFIG is shown in [Figure 6-5](#) and described in [Table 6-6](#).

This register controls the various parameters of the OCP interface

**Figure 6-5. INTC\_SYSCONFIG Register**

|          |    |    |          |          |          |           |          |
|----------|----|----|----------|----------|----------|-----------|----------|
| 31       | 30 | 29 | 28       | 27       | 26       | 25        | 24       |
| Reserved |    |    |          |          |          |           |          |
| 23       | 22 | 21 | 20       | 19       | 18       | 17        | 16       |
| Reserved |    |    |          |          |          |           |          |
| 15       | 14 | 13 | 12       | 11       | 10       | 9         | 8        |
| Reserved |    |    |          |          |          |           |          |
| 7        | 6  | 5  | 4        | 3        | 2        | 1         | 0        |
| Reserved |    |    | Reserved | Reserved | Reserved | SoftReset | Autoidle |
|          |    |    | R-0h     |          | R/W-0h   |           | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-6. INTC\_SYSCONFIG Register Field Descriptions**

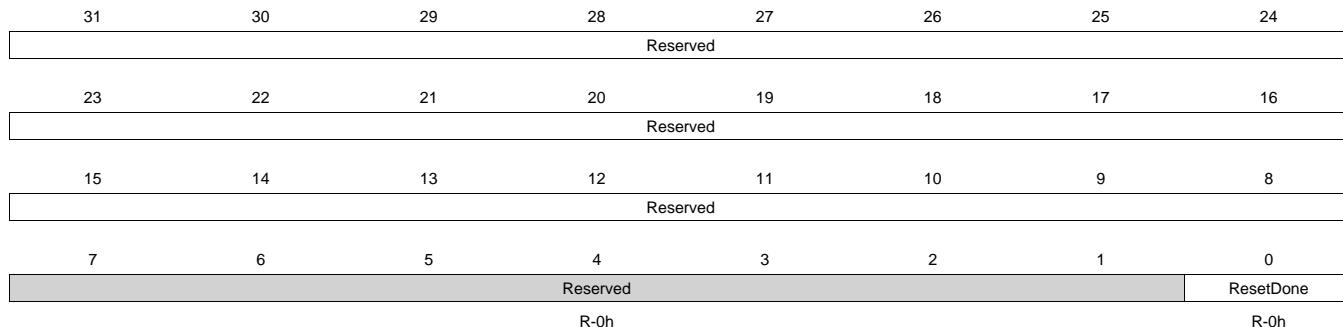
| Bit | Field     | Type | Reset | Description   |
|-----|-----------|------|-------|---|
| 4-3 | Reserved  | R    | 0h    | Write 0's for future compatibility.<br>Reads returns 0  |
| 1   | SoftReset | R/W  | 0h    | Software reset.<br>Set this bit to trigger a module reset.<br>The bit is automatically reset by the hardware.<br>During reads, it always returns 0.<br>0x0(Read) = always_Always returns 0<br>0x1(Read) = never_never happens |
| 0   | Autoidle  | R/W  | 0h    | Internal OCP clock gating strategy<br>0x0 = clkfree : OCP clock is free running<br>0x1 = autoClkGate : Automatic OCP clock gating strategy is applied, based on the OCP interface activity                                    |

### 6.6.1.3 INTC\_SYSSTATUS Register (offset = 14h) [reset = 0h]

INTC\_SYSSTATUS is shown in [Figure 6-6](#) and described in [Table 6-7](#).

This register provides status information about the module

**Figure 6-6. INTC\_SYSSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-7. INTC\_SYSSTATUS Register Field Descriptions**

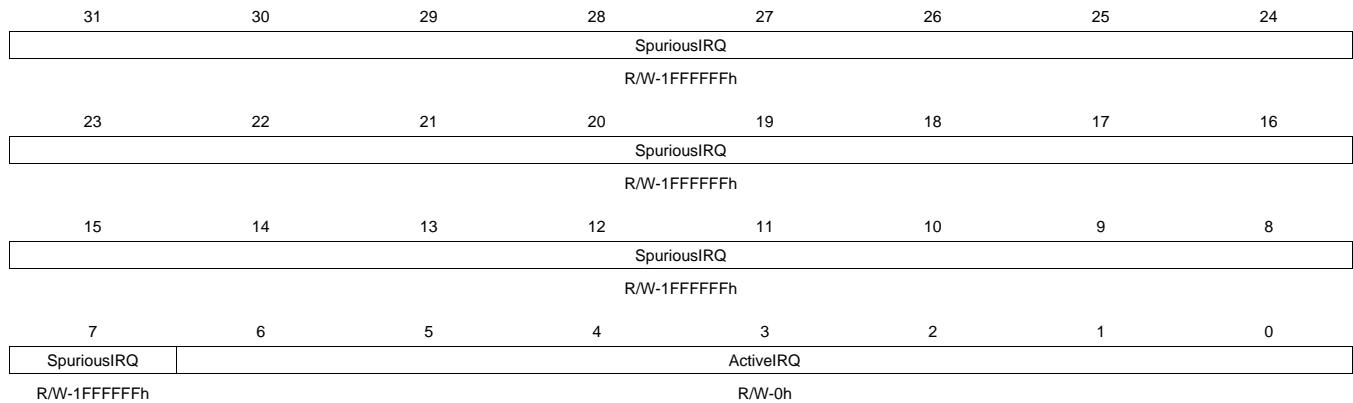
| Bit | Field     | Type | Reset | Description  |
|-----|-----------|------|-------|--|
| 7-1 | Reserved  | R    | 0h    | Reserved for OCP socket status information Read returns 0  |
| 0   | ResetDone | R    | 0h    | Internal reset monitoring<br>0x0 = rstOngoing : Internal module reset is on-going<br>0x1 = rstComp : Reset completed |

#### 6.6.1.4 INTC\_SIR\_IRQ Register (offset = 40h) [reset = FFFFFFF80h]

INTC\_SIR\_IRQ is shown in [Figure 6-7](#) and described in [Table 6-8](#).

This register supplies the currently active IRQ interrupt number.

**Figure 6-7. INTC\_SIR\_IRQ Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-8. INTC\_SIR\_IRQ Register Field Descriptions**

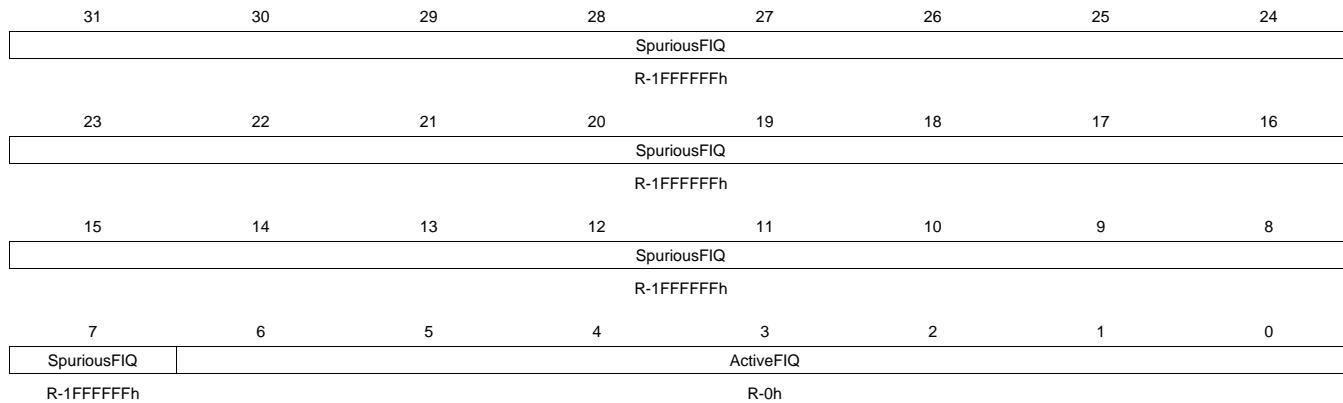
| Bit  | Field       | Type | Reset    | Description       |
|------|-------------|------|----------|-------------------|
| 31-7 | SpuriousIRQ | R/W  | 1FFFFFFh | Spurious IRQ flag |
| 6-0  | ActiveIRQ   | R/W  | 0h       | Active IRQ number |

### 6.6.1.5 INTC\_SIR\_FIQ Register (offset = 44h) [reset = FFFFFFF80h]

INTC\_SIR\_FIQ is shown in [Figure 6-8](#) and described in [Table 6-9](#).

This register supplies the currently active FIQ interrupt number

**Figure 6-8. INTC\_SIR\_FIQ Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-9. INTC\_SIR\_FIQ Register Field Descriptions**

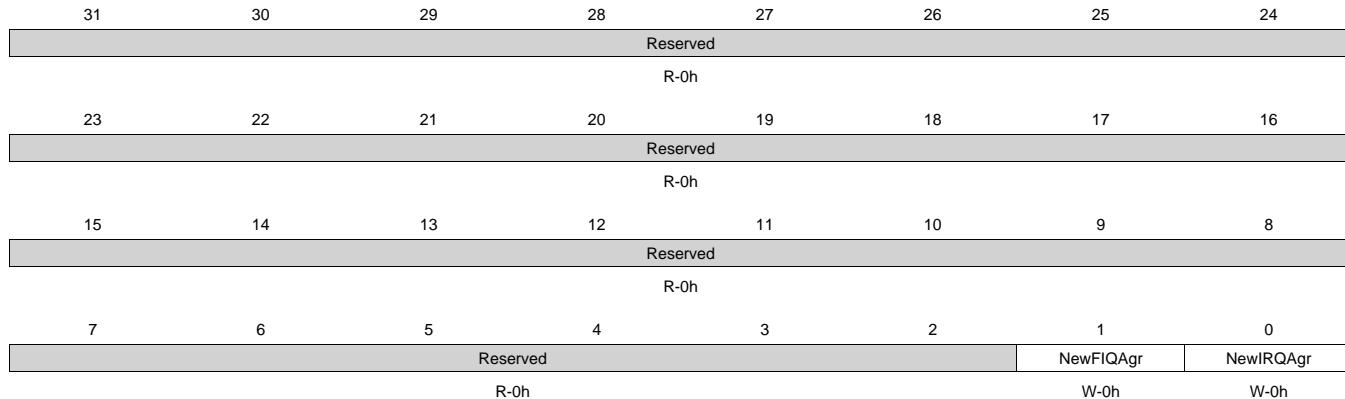
| Bit  | Field       | Type | Reset    | Description       |
|------|-------------|------|----------|-------------------|
| 31-7 | SpuriousFIQ | R    | 1FFFFFFh | Spurious FIQ flag |
| 6-0  | ActiveFIQ   | R    | 0h       | Active FIQ number |

### 6.6.1.6 INTC\_CONTROL Register (offset = 48h) [reset = 0h]

INTC\_CONTROL is shown in [Figure 6-9](#) and described in [Table 6-10](#).

This register contains the new interrupt agreement bits

**Figure 6-9. INTC\_CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-10. INTC\_CONTROL Register Field Descriptions**

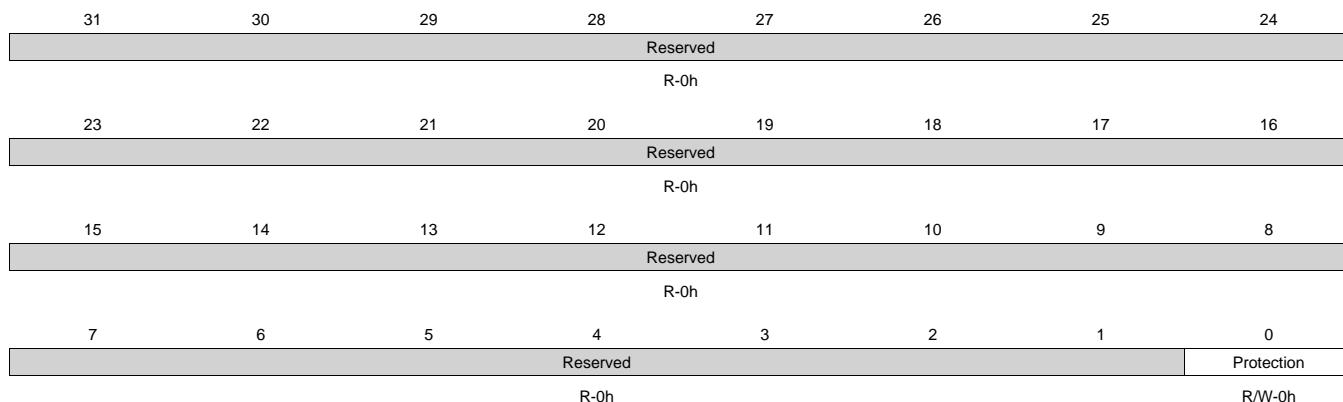
| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 31-2 | Reserved  | R    | 0h    | Write 0's for future compatibility.<br>Reads returns 0  |
| 1    | NewFIQAgr | W    | 0h    | Reset FIQ output and enable new FIQ generation<br>0x0(Write) = nofun_no function effect<br>0x1(Write) = NewFIQ_Reset FIQ output and enable new FIQ generation |
| 0    | NewIRQAgr | W    | 0h    | New IRQ generation<br>0x0(Write) = nofun_no function effect<br>0x1(Write) = NewIRQ_Reset IRQ output and enable new IRQ generation                             |

### 6.6.1.7 INTC\_PROTECTION Register (offset = 4Ch) [reset = 0h]

INTC\_PROTECTION is shown in [Figure 6-10](#) and described in [Table 6-11](#).

This register controls protection of the other registers. This register can only be accessed in privileged mode, regardless of the current value of the protection bit.

**Figure 6-10. INTC\_PROTECTION Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-11. INTC\_PROTECTION Register Field Descriptions**

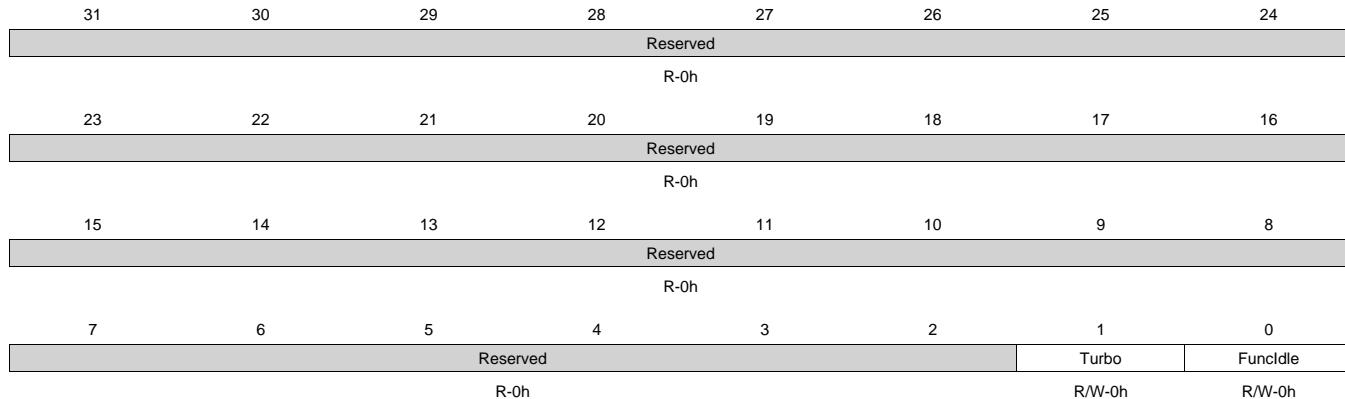
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-1 | Reserved   | R    | 0h    | Write 0's for future compatibility.<br>Reads returns 0   |
| 0    | Protection | R/W  | 0h    | Protection mode<br>0x0 = ProtMDis : Protection mode disabled (default)<br>0x1 = ProtMEnb : Protection mode enabled. When enabled, only privileged mode can access registers. |

### 6.6.1.8 INTC\_IDLE Register (offset = 50h) [reset = 0h]

INTC\_IDLE is shown in [Figure 6-11](#) and described in [Table 6-12](#).

This register controls the clock auto-idle for the functional clock and the input synchronisers

**Figure 6-11. INTC\_IDLE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-12. INTC\_IDLE Register Field Descriptions**

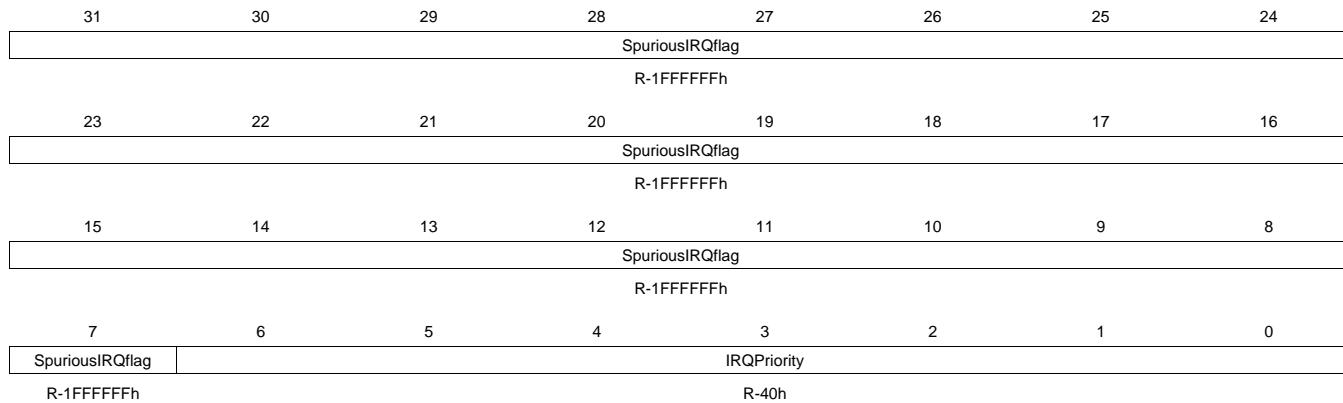
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    | Write 0's for future compatibility.<br>Reads returns 0   |
| 1    | Turbo    | R/W  | 0h    | Input synchroniser clock auto-gating<br>0x0 = SyncFree : Input synchroniser clock is free running (default)<br>0x1 = SyncAuto : Input synchroniser clock is auto-gated based on interrupt input activity |
| 0    | Funcidle | R/W  | 0h    | Functional clock auto-idle mode<br>0x0 = FuncAuto : Functional clock gating strategy is applied (default)<br>0x1 = FuncFree : Functional clock is free-running   |

### 6.6.1.9 INTC\_IRQ\_PRIORITY Register (offset = 60h) [reset = FFFFFFFC0h]

INTC\_IRQ\_PRIORITY is shown in [Figure 6-12](#) and described in [Table 6-13](#).

This register supplies the currently active IRQ priority level

**Figure 6-12. INTC\_IRQ\_PRIORITY Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-13. INTC\_IRQ\_PRIORITY Register Field Descriptions**

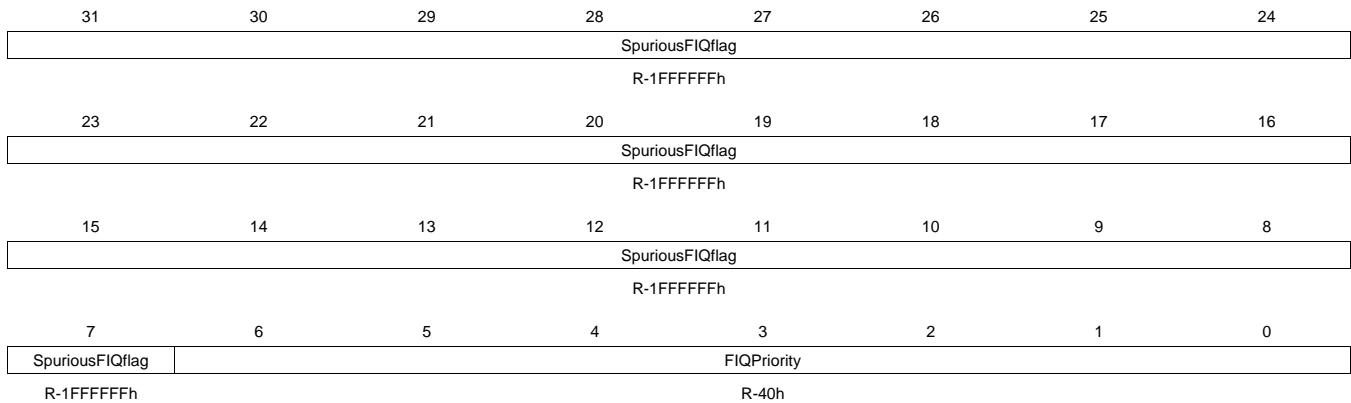
| Bit  | Field           | Type | Reset    | Description          |
|------|-----------------|------|----------|----------------------|
| 31-7 | SpuriousIRQflag | R    | 1FFFFFFh | Spurious IRQ flag    |
| 6-0  | IRQPriority     | R    | 40h      | Current IRQ priority |

### 6.6.1.10 INTC\_FIQ\_PRIORITY Register (offset = 64h) [reset = FFFFFFFC0h]

INTC\_FIQ\_PRIORITY is shown in [Figure 6-13](#) and described in [Table 6-14](#).

This register supplies the currently active FIQ priority level

**Figure 6-13. INTC\_FIQ\_PRIORITY Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-14. INTC\_FIQ\_PRIORITY Register Field Descriptions**

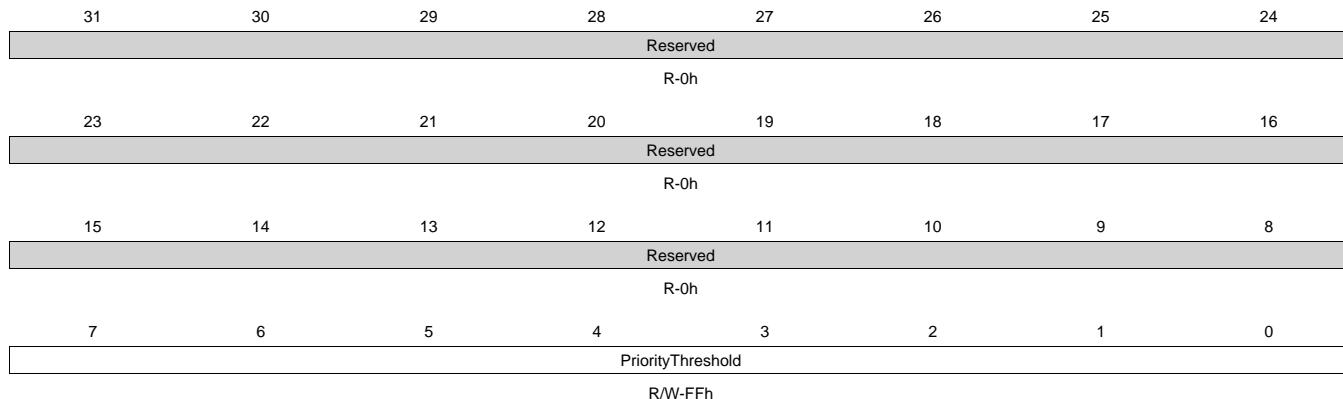
| Bit  | Field           | Type | Reset    | Description          |
|------|-----------------|------|----------|----------------------|
| 31-7 | SpuriousFIQflag | R    | 1FFFFFFh | Spurious FIQ flag    |
| 6-0  | FIQPriority     | R    | 40h      | Current FIQ priority |

### 6.6.1.11 INTC\_THRESHOLD Register (offset = 68h) [reset = FFh]

INTC\_THRESHOLD is shown in [Figure 6-14](#) and described in [Table 6-15](#).

This register sets the priority threshold

**Figure 6-14. INTC\_THRESHOLD Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-15. INTC\_THRESHOLD Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-8 | Reserved          | R    | 0h    | Reads returns 0   |
| 7-0  | PriorityThreshold | R/W  | FFh   | Priority threshold (used values 0x00-0x1f or 0x00-0x3f, 0xff disables the threshold). |

### 6.6.1.12 INTC\_ITR0 Register (offset = 80h) [reset = 0h]

INTC\_ITR0 is shown in [Figure 6-15](#) and described in [Table 6-16](#).

This register shows the raw interrupt input status before masking

**Figure 6-15. INTC\_ITR0 Register**

| 31   | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Itr  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-16. INTC\_ITR0 Register Field Descriptions**

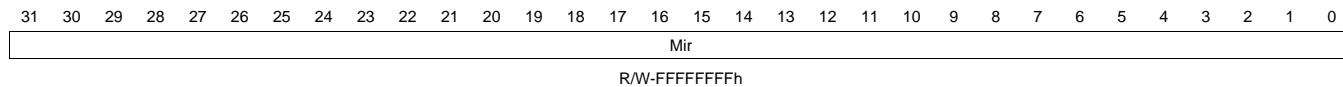
| Bit  | Field | Type | Reset | Description                     |
|------|-------|------|-------|---------------------------------|
| 31-0 | Itr   | R    | 0h    | Interrupt status before masking |

### 6.6.1.13 INTC\_MIR0 Register (offset = 84h) [reset = FFFFFFFFh]

INTC\_MIR0 is shown in [Figure 6-16](#) and described in [Table 6-17](#).

This register contains the interrupt mask

**Figure 6-16. INTC\_MIR0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-17. INTC\_MIR0 Register Field Descriptions**

| Bit  | Field | Type | Reset    | Description    |
|------|-------|------|----------|----------------|
| 31-0 | Mir   | R/W  | FFFFFFFh | Interrupt mask |

### 6.6.1.14 INTC\_MIR\_CLEAR0 Register (offset = 88h) [reset = 0h]

INTC\_MIR\_CLEAR0 is shown in [Figure 6-17](#) and described in [Table 6-18](#).

This register is used to clear the interrupt mask bits.

**Figure 6-17. INTC\_MIR\_CLEAR0 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MirClear |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-18. INTC\_MIR\_CLEAR0 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                                      |
|------|----------|------|-------|--|
| 31-0 | MirClear | W    | 0h    | Write 1 clears the mask bit to 0, reads return 0 |

### 6.6.1.15 INTC\_MIR\_SET0 Register (offset = 8Ch) [reset = 0h]

INTC\_MIR\_SET0 is shown in [Figure 6-18](#) and described in [Table 6-19](#).

This register is used to set the interrupt mask bits.

**Figure 6-18. INTC\_MIR\_SET0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-19. INTC\_MIR\_SET0 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description                                    |
|------|--------|------|-------|--|
| 31-0 | MirSet | W    | 0h    | Write 1 sets the mask bit to 1, reads return 0 |

### 6.6.1.16 INTC\_ISR\_SET0 Register (offset = 90h) [reset = 0h]

INTC\_ISR\_SET0 is shown in [Figure 6-19](#) and described in [Table 6-20](#).

This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.

**Figure 6-19. INTC\_ISR\_SET0 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrSet |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-20. INTC\_ISR\_SET0 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | IsrSet | R/W  | 0h    | Reads returns the currently active software interrupts, Write 1 sets the software interrupt bits to 1 |

### 6.6.1.17 INTC\_ISR\_CLEAR0 Register (offset = 94h) [reset = 0h]

INTC\_ISR\_CLEAR0 is shown in [Figure 6-20](#) and described in [Table 6-21](#).

This register is used to clear the software interrupt bits

**Figure 6-20. INTC\_ISR\_CLEAR0 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrClear |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-21. INTC\_ISR\_CLEAR0 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-0 | IsrClear | W    | 0h    | Write 1 clears the software interrupt bits to 0, reads return 0 |

### 6.6.1.18 INTC\_PENDING\_IRQ0 Register (offset = 98h) [reset = 0h]

INTC\_PENDING\_IRQ0 is shown in [Figure 6-21](#) and described in [Table 6-22](#).

This register contains the IRQ status after masking

**Figure 6-21. INTC\_PENDING\_IRQ0 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingIRQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-22. INTC\_PENDING\_IRQ0 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingIRQ | R    | 0h    | IRQ status after masking |

### 6.6.1.19 INTC\_PENDING\_FIQ0 Register (offset = 9Ch) [reset = 0h]

INTC\_PENDING\_FIQ0 is shown in [Figure 6-22](#) and described in [Table 6-23](#).

This register contains the FIQ status after masking

**Figure 6-22. INTC\_PENDING\_FIQ0 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingFIQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-23. INTC\_PENDING\_FIQ0 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingFIQ | R    | 0h    | FIQ status after masking |

### 6.6.1.20 INTC\_ITR1 Register (offset = A0h) [reset = 0h]

INTC\_ITR1 is shown in [Figure 6-23](#) and described in [Table 6-24](#).

This register shows the raw interrupt input status before masking

**Figure 6-23. INTC\_ITR1 Register**

| 31   | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Itr  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-24. INTC\_ITR1 Register Field Descriptions**

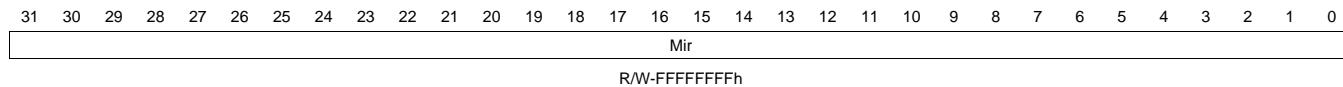
| Bit  | Field | Type | Reset | Description                     |
|------|-------|------|-------|---------------------------------|
| 31-0 | Itr   | R    | 0h    | Interrupt status before masking |

### 6.6.1.21 INTC\_MIR1 Register (offset = A4h) [reset = FFFFFFFFh]

INTC\_MIR1 is shown in [Figure 6-24](#) and described in [Table 6-25](#).

This register contains the interrupt mask

**Figure 6-24. INTC\_MIR1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-25. INTC\_MIR1 Register Field Descriptions**

| Bit  | Field | Type | Reset       | Description    |
|------|-------|------|-------------|----------------|
| 31-0 | Mir   | R/W  | FFFFFFFFFFh | Interrupt mask |

### 6.6.1.22 INTC\_MIR\_CLEAR1 Register (offset = A8h) [reset = 0h]

INTC\_MIR\_CLEAR1 is shown in [Figure 6-25](#) and described in [Table 6-26](#).

This register is used to clear the interrupt mask bits.

**Figure 6-25. INTC\_MIR\_CLEAR1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-26. INTC\_MIR\_CLEAR1 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                                      |
|------|----------|------|-------|--|
| 31-0 | MirClear | W    | 0h    | Write 1 clears the mask bit to 0, reads return 0 |

### 6.6.1.23 INTC\_MIR\_SET1 Register (offset = ACh) [reset = 0h]

INTC\_MIR\_SET1 is shown in [Figure 6-26](#) and described in [Table 6-27](#).

This register is used to set the interrupt mask bits.

**Figure 6-26. INTC\_MIR\_SET1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-27. INTC\_MIR\_SET1 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description                                    |
|------|--------|------|-------|--|
| 31-0 | MirSet | W    | 0h    | Write 1 sets the mask bit to 1, reads return 0 |

### 6.6.1.24 INTC\_ISR\_SET1 Register (offset = B0h) [reset = 0h]

INTC\_ISR\_SET1 is shown in [Figure 6-27](#) and described in [Table 6-28](#).

This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.

**Figure 6-27. INTC\_ISR\_SET1 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrSet |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-28. INTC\_ISR\_SET1 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | IsrSet | R/W  | 0h    | Reads returns the currently active software interrupts, Write 1 sets the software interrupt bits to 1 |

### 6.6.1.25 INTC\_ISR\_CLEAR1 Register (offset = B4h) [reset = 0h]

INTC\_ISR\_CLEAR1 is shown in [Figure 6-28](#) and described in [Table 6-29](#).

This register is used to clear the software interrupt bits

**Figure 6-28. INTC\_ISR\_CLEAR1 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| IsrClear |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-29. INTC\_ISR\_CLEAR1 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-0 | IsrClear | W    | 0h    | Write 1 clears the software interrupt bits to 0, reads return 0 |

### 6.6.1.26 INTC\_PENDING\_IRQ1 Register (offset = B8h) [reset = 0h]

INTC\_PENDING\_IRQ1 is shown in [Figure 6-29](#) and described in [Table 6-30](#).

This register contains the IRQ status after masking

**Figure 6-29. INTC\_PENDING\_IRQ1 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingIRQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-30. INTC\_PENDING\_IRQ1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingIRQ | R    | 0h    | IRQ status after masking |

### 6.6.1.27 INTC\_PENDING\_FIQ1 Register (offset = BCh) [reset = 0h]

INTC\_PENDING\_FIQ1 is shown in [Figure 6-30](#) and described in [Table 6-31](#).

This register contains the FIQ status after masking

**Figure 6-30. INTC\_PENDING\_FIQ1 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingFIQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-31. INTC\_PENDING\_FIQ1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingFIQ | R    | 0h    | FIQ status after masking |

### 6.6.1.28 INTC\_ITR2 Register (offset = C0h) [reset = 0h]

INTC\_ITR2 is shown in [Figure 6-31](#) and described in [Table 6-32](#).

This register shows the raw interrupt input status before masking

**Figure 6-31. INTC\_ITR2 Register**

| 31   | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Itr  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-32. INTC\_ITR2 Register Field Descriptions**

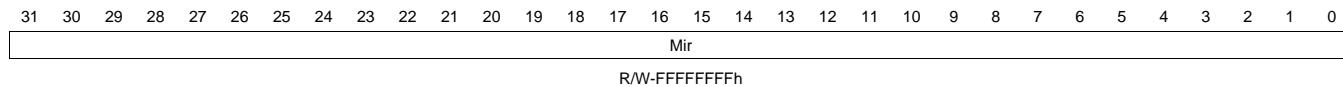
| Bit  | Field | Type | Reset | Description                     |
|------|-------|------|-------|---------------------------------|
| 31-0 | Itr   | R    | 0h    | Interrupt status before masking |

### 6.6.1.29 INTC\_MIR2 Register (offset = C4h) [reset = FFFFFFFFh]

INTC\_MIR2 is shown in [Figure 6-32](#) and described in [Table 6-33](#).

This register contains the interrupt mask

**Figure 6-32. INTC\_MIR2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-33. INTC\_MIR2 Register Field Descriptions**

| Bit  | Field | Type | Reset    | Description    |
|------|-------|------|----------|----------------|
| 31-0 | Mir   | R/W  | FFFFFFFh | Interrupt mask |

### 6.6.1.30 INTC\_MIR\_CLEAR2 Register (offset = C8h) [reset = 0h]

INTC\_MIR\_CLEAR2 is shown in [Figure 6-33](#) and described in [Table 6-34](#).

This register is used to clear the interrupt mask bits.

**Figure 6-33. INTC\_MIR\_CLEAR2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-34. INTC\_MIR\_CLEAR2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                                      |
|------|----------|------|-------|--|
| 31-0 | MirClear | W    | 0h    | Write 1 clears the mask bit to 0, reads return 0 |

### 6.6.1.31 INTC\_MIR\_SET2 Register (offset = CCh) [reset = 0h]

INTC\_MIR\_SET2 is shown in [Figure 6-34](#) and described in [Table 6-35](#).

This register is used to set the interrupt mask bits.

**Figure 6-34. INTC\_MIR\_SET2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-35. INTC\_MIR\_SET2 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description                                    |
|------|--------|------|-------|--|
| 31-0 | MirSet | W    | 0h    | Write 1 sets the mask bit to 1, reads return 0 |

### 6.6.1.32 INTC\_ISR\_SET2 Register (offset = D0h) [reset = 0h]

INTC\_ISR\_SET2 is shown in [Figure 6-35](#) and described in [Table 6-36](#).

This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.

**Figure 6-35. INTC\_ISR\_SET2 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrSet |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-36. INTC\_ISR\_SET2 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | IsrSet | R/W  | 0h    | Reads returns the currently active software interrupts, Write 1 sets the software interrupt bits to 1 |

### 6.6.1.33 INTC\_ISR\_CLEAR2 Register (offset = D4h) [reset = 0h]

INTC\_ISR\_CLEAR2 is shown in [Figure 6-36](#) and described in [Table 6-37](#).

This register is used to clear the software interrupt bits

**Figure 6-36. INTC\_ISR\_CLEAR2 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrClear |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-37. INTC\_ISR\_CLEAR2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-0 | IsrClear | W    | 0h    | Write 1 clears the software interrupt bits to 0, reads return 0 |

### 6.6.1.34 INTC\_PENDING\_IRQ2 Register (offset = D8h) [reset = 0h]

INTC\_PENDING\_IRQ2 is shown in [Figure 6-37](#) and described in [Table 6-38](#).

This register contains the IRQ status after masking

**Figure 6-37. INTC\_PENDING\_IRQ2 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingIRQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-38. INTC\_PENDING\_IRQ2 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingIRQ | R    | 0h    | IRQ status after masking |

### 6.6.1.35 INTC\_PENDING\_FIQ2 Register (offset = DCh) [reset = 0h]

INTC\_PENDING\_FIQ2 is shown in [Figure 6-38](#) and described in [Table 6-39](#).

This register contains the FIQ status after masking

**Figure 6-38. INTC\_PENDING\_FIQ2 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingFIQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-39. INTC\_PENDING\_FIQ2 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingFIQ | R    | 0h    | FIQ status after masking |

### 6.6.1.36 INTC\_ITR3 Register (offset = E0h) [reset = 0h]

INTC\_ITR3 is shown in [Figure 6-39](#) and described in [Table 6-40](#).

This register shows the raw interrupt input status before masking

**Figure 6-39. INTC\_ITR3 Register**

| 31   | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Itr  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-40. INTC\_ITR3 Register Field Descriptions**

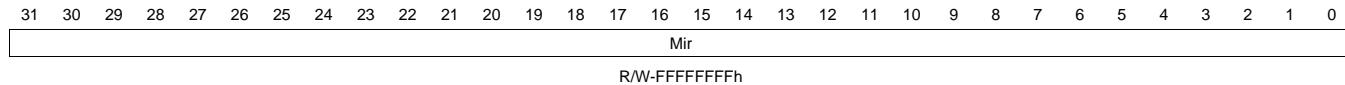
| Bit  | Field | Type | Reset | Description                     |
|------|-------|------|-------|---------------------------------|
| 31-0 | Itr   | R    | 0h    | Interrupt status before masking |

### 6.6.1.37 INTC\_MIR3 Register (offset = E4h) [reset = FFFFFFFFh]

INTC\_MIR3 is shown in [Figure 6-40](#) and described in [Table 6-41](#).

This register contains the interrupt mask

**Figure 6-40. INTC\_MIR3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-41. INTC\_MIR3 Register Field Descriptions**

| Bit  | Field | Type | Reset    | Description    |
|------|-------|------|----------|----------------|
| 31-0 | Mir   | R/W  | FFFFFFFh | Interrupt mask |

### 6.6.1.38 INTC\_MIR\_CLEAR3 Register (offset = E8h) [reset = 0h]

INTC\_MIR\_CLEAR3 is shown in [Figure 6-41](#) and described in [Table 6-42](#).

This register is used to clear the interrupt mask bits.

**Figure 6-41. INTC\_MIR\_CLEAR3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-42. INTC\_MIR\_CLEAR3 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                                      |
|------|----------|------|-------|--|
| 31-0 | MirClear | W    | 0h    | Write 1 clears the mask bit to 0, reads return 0 |

### 6.6.1.39 INTC\_MIR\_SET3 Register (offset = ECh) [reset = 0h]

INTC\_MIR\_SET3 is shown in [Figure 6-42](#) and described in [Table 6-43](#).

This register is used to set the interrupt mask bits.

**Figure 6-42. INTC\_MIR\_SET3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-43. INTC\_MIR\_SET3 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description                                    |
|------|--------|------|-------|--|
| 31-0 | MirSet | W    | 0h    | Write 1 sets the mask bit to 1, reads return 0 |

#### 6.6.1.40 INTC\_ISR\_SET3 Register (offset = F0h) [reset = 0h]

INTC\_ISR\_SET3 is shown in [Figure 6-43](#) and described in [Table 6-44](#).

This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.

**Figure 6-43. INTC\_ISR\_SET3 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrSet |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-44. INTC\_ISR\_SET3 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | IsrSet | R/W  | 0h    | Reads returns the currently active software interrupts, Write 1 sets the software interrupt bits to 1 |

#### 6.6.1.41 INTC\_ISR\_CLEAR3 Register (offset = F4h) [reset = 0h]

INTC\_ISR\_CLEAR3 is shown in [Figure 6-44](#) and described in [Table 6-45](#).

This register is used to clear the software interrupt bits

**Figure 6-44. INTC\_ISR\_CLEAR3 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IsrClear |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-45. INTC\_ISR\_CLEAR3 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-0 | IsrClear | W    | 0h    | Write 1 clears the software interrupt bits to 0, reads return 0 |

#### 6.6.1.42 INTC\_PENDING\_IRQ3 Register (offset = F8h) [reset = 0h]

INTC\_PENDING\_IRQ3 is shown in [Figure 6-45](#) and described in [Table 6-46](#).

This register contains the IRQ status after masking

**Figure 6-45. INTC\_PENDING\_IRQ3 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingIRQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-46. INTC\_PENDING\_IRQ3 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingIRQ | R    | 0h    | IRQ status after masking |

#### 6.6.1.43 INTC\_PENDING\_FIQ3 Register (offset = FCh) [reset = 0h]

INTC\_PENDING\_FIQ3 is shown in [Figure 6-46](#) and described in [Table 6-47](#).

This register contains the FIQ status after masking

**Figure 6-46. INTC\_PENDING\_FIQ3 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| PendingFIQ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-47. INTC\_PENDING\_FIQ3 Register Field Descriptions**

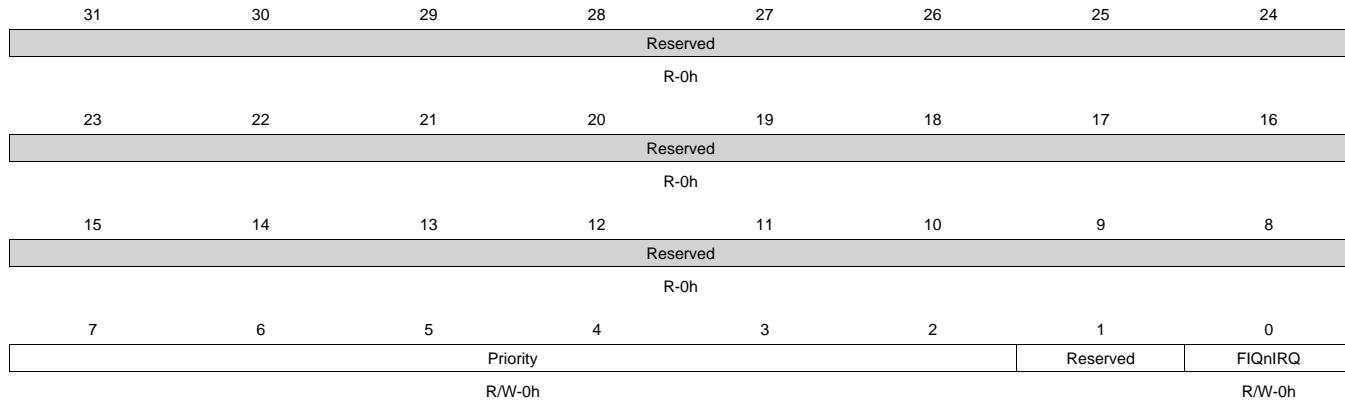
| Bit  | Field      | Type | Reset | Description              |
|------|------------|------|-------|--------------------------|
| 31-0 | PendingFIQ | R    | 0h    | FIQ status after masking |

#### 6.6.1.44 INTC\_ILR Register (offset = 100h to 2FCh) [reset = 0h]

INTC\_ILR0 to INTC\_ILR127 is shown in [Figure 6-47](#) and described in [Table 6-48](#).

The INTC\_ILRx registers contain the priority for the interrupts and the FIQ/IRQ steering.

**Figure 6-47. INTC\_ILR0 to INTC\_ILR127 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 6-48. INTC\_ILR0 to INTC\_ILR127 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    | Write 0's for future compatibility.<br>Reads returns 0   |
| 7-2  | Priority | R/W  | 0h    | Interrupt priority   |
| 0    | FIQnIRQ  | R/W  | 0h    | Interrupt IRQ/FIQ mapping<br>0x0 = IntIRQ : Interrupt is routed to IRQ.<br>0x1 = IntFIQ : Interrupt is routed to FIQ (this selection is reserved on GP devices). |

## **Memory Subsystem**

This chapter describes the memory subsystem of the device.

| Topic              | Page |
|--------------------|------|
| 7.1 GPMC .....     | 250  |
| 7.2 OCMC-RAM ..... | 397  |
| 7.3 EMIF .....     | 399  |
| 7.4 ELM .....      | 475  |

## 7.1 GPMC

### 7.1.1 Introduction

The general-purpose memory controller (GPMC) is an unified memory controller dedicated to interfacing external memory devices:

- Asynchronous SRAM-like memories and application-specific integrated circuit (ASIC) devices
- Asynchronous, synchronous, and page mode (only available in non-multiplexed mode) burst NOR flash devices
- NAND Flash
- Pseudo-SRAM devices

#### 7.1.1.1 GPMC Features

The general features of the GPMC module include:

- Data path to external memory device can be 16- or 8-bit wide
- 32-bit OCPIP 2.0 compliant core, single slave interface. Support non-wrapping and wrapping burst up to 16x32bits.
- Up to 100 MHz external memory clock performance (single device)
- Support for the following memory types:
  - External asynchronous or synchronous 8-bit width memory or device (non burst device)
  - External asynchronous or synchronous 16-bit width memory or device
  - External 16-bit non-multiplexed NOR Flash device
  - External 16-bit address and data multiplexed NOR Flash device
  - External 8-bit and 16-bit NAND flash device
  - External 16-bit pSRAM device
- Up to 16-bit ECC support for NAND flash using BCH code ( $t=4, 8$  or  $16$ ) or Hamming code for 8-bit or 16-bit NAND-flash, organized with page size of 512 bytes, 1K bytes, or more.
- Support 512M Bytes maximum addressing capability which can be divided into seven independent chip-select with programmable bank size and base address on 16M Bytes, 32M Bytes, 64M Bytes, or 128M Bytes boundary
- Fully pipelined operation for optimal memory bandwidth usage
- Support external device clock frequency of 1, 2, 3 and 4 divider from L3 clock.
- Support programmable auto-clock gating when there is no access.
- Support Midlereq/SidleAck protocol
- Support the following interface protocols when communicating with external memory or external devices.
  - Asynchronous read/write access
  - Asynchronous read page access (4-8-16 Word16)
  - Synchronous read/write access
  - Synchronous read burst access without wrap capability (4-8-16 Word16)
  - Synchronous read burst access with wrap capability (4-8-16 Word16)
- Address and Data multiplexed access
- Each chip-select as independent and programmable control signal timing parameters for Setup and Hold time. Parameters are set according to the memory device timing parameters, with one L3 clock cycle timing granularity.
- Flexible internal access time control (wait state) and flexible handshake mode using external WAIT pins monitoring (up to two WAIT pins)
- Support bus keeping
- Support bus turn around

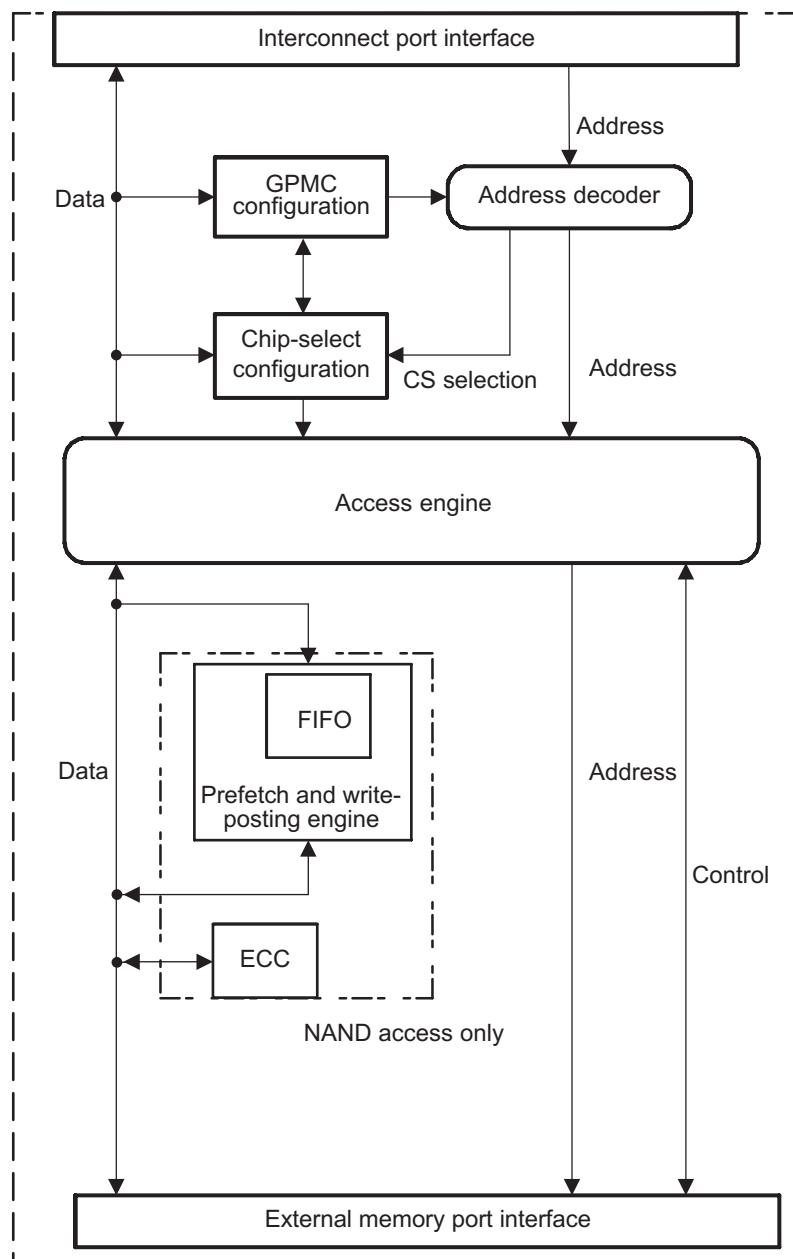
- Pre-fetch and write posting engine associated with system DMA to get full performance from NAND device with minimum impact on NOR/SRAM concurrent access.
- On the fly ECC Hamming Code calculation to improve NAND usage reliability with minimum impact on SW

### 7.1.1.2 Block Diagram

The GPMC can access various external devices through the L3 Slow Interconnect. The flexible programming model allows a wide range of attached device types and access schemes. Based on the programmed configuration bit fields stored in the GPMC registers, the GPMC is able to generate all control signals timing depending on the attached device and access type. Given the chip-select decoding and its associated configuration registers, the GPMC selects the appropriate device type control signals timing.

Figure 7-1 shows the GPMC functional block diagram. The GPMC consists of six blocks:

- Interconnect port interface
- Address decoder, GPMC configuration, and chip-select configuration register file
- Access engine
- Prefetch and write-posting engine
- Error correction code engine (ECC)
- External device/memory port interface

**Figure 7-1. GPMC Block Diagram**


### 7.1.1.3 Unsupported GPMC Features

The following module features are not supported in this device.

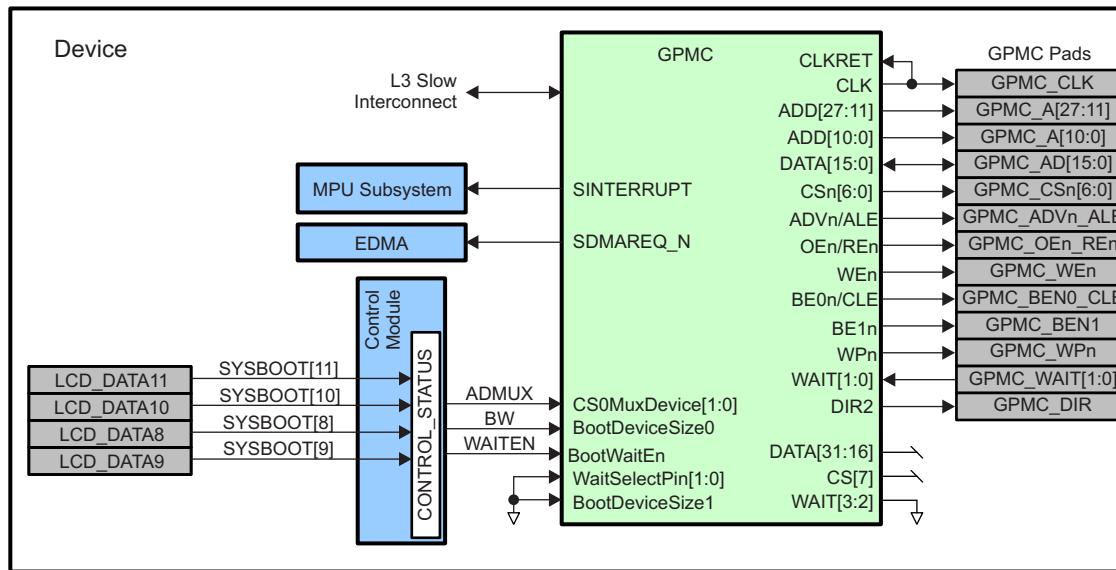
**Table 7-1. Unsupported GPMC Features**

| Feature        | Reason   |
|----------------|--|
| Chip Select 7  | Not pinned out   |
| 32-bit devices | Only 16 data lines pinned out                          |
| WAIT[3:2]      | Not pinned out. All CS regions must use WAIT0 or WAIT1 |

## 7.1.2 Integration

An instantiation of GPMC provides this device with access to NAND Flash, NOR Flash, and other asynchronous and synchronous interface peripherals. [Figure 7-2](#) shows the integration of the GPMC module in this device.

**Figure 7-2. GPMC Integration**



### 7.1.2.1 GPMC Connectivity Attributes

The general connectivity attributes for the GPMC module are shown in [Table 7-2](#).

**Table 7-2. GPMC Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L3S_GCLK  |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 1 interrupt to MPU Subsystem (GPMCIINT)  |
| DMA Requests        | 1 DMA request to EDMA (GPMCEVT)  |
| Physical Address    | L3 Slow Slave Port<br>Memory and control register regions qualified with MAddressSpace bit |

### 7.1.2.2 GPMC Clock and Reset Management

The GPMC is a synchronous design and operates from the same clock as the Slow L3. All timings use this clock as a reference.

**Table 7-3. GPMC Clock Signals**

| Clock Signal                                  | Max Freq | Reference / Source | Comments                     |
|---|----------|--------------------|------------------------------|
| prcm_gpmc_clk<br>Interface / Functional clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l3s_gclk<br>From PRCM |

### 7.1.2.3 GPMC Signal List

The GPMC external interface signals are shown in [Table 7-4](#).

**Table 7-4. GPMC Signal List**

| Signal         | Type | Description   |
|----------------|------|---|
| GPMC_A[27:0]   | O    | Address outputs   |
| GPMC_AD[15:0]  | I/O  | Data - multiplexed with Address[16:1] and Address[27:12]  |
| GPMC_CSn[6:0]  | O    | Chip selects (active low)   |
| GPMC_CLK       | O    | Synchronous mode clock  |
| GPMC_ADVn_ALE  | O    | Address Valid or Address Latch Enable depending if NOR or NAND protocol memories are selected.  |
| GPMC_OEn_REn   | O    | Output Enable (active low). Also used as Read Enable (active low) for NAND protocol memories  |
| GPMC_WEn       | O    | Write Enable (active low)   |
| GPMC_BE0n_CLE  | O    | Lower Byte Enable (active low). Also used as Command Latch Enable for NAND protocol memories  |
| GPMC_BE1n      | O    | Upper Byte Enable (active low)  |
| GPMC_WPn       | O    | Write Protect (active low)  |
| GPMC_WAIT[1:0] | I    | External wait signal for NOR and NAND protocol memories.  |
| GPMC_DIR       | O    | GPMC.D[15:0] signal direction control<br>Low during transmit (for write access: data OUT from GPMC to memory)<br>High during receive (for read access: data IN from memory to GPMC) |

### 7.1.3 Functional Description

#### 7.1.3.1 GPMC Signals

Table 7-5 shows the use of address and data GPMC controller pins based on the type of external device.

**Table 7-5. GPMC Pin Multiplexing Options**

| GPMC Signal | Non Multiplexed Address Data 16-Bit Device | Non Multiplexed Address Data 8-Bit Device | Multiplexed Address Data 16-Bit Device | 16-Bit NAND Device | 8-Bit NAND Device |
|-------------|--|---|--|--------------------|-------------------|
| GPMC_A[27]  | A26  | A27                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[26]  | A25  | A26                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[25]  | A24  | A25                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[24]  | A23  | A24                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[23]  | A22  | A23                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[22]  | A21  | A22                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[21]  | A20  | A21                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[20]  | A19  | A20                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[19]  | A18  | A19                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[18]  | A17  | A18                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[17]  | A16  | A17                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[16]  | A15  | A16                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[15]  | A14  | A15                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[14]  | A13  | A14                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[13]  | A12  | A13                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[12]  | A11  | A12                                       | Not Used                               | Not Used           | Not Used          |
| GPMC_A[11]  | A10  | A11                                       | A26                                    | Not Used           | Not Used          |
| GPMC_A[10]  | A9   | A10                                       | A25                                    | Not Used           | Not Used          |
| GPMC_A[9]   | A8   | A9  | A24                                    | Not Used           | Not Used          |
| GPMC_A[8]   | A7   | A8  | A23                                    | Not Used           | Not Used          |
| GPMC_A[7]   | A6   | A7  | A22                                    | Not Used           | Not Used          |
| GPMC_A[6]   | A5   | A6  | A21                                    | Not Used           | Not Used          |
| GPMC_A[5]   | A4   | A5  | A20                                    | Not Used           | Not Used          |
| GPMC_A[4]   | A3   | A4  | A19                                    | Not Used           | Not Used          |
| GPMC_A[3]   | A2   | A3  | A18                                    | Not Used           | Not Used          |
| GPMC_A[2]   | A1   | A2  | A17                                    | Not Used           | Not Used          |
| GPMC_A[1]   | A0   | A1  | A16                                    | Not Used           | Not Used          |
| GPMC_A[0]   | Not Used                                   | A0  | Not Used                               | Not Used           | Not Used          |
| GPMC_AD[15] | D15  | Not Used                                  | A/D[15]                                | D15                | Not Used          |
| GPMC_AD[14] | D14  | Not Used                                  | A/D[14]                                | D14                | Not Used          |
| GPMC_AD[13] | D13  | Not Used                                  | A/D[13]                                | D13                | Not Used          |
| GPMC_AD[12] | D12  | Not Used                                  | A/D[12]                                | D12                | Not Used          |
| GPMC_AD[11] | D11  | Not Used                                  | A/D[11]                                | D11                | Not Used          |
| GPMC_AD[10] | D10  | Not Used                                  | A/D[10]                                | D10                | Not Used          |
| GPMC_AD[9]  | D9   | Not Used                                  | A/D[9]                                 | D9                 | Not Used          |
| GPMC_AD[8]  | D8   | Not Used                                  | A/D[8]                                 | D8                 | Not Used          |
| GPMC_AD[7]  | D7   | D7  | A/D[7]                                 | D7                 | D7                |
| GPMC_AD[6]  | D6   | D6  | A/D[6]                                 | D6                 | D6                |
| GPMC_AD[5]  | D5   | D5  | A/D[5]                                 | D5                 | D5                |
| GPMC_AD[4]  | D4   | D4  | A/D[4]                                 | D4                 | D4                |

Table 7-5. GPMC Pin Multiplexing Options (continued)

| GPMC Signal   | Non Multiplexed Address Data 16-Bit Device | Non Multiplexed Address Data 8-Bit Device | Multiplexed Address Data 16-Bit Device | 16-Bit NAND Device         | 8-Bit NAND Device          |
|---------------|--|---|--|----------------------------|----------------------------|
| GPMC_AD[3]    | D3   | D3  | A/D[3]                                 | D3                         | D3                         |
| GPMC_AD[2]    | D2   | D2  | A/D[2]                                 | D2                         | D2                         |
| GPMC_AD[1]    | D1   | D1  | A/D[1]                                 | D1                         | D1                         |
| GPMC_AD[0]    | D0   | D0  | A/D[0]                                 | D0                         | D0                         |
| GPMC_CS[0]n   | CS0n (Chip Select)                         | CS0n (Chip Select)                        | CS0n (Chip Select)                     | CE0n (Chip Enable)         | CE0n (Chip Enable)         |
| GPMC_CS[1]n   | CS1n                                       | CS1n                                      | CS1n                                   | CE1n                       | CE1n                       |
| GPMC_CS[2]n   | CS2n                                       | CS2n                                      | CS2n                                   | CE2n                       | CE2n                       |
| GPMC_CS[3]n   | CS3n                                       | CS3n                                      | CS3n                                   | CE3n                       | CE3n                       |
| GPMC_CS[4]n   | CS4n                                       | CS4n                                      | CS4n                                   | CE4n                       | CE4n                       |
| GPMC_CS[5]n   | CS5n                                       | CS5n                                      | CS5n                                   | CE5n                       | CE5n                       |
| GPMC_CS[6]n   | CS6n                                       | CS6n                                      | CS6n                                   | CE6n                       | CE6n                       |
| GPMC_ADVn_ALE | ADVn (Address Value)                       | ADVn (Address Value)                      | ADVn (Address Value)                   | ALE (address latch enable) | ALE (address latch enable) |
| GPMC_BE0n_CLE | BE0n (Byte Enable)                         | BE0n (Byte Enable)                        | BE0n (Byte Enable)                     | CLE (command latch enable) | CLE (command latch enable) |
| GPMC_BE1n     | BE1n                                       | BE1n                                      | BE1n                                   |                            |                            |
| GPMC_CLK      | CLK  | CLK                                       | CLK                                    |                            |                            |
| GPMC_OE_REn   | OEn (Output Enable)                        | OEn (Output Enable)                       | OEn (Output Enable)                    | REn (read enable)          | REn (read enable)          |
| GPMC_WAIT0    | WAIT0                                      | WAIT0                                     | WAIT0                                  | R/B0n (ready/busy)         | R/B0n (ready/busy)         |
| GPMC_WAIT1    | WAIT1                                      | WAIT1                                     | WAIT1                                  | R/B1n (ready/busy)         | R/B1n (ready/busy)         |
| GPMC_WEn      | WEn (Write Enable)                         | WEn (Write Enable)                        | WEn (Write Enable)                     | WEn (write enable)         | WEn (write enable)         |
| GPMC_WPn      | WPn (Write Protect)                        | WPn (Write Protect)                       | WPn (Write Protect)                    | WPn (write protect)        | WPn (write protect)        |

With all device types, the GPMC does not drive unnecessary address lines. They stay at their reset value of 00.

Address mapping supports address/data-multiplexed 16-bit wide devices:

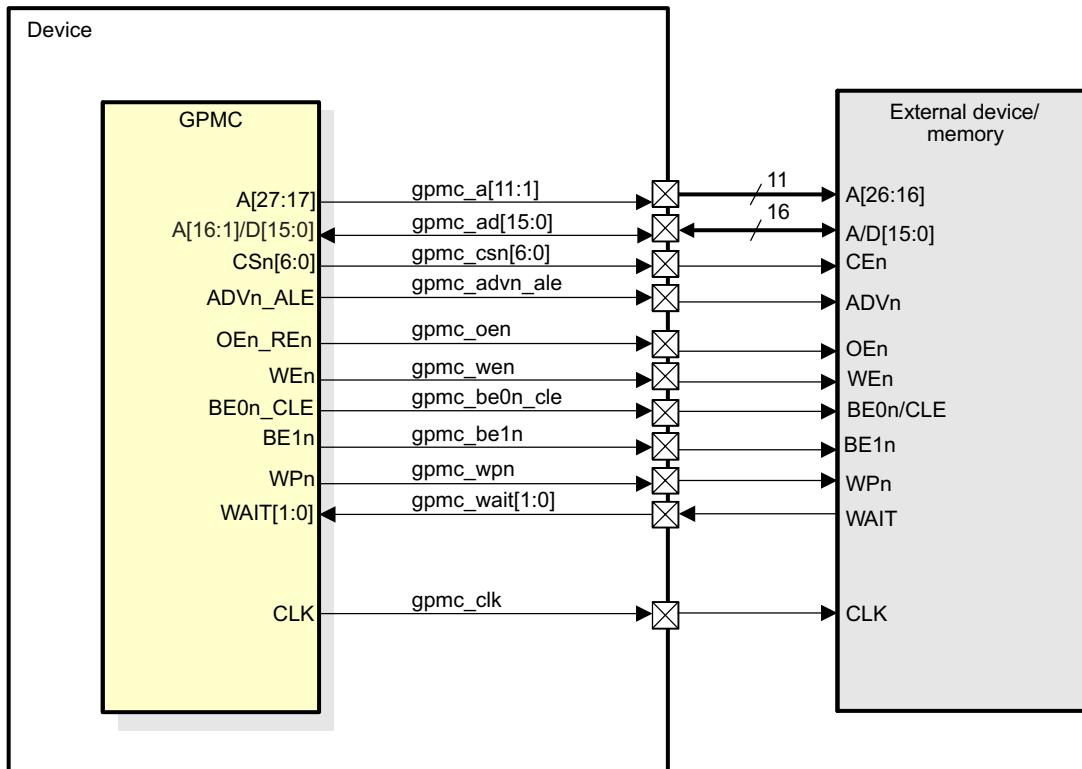
- The NOR flash memory controller still supports non-multiplexed address and data memory devices.
- Multiplexing mode can be selected through the GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit field.
- Asynchronous page mode is not supported for multiplexed address and data devices.

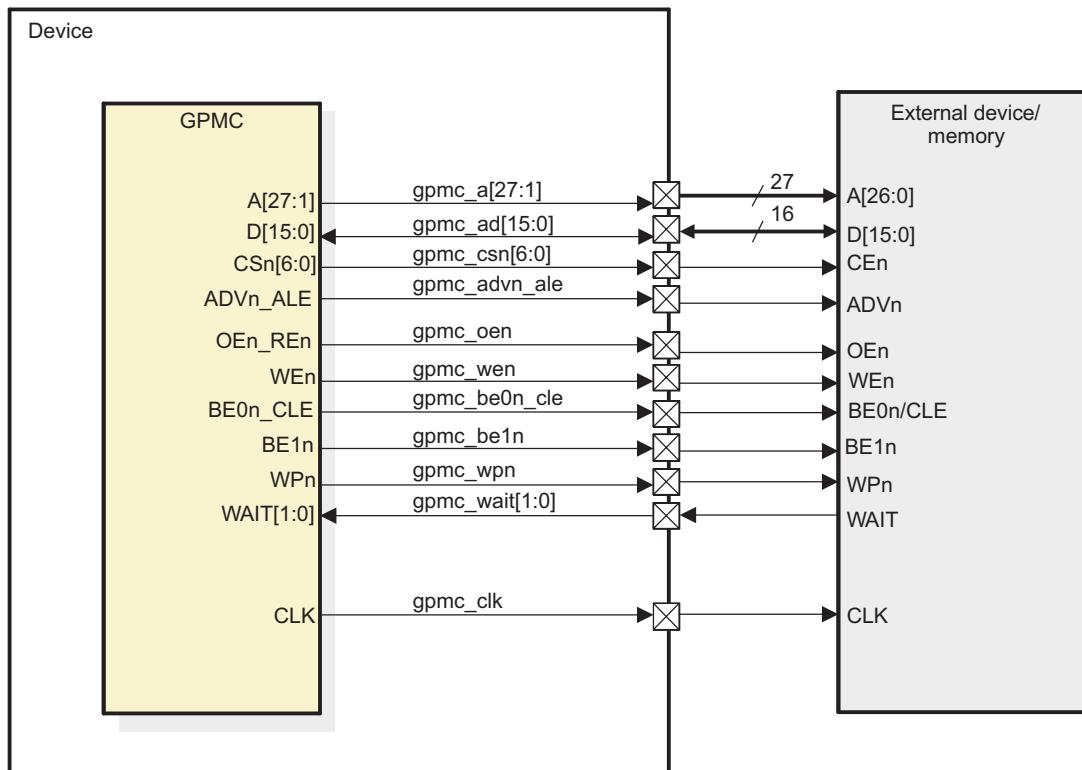
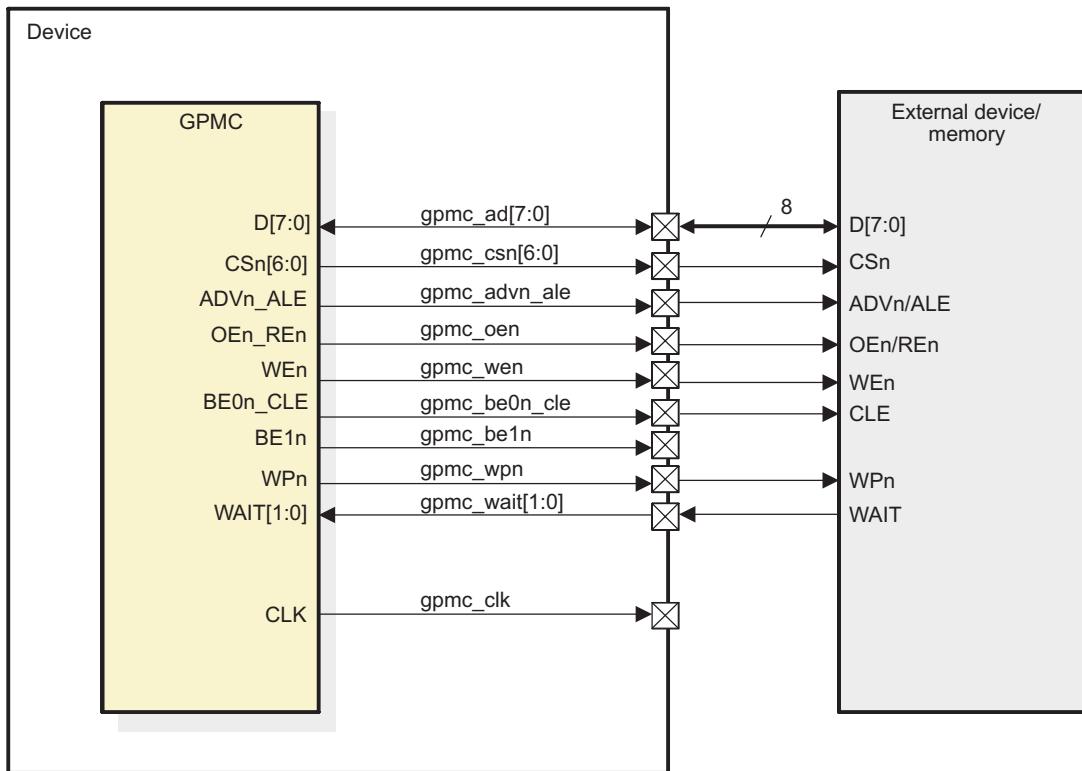
### 7.1.3.2 GPMC Modes

This section shows three GPMC external connections options:

- [Figure 7-3](#) shows a connection between the GPMC and a 16-bit synchronous address/data-multiplexed (or AAD-multiplexed, but this protocol use less address pins) external memory device.
- [Figure 7-4](#) shows a connection between the GPMC and a 16-bit synchronous nonmultiplexed external memory device .
- [Figure 7-5](#) shows a connection between the GPMC and a 8-bit NAND device

**Figure 7-3. GPMC to 16-Bit Address/Data-Multiplexed Memory**



**Figure 7-4. GPMC to 16-Bit Non-multiplexed Memory**

**Figure 7-5. GPMC to 8-Bit NAND Device**


### 7.1.3.3 GPMC Functional Description

The GPMC basic programming model offers maximum flexibility to support various access protocols for each of the eight configurable chip-selects. Use optimal chip-select settings, based on the characteristics of the external device:

- Different protocols can be selected to support generic asynchronous or synchronous random-access devices (NOR flash, SRAM) or to support specific NAND devices.
- The address and the data bus can be multiplexed on the same external bus.
- Read and write access can be independently defined as asynchronous or synchronous.
- System requests (byte, 16-bit word, burst) are performed through single or multiple accesses. External access profiles (single, multiple with optimized burst length, native- or emulated-wrap) are based on external device characteristics (supported protocol, bus width, data buffer size, native-wrap support).
- System burst read or write requests are synchronous-burst (multiple-read or multiple-write). When neither burst nor page mode is supported by external memory or ASIC devices, system burst read or write requests are translated to successive single synchronous or asynchronous accesses (single reads or single writes). 8-bit wide devices are supported only in single synchronous or single asynchronous read or write mode.
- To simulate a programmable internal-wait state, an external wait pin can be monitored to dynamically control external access at the beginning (initial access time) of and during a burst access.

Each control signal is controlled independently for each chip-select. The internal functional clock of the GPMC (GPMC\_FCLK) is used as a time reference to specify the following:

- Read- and write-access duration
- Most GPMC external interface control-signal assertion and deassertion times
- Data-capture time during read access
- External wait-pin monitoring time
- Duration of idle time between accesses, when required

### 7.1.3.3.1 GPMC Clock Configuration

Table 7-6 describes the GPMC clocks.

**Table 7-6. GPMC Clocks**

| Signal    | I/O | Description   |
|-----------|-----|---|
| GPMC_FCLK | I   | Functional and interface clock                                  |
| GPMC_CLK  | O   | External clock provided to synchronous external memory devices. |

The GPMC\_CLK is generated by the GPMC from the internal GPMC\_FCLK clock. The source of the GPMC\_FCLK is described in [Table 7-3](#). The GPMC\_CLK is configured via the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER field (for  $i = 0$  to 3) as shown in [Table 7-7](#).

**Table 7-7. GPMC\_CONFIG1\_i Configuration**

| Source Clock | GPMC_CONFIG1_i[1-0]<br>GPMCFCLKDIVIDER | GPMC_CLK Generated Clock<br>Provided to External Memory Device |
|--------------|--|--|
| GPMC_FCLK    | 00                                     | GPMC_FCLK  |
|              | 01                                     | GPMC_FCLK/2  |
|              | 10                                     | GPMC_FCLK/3  |
|              | 11                                     | GPMC_FCLK/4  |

### 7.1.3.3.2 GPMC Software Reset

The GPMC can be reset by software through the GPMC\_SYSCONFIG[1] SOFTRESET bit. Setting the bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Hardware and software resets initialize all GPMC registers and the finite state-machine (FSM) immediately and unconditionally. The GPMC\_SYSSTATUS[0] RESETDONE bit indicates that the software reset is complete when its value is 1. The software must ensure that the software reset completes before doing GPMC operations.

### 7.1.3.3.3 GPMC Power Management

GPMC power is supplied by the CORE power domain, and GPMC power management complies with system power-management guidelines. [Table 7-8](#) describes power-management features available for the GPMC module.

**Table 7-8. GPMC Local Power Management Features**

| Feature                | Registers                                  | Description   |
|------------------------|--|---|
| Clock Auto Gating      | GPMC_SYSCONFIG[0]<br>AUTOIDLE bit          | This bit allows a local power optimization inside the module, by gating the GPMC_FCLK clock upon the internal activity. |
| Slave Idle Modes       | GPMC_SYSCONFIG[4-3]<br>SIDLEMODE bit field | Force-idle, No-idle and Smart-idle wakeup modes are available   |
| Clock Activity         | N/A  | Feature not available   |
| Master Standby Modes   | N/A  | Feature not available   |
| Global Wake-up Enable  | N/A  | Feature not available   |
| Wake-up Sources Enable | N/A  | Feature not available   |

#### 7.1.3.3.4 GPMC Interrupt Requests

The GPMC generates one interrupt event as shown in [Figure 7-2](#).

- The interrupt request goes from GPMC (GPMC\_IRQ) to the Cortex-A8 MPU subsystem: A\_IRQ\_100

[Table 7-9](#) lists the event flags, and their mask, that can cause module interrupts.

**Table 7-9. GPMC Interrupt Events**

| Event Flag  | Event Mask  | Sensitivity | Map to    | Description  |
|---|---|-------------|-----------|--|
| GPMC IRQSTATUS[9]<br>WAIT1EDGEDETECTIO<br>NSTATUS | GPMC IRQENABLE[9]<br>WAIT1EDGEDETECTIO<br>NENABLE | Edge        | A_IRQ_100 | Wait1 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT1 signal. The rising or falling edge detection of Wait1 is selected through GPMC_CONFIG[9] WAIT1PINPOLARITY bit. |
| GPMC IRQSTATUS[8]<br>WAIT0EDGEDETECTIO<br>NSTATUS | GPMC IRQENABLE[8]<br>WAIT0EDGEDETECTIO<br>NENABLE | Edge        | A_IRQ_100 | Wait0 edge detection interrupt: Triggered if a rising or falling edge is detected on the GPMC_WAIT0 signal. The rising or falling edge detection of Wait0 is selected through GPMC_CONFIG[8] WAIT0PINPOLARITY bit. |
| GPMC IRQSTATUS[1]<br>TERMINALCOUNTSTAT<br>US      | GPMC IRQENABLE[1]<br>TERMINALCOUNTENA<br>BLE      | Level       | A_IRQ_100 | Terminal count event: Triggered on prefetch process completion, that is when the number of currently remaining data to be requested reaches 0.   |
| GPMC IRQSTATUS[0]<br>FIFOEVENTSTATUS              | GPMC IRQENABLE[0]<br>FIFOEVENTENABLE              | Level       | A_IRQ_100 | FIFO event interrupt: Indicates FIFO levels availability for in Write-Posting mode and prefetch mode. GPMC_PREFETCH_CONFIG[2] DMAMODE bit shall be cleared to 0.   |

#### 7.1.3.3.5 GPMC DMA Requests

The GPMC generates one DMA event, from GPMC (GPMC\_DMA\_REQ) to the eDMA: e\_DMA\_53

#### 7.1.3.3.6 L3 Slow Interconnect Interface

The GPMC L3 Slow interconnect interface is a pipelined interface including an  $16 \times 32$ -bit word write buffer. Any system host can issue external access requests through the GPMC. The device system can issue the following requests through this interface:

- One 8-bit / 16-bit / 32-bit interconnect access (read/write)
- Two incrementing 32-bit interconnect accesses (read/write)
- Two wrapped 32-bit interconnect accesses (read/write)
- Four incrementing 32-bit interconnect accesses (read/write)
- Four wrapped 32-bit interconnect accesses (read/write)
- Eight incrementing 32-bit interconnect accesses (read/write)
- Eight wrapped 32-bit interconnect accesses (read/write)

Only linear burst transactions are supported; interleaved burst transactions are not supported. Only power-of-two-length precise bursts  $2 \times 32$ ,  $4 \times 32$ ,  $8 \times 32$  or  $16 \times 32$  with the burst base address aligned on the total burst size are supported (this limitation applies to incrementing bursts only).

This interface also provides one interrupt and one DMA request line, for specific event control.

It is recommended to program the GPMC\_CONFIG1\_i ATTACHEDDEVICEPAGELENGTH field ([24-23]) according to the effective attached device page length and to enable the GPMC\_CONFIG1\_i WRAPBURST bit ([31]) if the attached device supports wrapping burst. However, it is possible to emulate wrapping burst on a non-wrapping memory by providing relevant addresses within the page or splitting transactions. Bursts larger than the memory page length are chopped into multiple bursts transactions. Due to the alignment requirements, a page boundary is never crossed.

### 7.1.3.3.7 GPMC Address and Data Bus

The current application supports GPMC connection to NAND devices and to address/data-multiplexed memories or devices. Connection to address/data-nonmultiplexed memories Depending on the GPMC configuration of each chip-select, address and data bus lines that are not required for a particular access protocol are not updated (changed from current value) and are not sampled when input (input data bus).

- For address/data-multiplexed and AAD-multiplexed NOR devices, the address is multiplexed on the data bus.
- 8-bit wide NOR devices do not use GPMC I/O: GPMC\_AD[15-8] for data (they are used for address if needed).
- 16-bit wide NAND devices do not use GPMC I/O: GPMC\_A[27-0].
- 8-bit wide NAND devices do not use GPMC I/O: GPMC\_A[27-0] and GPMC I/O: GPMC\_AD[15-8].

#### 7.1.3.3.7.1 GPMC I/O Configuration Setting

---

**NOTE:** In this section and next sections, the i in GPMC\_CONFIGx\_i stands for the GPMC chip-select i where i = 0 to 6.

---

To select a NAND device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE field = 10b
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit = 00

To select an address/data-multiplexed device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE field = 00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit = 10b

To select an address/address/data-multiplexed device, program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE field = 00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit = 01b

To select an address/data-nonmultiplexed device , program the following register fields:

- GPMC\_CONFIG1\_i[11-10] DEVICETYPE field = 00
- GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit = 00

#### 7.1.3.3.8 Address Decoder and Chip-Select Configuration

Addresses are decoded accordingly with the address request of the chip-select and the content of the chip-select base address register file, which includes a set of global GPMC configuration registers and eight sets of chip-select configuration registers.

The GPMC configuration register file is memory-mapped and can be read or written with byte, 16-bit word, or 32-bit word accesses. The register file should be configured as a noncacheable, nonbufferable region to prevent any desynchronization between host execution (write request) and the completion of register configuration (write completed with register updated). [Section 7.1.5](#) provides the GPMC register locations. For the map of GPMC memory locations, see [Table 7-54](#).

After the chip-select is configured, the access engine accesses the external device, drives the external interface control signals, and applies the interface protocol based on user-defined timing parameters and settings.

### 7.1.3.3.8.1 Chip-Select Base Address and Region Size

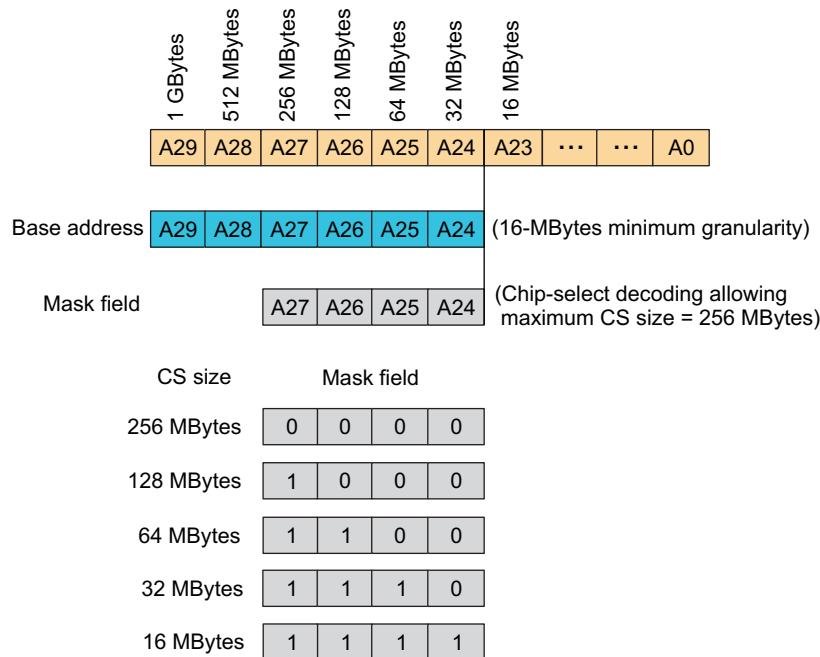
Any external memory or ASIC device attached to the GPMC external interface can be accessed by any device system host within the GPMC 512-Mbyte contiguous address space. For details, see [Table 7-54](#).

The GPMC 512 Mbyte address space can be divided into a maximum of seven chip-select regions with programmable base address and programmable CS size. The CS size is programmable from 16 Mbytes to 256 Mbytes (must be a power-of-2) and is defined by the mask field. Attached memory smaller than the programmed CS region size is accessed through the entire CS region (aliasing).

Each chip-select has a 6-bit base address encoding and a 4-bit decoding mask, which must be programmed according to the following rules:

- The programmed chip-select region base address must be aligned on the chip-select region size address boundary and is limited to a power-of-2 address value. During access decoding, the register base address value is used for address comparison with the address-bit line mapping as described in [Figure 7-6](#) (with A0 as the device system byte-address line). Base address is programmed through the GPMC\_CONFIG7\_i[5:0] BASEADDRESS bit field.
- The register mask is used to exclude some address lines from the decoding. A register mask bit field cleared to 0 suppresses the associated address line from the address comparison (incoming address bit line is don't care). The register mask value must be limited to the subsequent value, based on the desired chip-select region size. Any other value has an undefined result. When multiple chip-select regions with overlapping addresses are enabled concurrently, access to these chip-select regions is cancelled and a GPMC access error is posted. The mask field is programmed through the GPMC\_CONFIG7\_i[11:8] MASKADDRESS bit field.

**Figure 7-6. Chip-Select Address Mapping and Decoding Mask**



A mask value of 0010 or 1001 must be avoided because it will create holes in the chip-select address space.

Chip-select configuration (base and mask address or any protocol and timing settings) must be performed while the associated chip-select is disabled through the GPMC\_CONFIG7\_i[6] CSVALID bit. In addition, a chip-select configuration can only be disabled if there is no ongoing access to that chip-select. This requires activity monitoring of the prefetch or write-posting engine if the engine is active on the chip-select. Also, the write buffer state must be monitored to wait for any posted write completion to the chip-select.

Any access attempted to a nonvalid GPMC address region (CSVALID disabled or address decoding outside a valid chip-select region) is not propagated to the external interface and a GPMC access error is posted. In case of chip-selects overlapping, an error is generated and no access will occur on either chip-select. Chip-select 0 is the only chip-select region enabled after either a power-up or a GPMC reset.

Although the GPMC interface can drive up to seven chip-selects, the frequency specified for this interface is for a specific load. If this load is exceeded, the maximum frequency cannot be reached. One solution is to implement a board with buffers, to allow the slowest device to maintain the total load on the lines.

#### **7.1.3.3.8.2 Access Protocol**

##### **7.1.3.3.8.2.1 Supported Devices**

The access protocol of each chip-select can be independently specified through the GPMC\_CONFIG1\_i[11-10] DEVICETYPE parameter for:

- Random-access synchronous or asynchronous memory like NOR flash, SRAM
- NAND flash asynchronous devices

For more information about the NAND flash GPMC basic programming model and NAND support, see [Section 7.1.3.3.12](#) and [Section 7.1.3.3.12.1](#).

##### **7.1.3.3.8.2.2 Access Size Adaptation and Device Width**

Each chip-select can be independently configured through the GPMC\_CONFIG1\_i[13-12] DEVICESIZE field to interface with a 16-bit wide device or an 8-bit wide device. System requests with data width greater than the external device data bus width are split into successive accesses according to both the external device data-bus width and little-endian data organization.

An 8-bit wide device must be interfaced to the D0-D7 external interface bus lane. GPMC data accesses only use this bus lane when the associated chip-select is attached to an 8-bit wide device.

The 8-bit wide device can be interfaced in asynchronous or synchronous mode in single data phase (no 8-bit wide device burst mode). If the 8-bit wide device is set in the chip-select configuration register, ReadMultiple and WriteMultiple bit fields are considered “don’t care” and only single accesses are performed.

A 16-bit wide device can be interfaced in asynchronous or synchronous mode, with single or multiple data phases for an access, and with native or emulated wrap mode support.

##### **7.1.3.3.8.2.3 Address/Data-Multiplexing Interface**

For random synchronous or asynchronous memory interfacing (DEVICETYPE = 0b00), an address- and data-multiplexing protocol can be selected through the GPMC\_CONFIG1\_i[9-8] MUXADDDATA bit field. The ADVn signal must be used as the external device address latch control signal. For the associated chip-select configuration, ADVn assertion and deassertion time and OEn assertion time must be set to the appropriate value to meet the address latch setup/hold time requirements of the external device (see [Section 7.1.2](#)).

This address/data-multiplexing interface is not applicable to NAND device interfacing. NAND devices require a specific address, command, and data multiplexing protocol (see [Section 7.1.3.3.12](#)).

#### **7.1.3.3.8.3 External Signals**

##### **7.1.3.3.8.3.1 WAIT Pin Monitoring Control**

GPMC access time can be dynamically controlled using an external gpmc\_wait pin when the external device access time is not deterministic and cannot be defined and controlled only using the GPMC internal RDACCESSTIME, WRACCESSTIME and PAGEBURSTACCESSTIME wait state generator.

The GPMC features two input wait pin:gpmc\_wait1, and gpmc\_wait0. This pin allow control of external devices with different wait-pin polarity. They also allow the overlap of wait-pin assertion from different devices without affecting access to devices for which the wait pin is not asserted.

- The GPMC\_CONFIG1\_i[17-16] WAITPINSELECT bit field (where i = 0 to 6) selects which input gpmc\_wait pin is used for the device attached to the corresponding chip-select.
- The polarity of the wait pin is defined through the WAITxPINPOLARITY bit of the GPMC\_CONFIG register. A wait pin configured to be active low means that low level on the WAIT signal indicates that the data is not ready and that the data bus is invalid. When WAIT is inactive, data is valid.

The GPMC access engine can be configured per CS to monitor the wait pin of the external memory device or not, based on the access type: read or write.

- The GPMC\_CONFIG1\_i[22] WAITREADMONITORING bit defines whether the wait pin should be monitored during read accesses or not.
- The GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING bit defines whether the wait pin should be monitored during write accesses or not.

The GPMC access engine can be configured to monitor the wait pin of the external memory device asynchronously or synchronously with the GPMC\_CLK clock, depending on the access type: synchronous or asynchronous (the GPMC\_CONFIG1\_i[29] READTYPE and GPMC\_CONFIG1\_i[27] WRITETYPE bits).

#### **7.1.3.3.8.3.2 Wait Monitoring During an Asynchronous Read Access**

When wait-pin monitoring is enabled for read accesses (WAITREADMONITORING), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the wait-deasserted state.

During asynchronous read accesses with wait-pin monitoring enabled, the wait pin must be at a valid level (asserted or deasserted) for at least two GPMC clock cycles before RDACCESSTIME completes, to ensure correct dynamic access-time control through wait-pin monitoring. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

In this context, RDACCESSTIME is used as a WAIT invalid timing window and is set to such a value that the wait pin is at a valid state two GPMC clock cycles before RDACCESSTIME completes.

Similarly, during a multiple-access cycle (for example, asynchronous read page mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-deasserted state. Wait-monitoring pipelining is also applicable to multiple accesses (access within a page).

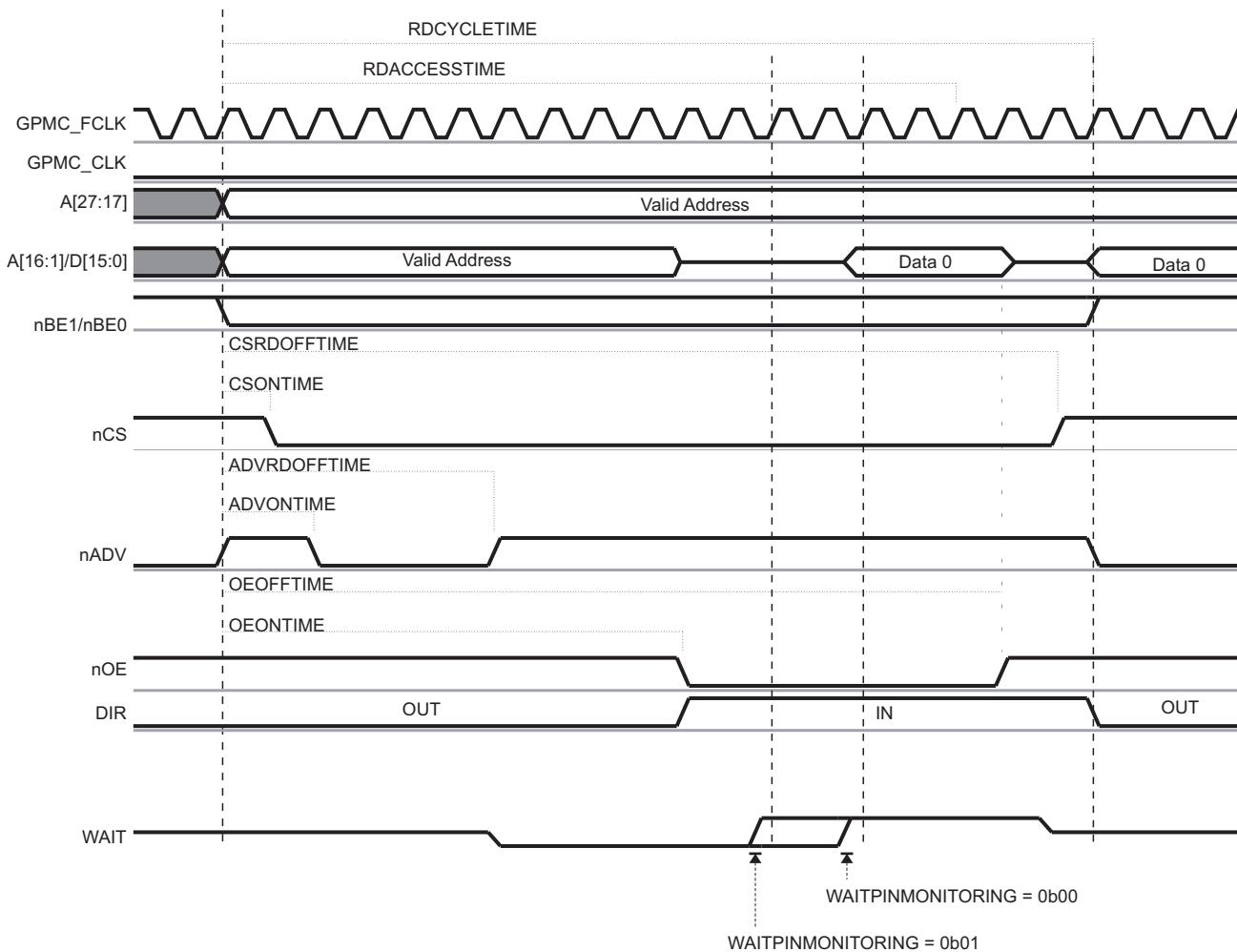
- WAIT monitored as active freezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as asserted extends the current access time in the page. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a page, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive completes the current access time and starts the next access phase in the page. The data bus is considered valid, and data are captured during this clock cycle. In case of a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their related control timing value and according to the CYCLETIME counter status.

When a delay larger than two GPMC clocks must be observed between wait-pin deactivation time and data valid time (including the required GPMC and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data-capture time and the effective unlock of the CYCLETIME counter. This extra delay can be programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME field.

- The WAITMONITORINGTIME parameter does not delay the wait-pin active or inactive detection, nor does it modify the two GPMC clocks pipelined detection delay.
- This extra delay is expressed as a number of GPMC\_CLK clock cycles, even though the access is defined as asynchronous, and no GPMC\_CLK clock is provided to the external device. Still, GPMCFCLKDIVIDER is used as a divider for the GPMC clock, so it must be programmed to define the correct WAITMONITORINGTIME delay.

Figure 7-7 shows wait behavior during an asynchronous single read access.

**Figure 7-7. Wait Behavior During an Asynchronous Single Read Access (GPMCFCLKDivider = 1)**



The WAIT signal is active low. GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME = 00b or 01b.

#### 7.1.3.3.8.3.3 Wait Monitoring During an Asynchronous Write Access

When wait-pin monitoring is enabled for write accesses (GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING bit = 1), the WAIT-invalid timing window is defined by the WRACCESSTIME field. WRACCESSTIME must be set so that the wait pin is at a valid state two GPMC clock cycles before WRACCESSTIME completes. The advance pipelining of the two GPMC clock cycles is the result of the internal synchronization requirements for the WAIT signal.

- WAIT monitored as active freezes the CYCLETIME counter. This informs the GPMC that the data bus is not captured by the external device. The control signals are kept in their current state. The data bus still drives the data.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. This informs that the data bus is correctly captured by the external device. All signals, including the data bus, are controlled according to their related control timing value and to the CYCLETIME counter status.

When a delay larger than two GPMC clock cycles must be observed between wait-pin deassertion time and the effective data write into the external device (including the required GPMC data setup time and the device data setup time), an extra delay can be added between wait-pin deassertion time detection and effective data write time into the external device and the effective unfreezing of the CYCLETIME counter. This extra delay can be programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME fields.

- The WAITMONITORINGTIME parameter does not delay the wait-pin assertion or deassertion detection, nor does it modify the two GPMC clock cycles pipelined detection delay.
- This extra delay is expressed as a number of GPMC\_CLK clock cycles, even though the access is defined as asynchronous, and even though no clock is provided to the external device. Still, GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER is used as a divider for the GPMC clock and so it must be programmed to define the correct WAITMONITORINGTIME delay.

#### **7.1.3.3.8.3.4 Wait Monitoring During a Synchronous Read Access**

During synchronous accesses with wait-pin monitoring enabled, the wait pin is captured synchronously with GPMC\_CLK, using the rising edge of this clock.

The WAIT signal can be programmed to apply to the same clock cycle it is captured in. Alternatively, it can be sampled one or two GPMC\_CLK cycles ahead of the clock cycle it applies to. This pipelining is applicable to the entire burst access, and to all data phase in the burst access. This WAIT pipelining depth is programmed in the GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME field, and is expressed as a number of GPMC\_CLK clock cycles.

In synchronous mode, when wait-pin monitoring is enabled (GPMC\_CONFIG1\_i[22] WAITREADMONITORING bit), the effective access time is a logical AND combination of the RDACCESSTIME timing completion and the WAIT deasserted-state detection.

Depending on the programmed WAITMONITORINGTIME value, the wait pin should be at a valid level, either asserted or deasserted:

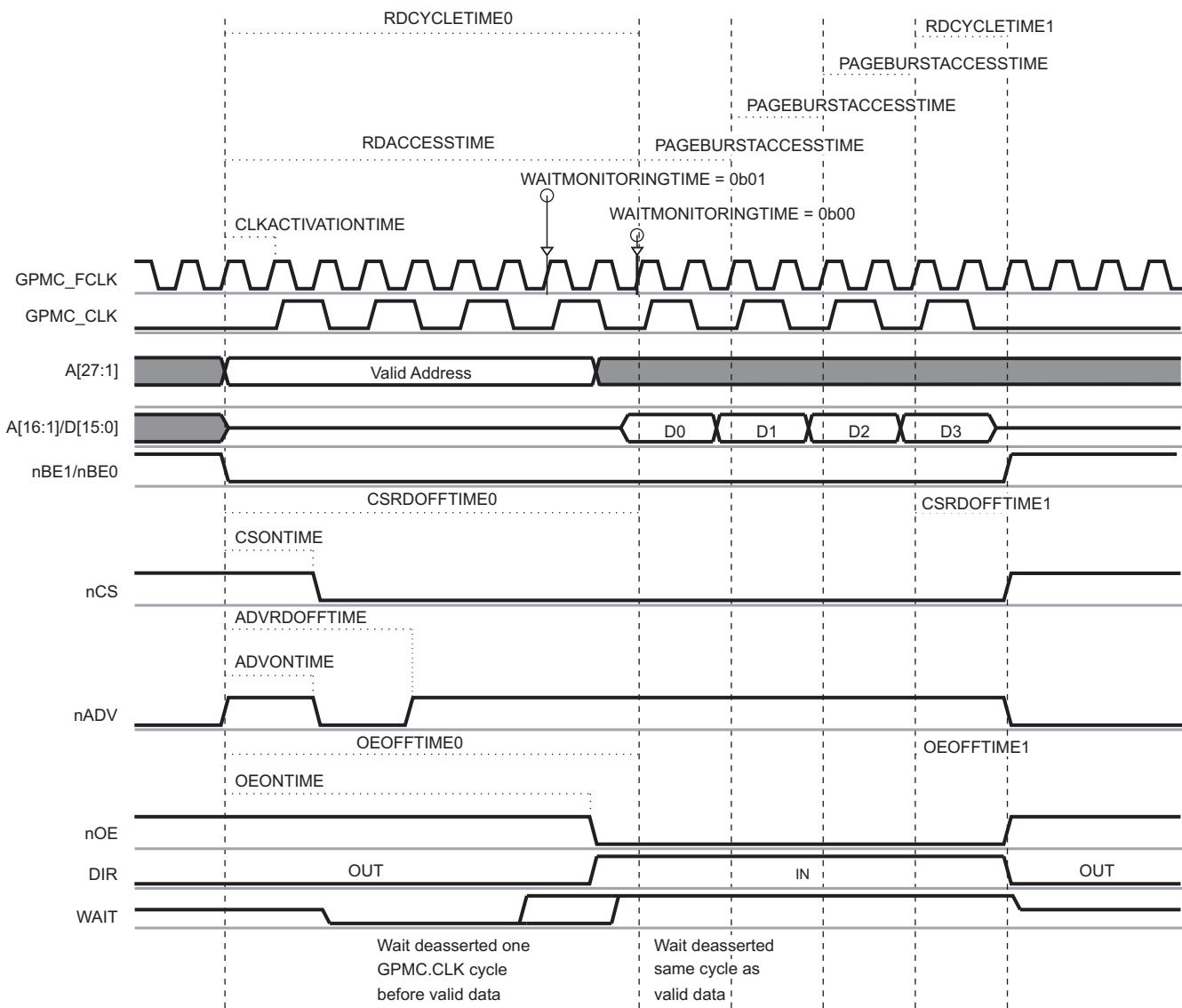
- In the same clock cycle the data is valid if WAITMONITORINGTIME = 0 (at RDACCESSTIME completion)
- In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC\_FCLK clock cycles before RDACCESSTIME completion if WAITMONITORINGTIME not equal to 0

Similarly, during a multiple-access cycle (burst mode), the effective access time is a logical AND combination of PAGEBURSTACCESSTIME timing completion and the wait-inactive state. The Wait pipelining depth programming applies to the whole burst access.

- WAIT monitored as active freezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in a lock state), WAIT monitored as active extends the current access time in the burst. Control signals are kept in their current state. The data bus is considered invalid, and no data are captured during this clock cycle.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For an access within a burst (when the CYCLETIME counter is by definition in lock state), WAIT monitored as inactive completes the current access time and starts the next access phase in the burst. The data bus is considered valid, and data are captured during this clock cycle. In a single access or if this was the last access in a multiple-access cycle, all signals are controlled according to their relative control timing value and the CYCLETIME counter status.

Figure 7-8 shows wait behavior during a synchronous read burst access.

**Figure 7-8. Wait Behavior During a Synchronous Read Burst Access**



The WAIT signal is active low. WAITMONITORINGTIME = 00b or 01b.

### **7.1.3.3.8.3.5 Wait Monitoring During a Synchronous Write Access**

During synchronous accesses with wait-pin monitoring enabled (the WAITWRITEMONITORING bit), the wait pin is captured synchronously with GPMC\_CLK, using the rising edge of this clock.

If enabled, external wait-pin monitoring can be used in combination with WRACCESSTIME to delay the effective memory device GPMC\_CLK capture edge.

Wait-monitoring pipelining depth is similar to synchronous read access:

- At WRACCESSTIME completion if WAITMONITORINGTIME = 0
  - In the WAITMONITORINGTIME x (GPMCFCLKDIVIDER + 1) GPMC\_FCLK cycles before WRACCESSTIME completion if WAITMONITORINGTIME not equal to 0.

Wait-monitoring pipelining definition applies to whole burst accesses:

- WAIT monitored as active freezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as active indicates that the data bus is not being captured by the external device. Control signals are kept in their current state. The data bus is kept in its current state.
- WAIT monitored as inactive unfreezes the CYCLETIME counter. For accesses within a burst, when the CYCLETIME counter is by definition in a lock state, WAIT monitored as inactive indicates the effective data capture of the bus by the external device and starts the next access of the burst. In case of a single access or if this was the last access in a multiple access cycle, all signals, including the data bus, are controlled according to their related control timing value and the CYCLETIME counter status.

Wait monitoring is supported for all configurations except for GPMC\_CONFIG1\_i[19-18] WAITMONITORINGTIME = 0 for write bursts with a clock divider of 1 or 2 (GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER field equal to 0 or 1, respectively).

#### **7.1.3.3.8.3.6 WAIT With NAND Device**

For details about the use of the wait pin for communication with a NAND flash external device, see [Section 7.1.3.3.12.2](#).

#### **7.1.3.3.8.3.7 Idle Cycle Control Between Successive Accesses**

##### **7.1.3.3.8.3.7.1 Bus Turnaround (BUSTURNAROUND)**

To prevent data-bus contention, an access that follows a read access to a slow memory/device must be delayed (in other words, control the CSn/OEn de-assertion to data bus in high-impedance delay).

The bus turnaround is a time-out counter starting after CSn or OEn de-assertion time, whichever occurs first, and delays the next access start-cycle time. The counter is programmed through the GPMC\_CONFIG6\_i[3-0] BUSTURNAROUND bit field.

After a read access to a chip-select with a non zero BUSTURNAROUND, the next access is delayed until the BUSTURNAROUND delay completes, if the next access is one of the following:

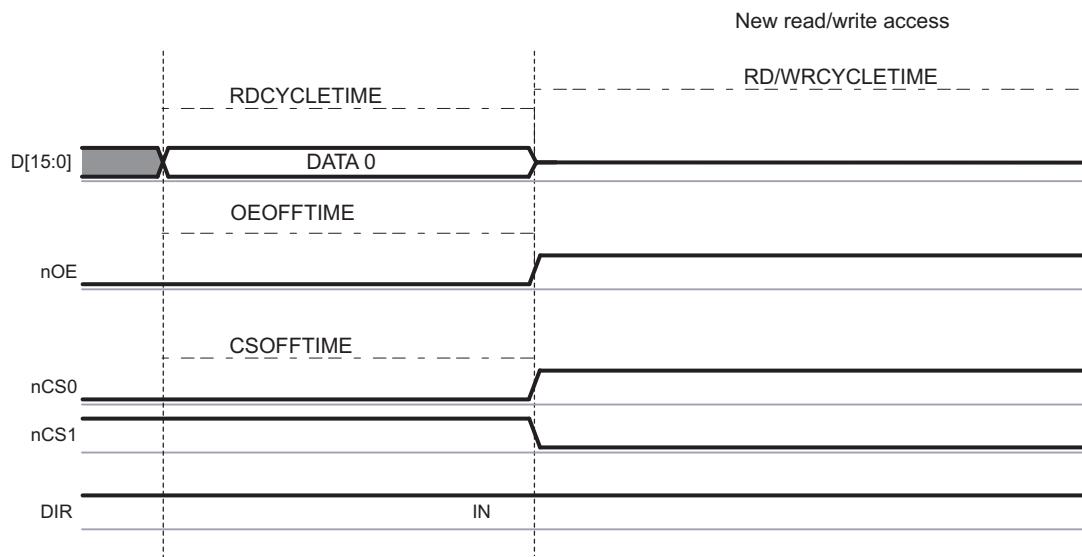
- A write access to any chip-select (same or different from the chip-select data was read from)
- A read access to a different chip-select from the chip-select data was read access from
- A read or write access to a chip-select associated with an address/data-multiplexed device

Bus keeping starts after bus turnaround completion so that DIR changes from IN to OUT after bus turnaround. The bus will not have enough time to go into high-impedance even though it could be driven with the same value before bus turnaround timing.

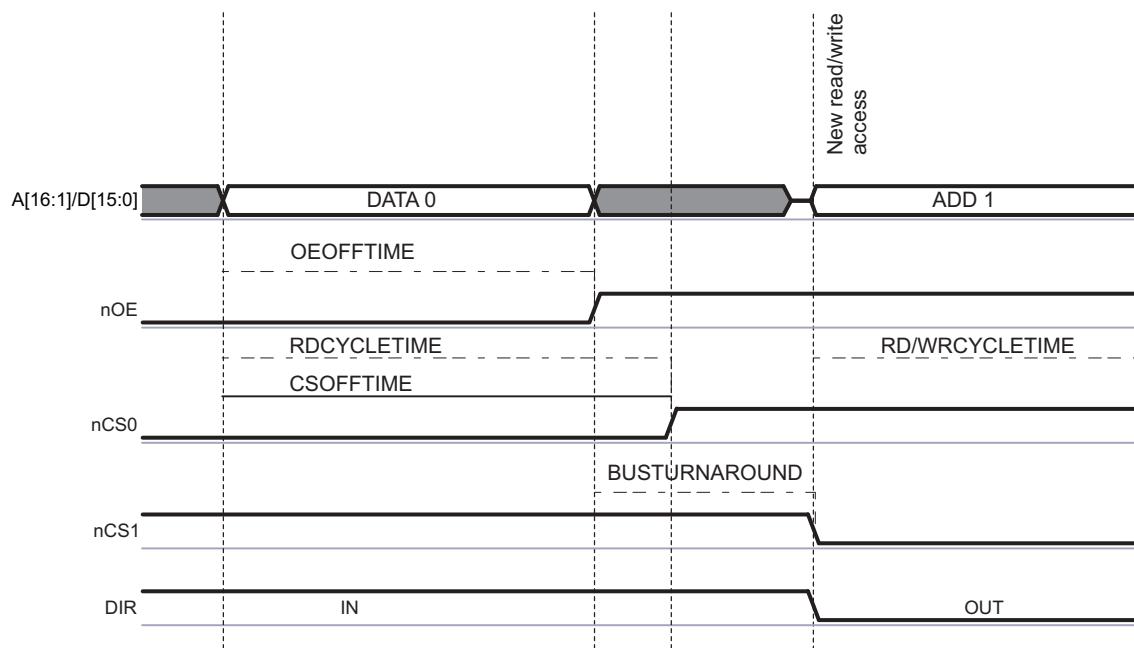
BUSTURNAROUND delay runs in parallel with GPMC\_CONFIG6\_i[3-0] CYCLE2CYCLEDELAY delays. It should be noted that BUSTURNAROUND is a timing parameter for the ending chip-select access while CYCLE2CYCLEDELAY is a timing parameter for the following chip-select access. The effective minimum delay between successive accesses is driven by these delay timing parameters and by the access type of the following access. See [Figure 7-9](#) to [Figure 7-11](#).

Another way to prevent bus contention is to define an earlier CSn or OEn deassertion time for slow devices or to extend the value of RDCYCLETIME. Doing this prevents bus contention, but affects all accesses of this specific chip-select.

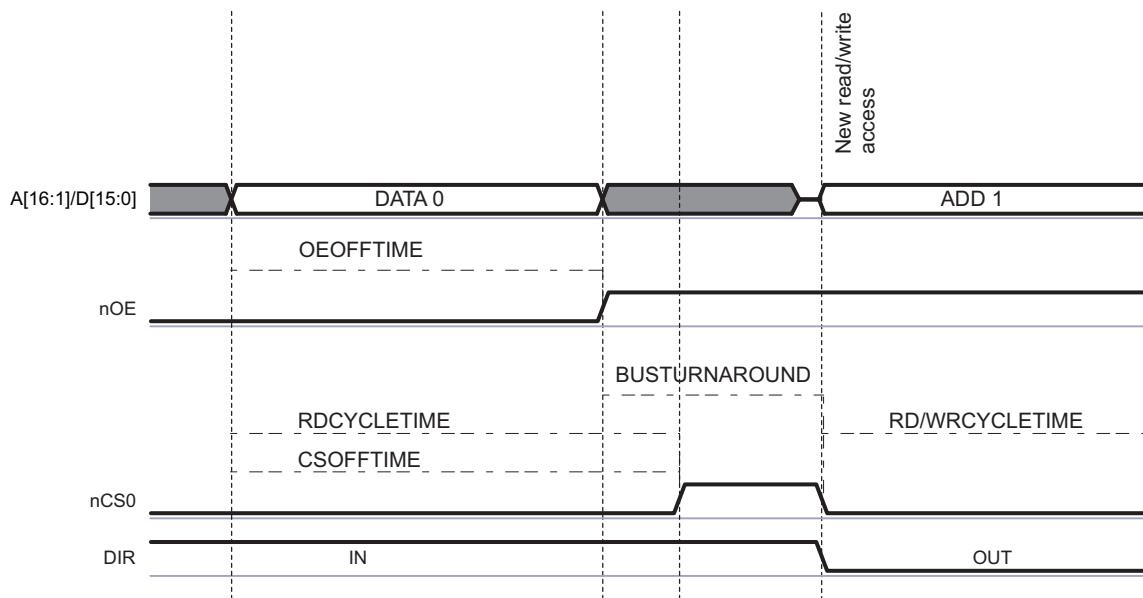
**Figure 7-9. Read to Read for an Address-Data Multiplexed Device, On Different CS, Without Bus Turnaround (CS0n Attached to Fast Device)**



**Figure 7-10. Read to Read / Write for an Address-Data Multiplexed Device, On Different CS, With Bus Turnaround**



**Figure 7-11. Read to Read / Write for a Address-Data or AAD-Multiplexed Device, On Same CS, With Bus Turnaround**



#### 7.1.3.3.8.3.7.2 Idle Cycles Between Accesses to Same Chip-Select (CYCLE2CYCLESAMECSEN, CYCLE2CYCLEDELAY)

Some devices require a minimum chip-select signal inactive time between accesses. The GPMC\_CONFIG6\_i[7] CYCLE2CYCLESAMECSEN bit enables insertion of a minimum number of GPMC\_FCLK cycles, defined by the GPMC\_CONFIG6\_i[11-8] CYCLE2CYCLEDELAY field, between successive accesses of any type (read or write) to the same chip-select.

If CYCLE2CYCLESAMECSEN is enabled, any subsequent access to the same chip-select is delayed until its CYCLE2CYCLEDELAY completes. The CYCLE2CYCLEDELAY counter starts when CSRDOFFTIME/CSWROFFTIME completes.

The same applies to successive accesses occurring during 32-bit word or burst accesses split into successive single accesses when the single-access mode is used (GPMC\_CONFIG1\_i[30] READMULTIPLE = 0 or GPMC\_CONFIG1\_i[28] WRITEMULTIPLE = 0).

All control signals are kept in their default states during these idle GPMC\_FCLK cycles. This prevents back-to-back accesses to the same chip-select without idle cycles between accesses.

#### 7.1.3.3.8.3.7.3 Idle Cycles Between Accesses to Different Chip-Select (CYCLE2CYCLEDIFFCSEN, CYCLE2CYCLEDELAY)

Because of the pipelined behavior of the system, successive accesses to different chip-selects can occur back-to-back with no idle cycles between accesses. Depending on the control signals (CSn, ADV\_ALEn, BE0\_CLEN, OE\_REn, WEn) assertion and de-assertion timing parameters and on the IC timing parameters, some control signals assertion times may overlap between the successive accesses to different CS. Similarly, some control signals (WEn, OE\_REn) may not respect required transition times.

To work around the overlapping and to observe the required control-signal transitions, a minimum of CYCLE2CYCLEDELAY inactive cycles is inserted between the access being initiated to this chip-select and the previous access ending for a different chip-select. This applies to any type of access (read or write).

If GPMC\_CONFIG6\_i[6] CYCLE2CYCLEDIFFCSEN is enabled, the chip-select access is delayed until CYCLE2CYCLEDELAY cycles have expired since the end of a previous access to a different chip-select. CYCLE2CYCLEDELAY count starts at CSRDOFFTIME/CSWROFFTIME completion. All control signals are kept inactive during the idle GPMC\_FCLK cycles.

CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDIFFCSEN should be set in registers to respectively get idle cycles inserted between accesses on this chip-select and after accesses to a different chip-select.

The CYCLE2CYCLEDELAY delay runs in parallel with the BUSTURNAROUND delay. It should be noted that BUSTURNAROUND is a timing parameter defined for the ending chip-select access, whereas CYCLE2CYCLEDELAY is a timing parameter defined for the starting chip-select access. The effective minimum delay between successive accesses is based on the larger delay timing parameter and on access type combination, since bus turnaround does not apply to all access types. See [Section 7.1.3.3.8.3.7.1](#) for more details on bus turnaround.

[Table 7-10](#) describes the configuration required for idle cycle insertion.

**Table 7-10. Idle Cycle Insertion Configuration**

| First Access Type | BUSTURN AROUND Timing Parameter | Second Access Type | Chip-Select | Addr/Data Multiplexed | CYCLE2 CYCLE SAMECSEN Parameter | CYCLE2 CYCLE DIFFCSEN Parameter | Idle Cycle Insertion Between the Two Accesses  |
|-------------------|---------------------------------|--------------------|-------------|-----------------------|---------------------------------|---------------------------------|--|
| R/W               | 0                               | R/W                | Any         | Any                   | 0                               | x                               | No idle cycles are inserted if the two accesses are well pipelined.  |
| R                 | >0                              | R                  | Same        | Nonmuxed              | x                               | 0                               | No idle cycles are inserted if the two accesses are well pipelined.  |
| R                 | >0                              | R                  | Different   | Nonmuxed              | 0                               | 0                               | BUSTURNAROUND cycles are inserted.   |
| R                 | >0                              | R/W                | Any         | Muxed                 | 0                               | 0                               | BUSTURNAROUND cycles are inserted.   |
| R                 | >0                              | W                  | Any         | Any                   | 0                               | 0                               | BUSTURNAROUND cycles are inserted.   |
| W                 | >0                              | R/W                | Any         | Any                   | 0                               | 0                               | No idle cycles are inserted if the two accesses are well pipelined.  |
| R/W               | 0                               | R/W                | Same        | Any                   | 1                               | x                               | CYCLE2CYCLEDELAY cycles are inserted.  |
| R/W               | 0                               | R/W                | Different   | Any                   | x                               | 1                               | CYCLE2CYCLEDELAY cycles are inserted.  |
| R/W               | >0                              | R/W                | Same        | Any                   | 1                               | x                               | CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is max (BUSTURNAROUND, CYCLE2CYCLEDELAY).     |
| R/W               | >0                              | R/W                | Different   | Any                   | x                               | 1                               | CYCLE2CYCLEDELAY cycles are inserted. If BTA idle cycles already apply on these two back-to-back accesses, the effective delay is maximum (BUSTURNAROUND, CYCLE2CYCLEDELAY). |

#### **7.1.3.3.8.3.8 Slow Device Support (*TIMEPARAGRANULARITY* Parameter)**

All access-timing parameters can be multiplied by 2 by setting the GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY bit. Increasing all access timing parameters allows support of slow devices.

#### **7.1.3.3.8.3.9 GPMC\_DIR Pin**

The GPMC\_DIR pin is used to control I/O direction on the GPMC data bus GPMC\_D[15-0]. Depending on top-level pad multiplexing, this signal can be output and used externally to the device, if required. The GPMC\_DIR pin is low during transmit (OUT) and high during receive (IN).

For write accesses, the GPMC\_DIR pin stays OUT from start-cycle time to end-cycle time.

For read accesses, the GPMC\_DIR pin goes from OUT to IN at OEn assertion time and stays IN until:

- BUSTURNAROUND is enabled
  - The GPMC\_DIR pin goes from IN to OUT at end-cycle time plus programmable bus turnaround time.
- BUSTURNAROUND is disabled
  - After an asynchronous read access, the GPMC\_DIR pin goes from IN to OUT at RDACCESSTIME + 1 GPMC\_FCLK cycle or when RDCYCLETIME completes, whichever occurs last.
  - After a synchronous read access, the GPMC\_DIR pin goes from IN to OUT at RDACCESSTIME + 2 GPMC\_FCLK cycles or when RDCYCLETIME completes, whichever occurs last.

Because of the bus-keeping feature of the GPMC, after a read or write access and with no other accesses pending, the default value of the GPMC\_DIR pin is OUT (see [Section 7.1.3.3.9.10](#)). In nonmultiplexed devices, the GPMC\_DIR pin stays IN between two successive read accesses to prevent unnecessary toggling.

#### **7.1.3.3.8.3.10 Reset**

No reset signal is sent to the external memory device by the GPMC. For more information about external-device reset, see [Chapter 8, Power, Reset, and Clock Management \(PRCM\)](#).

The PRCM module provides an input pin, global\_rst\_n, to the GPMC:

- The global\_rst\_n pin is activated during device warm reset and cold reset.
- The global\_rst\_n pin initializes the internal state-machine and the internal configuration registers.

#### **7.1.3.3.8.3.11 Write Protect Signal (WPn)**

When connected to the attached memory device, the write protect signal can enable or disable the lockdown function of the attached memory. The GPMC\_WPn output pin value is controlled through the GPMC\_CONFIG[4] WRITEPROTECT bit, which is common to all CS.

#### **7.1.3.3.8.3.12 Byte Enable (BE1n/BE0n)**

Byte enable signals (BE1n/BE0n) are:

- Valid (asserted or nonasserted according to the incoming system request) from access start to access completion for asynchronous and synchronous single accesses
- Asserted low from access start to access completion for asynchronous and synchronous multiple read accesses
- Valid (asserted or nonasserted, according to the incoming system request) synchronously to each written data for synchronous multiple write accesses

#### **7.1.3.3.8.4 Error Handling**

When an error occurs in the GPMC, the error information is stored in the GPMC\_ERR\_TYPE register and the address of the illegal access is stored in the GPMC\_ERR\_ADDRESS register. The GPMC keeps only the first error abort information until the GPMC\_ERR\_TYPE register is reset. Subsequent accesses that cause errors are not logged until the error is cleared by hardware with the GPMC\_ERR\_TYPE[0]ERRORVALID bit.

- ERRORNOTSUPPADD occurs when an incoming system request address decoding does not match any valid chip-select region, or if two chip-select regions are defined as overlapped, or if a register file access is tried outside the valid address range of 1KB.
- ERRORNOTSUPPMCMD occurs when an unsupported command request is decoded at the L3 Slow interconnect interface
- ERRORTIMEOUT: A time-out mechanism prevents the system from hanging. The start value of the 9-bit time-out counter is defined in the GPMC\_TIMEOUT\_CONTROL register and enabled with the GPMC\_TIMEOUT\_CONTROL[0] TIMEOUTENABLE bit. When enabled, the counter starts at start-cycle time until it reaches 0 and data is not responded to from memory, and then a time-out error occurs. When data are sent from memory, this counter is reset to its start value. With multiple accesses (asynchronous page mode or synchronous burst mode), the counter is reset to its start value for each data access within the burst.

The GPMC does not generate interrupts on these errors. True abort to the MPU or interrupt generation is handled at the interconnect level.

#### 7.1.3.3.9 Timing Setting

The GPMC offers the maximum flexibility to support various access protocols. Most of the timing parameters of the protocol access used by the GPMC to communicate with attached memories or devices are programmable on a chip-select basis. Assertion and deassertion times of control signals are defined to match the attached memory or device timing specifications and to get maximum performance during accesses. For more information on GPMC\_CLK and GPMC\_FCLK see [Section 7.1.3.3.9.6](#).

In the following sections, the start access time refer to the time at which the access begins.

##### 7.1.3.3.9.1 Read Cycle Time and Write Cycle Time (RDCYCLETIME / WRCYCLETIME)

The GPMC\_CONFIG5\_i[4-0] RDCYCLETIME and GPMC\_CONFIG5\_i[12-8] WRCYCLETIME fields define the address bus and byte enables valid times for read and write accesses. To ensure a correct duty cycle of GPMC\_CLK between accesses, RDCYCLETIME and WRCYCLETIME are expressed in GPMC\_FCLK cycles and must be multiples of the GPMC\_CLK cycle. RDCYCLETIME and WRCYCLETIME bit fields can be set with a granularity of 1 or 2 through GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY.

When either RDCYCLETIME or WRCYCLETIME completes, if they are not already deasserted, all control signals (CSn, ADV\_ALEn, OE\_REn, WEn, and BE0\_CLEN) are deasserted to their reset values, regardless of their deassertion time parameters.

An exception to this forced deassertion occurs when a pipelined request to the same chip-select or to a different chip-select is pending. In such a case, it is not necessary to deassert a control signal with deassertion time parameters equal to the cycle-time parameter. This exception to forced deassertion prevents any unnecessary glitches. This requirement also applies to BE signals, thus avoiding an unnecessary BE glitch transition when pipelining requests.

If no inactive cycles are required between successive accesses to the same or to a different chip-select (GPMC\_CONFIG6\_i[7] CYCLE2CYCLESAMECSEN = 0 or GPMC\_CONFIG6\_i[6] CYCLE2CYCLEDIFFCSEN = 0, where i = 0 to 3), and if assertion-time parameters associated with the pipelined access are equal to 0, asserted control signals (CSn, ADV\_ALEn, BE0\_CLEN, WEn, and OE\_REn) are kept asserted. This applies to any read/write to read/write access combination.

If inactive cycles are inserted between successive accesses, that is, CYCLE2CYCLESAMECSEN = 1 or CYCLE2CYCLEDIFFCSEN = 1, the control signals are forced to their respective default reset values for the number of GPMC\_FCLK cycles defined in CYCLE2CYCLEDELAY.

##### 7.1.3.3.9.2 CSn: Chip-Select Signal Control Assertion/Deassertion Time (CSONTIME / CSRDOFFTIME / CSWROFFTIME / CSEXTRADELAY)

The GPMC\_CONFIG2\_i[3-0] CSONTIME field defines the CSn signal-assertion time relative to the start access time. It is common for read and write accesses.

The GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME (read access) and GPMC\_CONFIG2\_i[20-16] CSWROFFTIME (write access) bit fields define the CSn signal deassertion time relative to start access time.

CSONTIME, CSRDOFFTIME and CSWROFFTIME parameters are applicable to synchronous and asynchronous modes. CSONTIME can be used to control an address and byte enable setup time before chip-select assertion. CSRDOFFTIME and CSWROFFTIME can be used to control an address and byte enable hold time after chip-select deassertion.

CSn signal transitions as controlled through CSONTIME, CSRDOFFTIME, and CSWROFFTIME can be delayed by half a GPMC\_FCLK period by enabling the GPMC\_CONFIG2\_i[7] CSEXTRADELAY bit. This half of a GPMC\_FCLK period provides more granularity on the CSn assertion and deassertion time to guarantee proper setup and hold time relative to GPMC\_CLK. CSEXTRADELAY is especially useful in configurations where GPMC\_CLK and GPMC\_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, CSEXTRADELAY applies to all parameters controlling CSn transitions.

The CSEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than the CSn signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

#### **7.1.3.3.9.3 ADV<sub>n</sub>/ALE: Address Valid/Address Latch Enable Signal Control Assertion/Deassertion Time (ADVONTIME / ADVRDOFFTIME / ADVWROFFTIME / ADVEXTRADELAY/ADVAADMUXONTIME/ADVAADMUXRDOFFTIME/ADVAADMUXWROFFTIME )**

The GPMC\_CONFIG3\_i[3-0] ADVONTIME field defines the ADV<sub>n</sub>\_ALE signal-assertion time relative to start access time. It is common to read and write accesses.

The GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME (read access) and GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME (write access) bit fields define the ADV<sub>n</sub>\_ALE signal-deassertion time relative to start access time.

ADVONTIME can be used to control an address and byte enable valid setup time control before ADV<sub>n</sub>\_ALE assertion. ADVRDOFFTIME and ADVWROFFTIME can be used to control an address and byte enable valid hold time control after ADV<sub>n</sub>\_ALE de-assertion. ADVRDOFFTIME and ADVWROFFTIME are applicable to both synchronous and asynchronous modes.

ADV<sub>n</sub>\_ALE signal transitions as controlled through ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME can be delayed by half a GPMC\_FCLK period by enabling the GPMC\_CONFIG3\_i[7] ADVEXTRADELAY bit. This half of a GPMC\_FCLK period provides more granularity on ADV<sub>n</sub>\_ALE assertion and deassertion time to assure proper setup and hold time relative to GPMC\_CLK. The ADVEXTRADELAY configuration parameter is especially useful in configurations where GPMC\_CLK and GPMC\_FCLK have the same frequency, but can be used for all GPMC configurations. If enabled, ADVEXTRADELAY applies to all parameters controlling ADV<sub>n</sub>\_ALE transitions.

ADVEXTRADELAY must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the RDCYCLETIME and WRCYCLETIME bit fields to be greater than ADV<sub>n</sub>\_ALE signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

The GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME, GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME, and GPMC\_CONFIG3\_i[30-28] ADVAADMUXWROFFTIME parameters have the same functions as ADVONTIME, ADVRDOFFTIME, and ADVWROFFTIME, but apply to the first address phase in the AAD-multiplexed protocol. It is the user responsibility to make sure ADVAADMUXxxOFFTIME is programmed to a value lower than or equal to ADVxxOFFTIME. Functionality in AAD-mux mode is undefined if the settings do not comply with this requirement. ADVAADMUXxxOFFTIME can be programmed to the same value as ADVONTIME if no high ADV<sub>n</sub> pulse is needed between the two AAD-mux address phases, which is the typical case in synchronous mode. In this configuration, ADV<sub>n</sub> is kept low until it reaches the correct ADVxxOFFTIME.

See [Section 7.1.3.3.12](#) for more details on ADVONTIME, ADVRDOFFTIME, ADVWROFFTIME, and ADVAADMUXRDOFFTIME, ADVAADMUXWROFFTIME usage for CLE and ALE (Command / Address Latch Enable) usage for a NAND Flash interface.

#### **7.1.3.3.9.4 OEn/REn: Output Enable / Read Enable Signal Control Assertion / Deassertion Time (OEONTIME / OEOFETIME / OEEXTRADELAY / OEAADMUXONTIME / OEAADMUXOFFTIME)**

The GPMC\_CONFIG4\_i[3-0] OEONTIME field defines the OEn\_REn signal assertion time relative to start access time. It is applicable only to read accesses.

The GPMC\_CONFIG4\_i[12-8] OEOFETIME field defines the OEn\_REn signal deassertion time relative to start access time. It is applicable only to read accesses. OEn\_REn is not asserted during a write cycle.

OEONTIME, OEOFETIME, OEAADMUXONTIME and OEAADMUXOFFTIME parameters are applicable to synchronous and asynchronous modes. OEONTIME can be used to control an address and byte enable valid setup time control before OEn\_REn assertion. OEOFETIME can be used to control an address and byte enable valid hold time control after OEn\_REn assertion.

OEAADMUXONTIME and OEAADMUXOFFTIME parameters have the same functions as OEONTIME and OEOFETIME, but apply to the first OE assertion in the AAD-multiplexed protocol for a read phase, or to the only OE assertion for a write phase. It is the user responsibility to make sure OEAADMUXOFFTIME is programmed to a value lower than OEONTIME. Functionality in AAD-mux mode is undefined if the settings do not comply with this requirement. OEAADMUXOFFTIME shall never be equal to OEONTIME because the AAD-mux protocol requires a second address phase with the OEn signal de-asserted before OEn can be asserted again to define a read command.

The OEn\_REn signal transitions as controlled through OEONTIME, OEOFETIME, OEAADMUXONTIME and OEAADMUXOFFTIME can be delayed by half a GPMC\_FCLK period by enabling the GPMC\_CONFIG4\_i[7] OEEXTRADELAY bit. This half of a GPMC\_FCLK period provides more granularity on OEn\_REn assertion and deassertion time to assure proper setup and hold time relative to GPMC\_CLK. If enabled, OEEXTRADELAY applies to all parameters controlling OEn\_REn transitions.

OEEXTRADELAY must be used carefully, to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program RDCYCLETIME and WRCYCLETIME to be greater than OEn\_REn signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

When the GPMC generates a read access to an address-/data-multiplexed device, it drives the address bus until OEn assertion time.

#### **7.1.3.3.9.5 WEn: Write Enable Signal Control Assertion / Deassertion Time (WEONTIME / WEOFETIME / WEEXTRADELAY)**

The GPMC\_CONFIG4\_i[19-16] WEONTIME field (where i = 0 to 3) defines the WEn signal-assertion time relative to start access time. The GPMC\_CONFIG4\_i[28-24] WEOFETIME field defines the WEn signal-deassertion time relative to start access time. These bit fields only apply to write accesses. WEn is not asserted during a read cycle.

WEONTIME can be used to control an address and byte enable valid setup time control before WEn assertion. WEOFETIME can be used to control an address and byte enable valid hold time control after WEn assertion.

WEn signal transitions as controlled through WEONTIME, and WEOFETIME can be delayed by half a GPMC\_FCLK period by enabling the GPMC\_CONFIG4\_i[23] WEEXTRADELAY bit. This half of a GPMC\_FCLK period provides more granularity on WEn assertion and deassertion time to guaranty proper setup and hold time relative to GPMC\_CLK. If enabled, WEEXTRADELAY applies to all parameters controlling WEn transitions.

The WEEXTRADELAY bit must be used carefully to avoid control-signal overlap between successive accesses to different chip-selects. This implies the need to program the WRCYCLETIME bit field to be greater than the WEn signal-deassertion time, including the extra half-GPMC\_FCLK-period delay.

### 7.1.3.3.9.6 GPMC\_CLK

GPMC\_CLK is the external clock provided to the attached synchronous memory or device.

- The GPMC\_CLK clock frequency is the GPMC\_FCLK functional clock frequency divided by 1, 2, 3, or 4, depending on the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER bit field, with a guaranteed 50-percent duty cycle.
- The GPMC\_CLK clock is only activated when the access in progress is defined as synchronous (read or write access).
- The GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME field defines the number of GPMC\_FCLK cycles from start access time to GPMC\_CLK activation.
- The GPMC\_CLK clock is stopped when cycle time completes and is asserted low between accesses.
- The GPMC\_CLK clock is kept low when access is defined as asynchronous.
- When the GPMC is configured for synchronous mode, the GPMC\_CLK signal (which is an output) must also be set as an input in the Pin Mux configuration for the pin. GPMC\_CLK is looped back through the output and input buffers of the corresponding GPMC\_CLK pad at the device boundary. The looped-back clock is used to synchronize the sampling of the memory signals.

When cycle time completes, the GPMC\_CLK may be high because of the GPMCFCLKDIVIDER bit field. To ensure correct stoppage of the GPMC\_CLK clock within the 50-percent required duty cycle, it is the user's responsibility to extend the RDCYCLETIME or WRCYCLETIME value.

To ensure a correct external clock cycle, the following rules must be applied:

- (RDCYCLETIME - CLKACTIVATIONTIME) must be a multiple of (GPMCFCLKDIVIDER + 1).
- The PAGEBURSTACCESSTIME value must be a multiple of (GPMCFCLKDIVIDER + 1).

### 7.1.3.3.9.7 GPMC\_CLK and Control Signals Setup and Hold

Control-signal transition (assertion and deassertion) setup and hold values with respect to the GPMC\_CLK edge can be controlled in the following ways:

- For the GPMC\_CLK signal, the GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME field allows setup and hold control of control-signal assertion time.
- The use of a divided GPMC\_CLK allows setup and hold control of control-signal assertion and deassertion times.
- When GPMC\_CLK runs at the GPMC\_FCLK frequency so that GPMC\_CLK edge and control-signal transitions refer to the same GPMC\_FCLK edge, the control-signal transitions can be delayed by half of a GPMC\_FCLK period to provide minimum setup and hold times. This half-GPMC\_FCLK delay is enabled with the CSEXTRADELAY, ADVEXTRADELAY, OEXTRADELAY, or WEEEXTRADELAY parameter. This delay must be used carefully to prevent control-signal overlap between successive accesses to different chip-selects. This implies that the RDCYCLETIME and WRCYCLETIME are greater than the last control-signal deassertion time, including the extra half-GPMC\_FCLK cycle.

### 7.1.3.3.9.8 Access Time (RDACCESSTIME / WRACCESSTIME)

The read access time and write access time durations can be programmed independently through GPMC\_CONFIG5\_i[20-16] RDACCESSTIME and GPMC\_CONFIG6\_i[28-24] WRACCESSTIME. This allows OE<sub>n</sub> and GPMC data capture timing parameters to be independent of WE<sub>n</sub> and memory device data capture timing parameters. RDACCESSTIME and WRACCESSTIME bit fields can be set with a granularity of 1 or 2 through GPMC\_CONFIG1\_i[4] TIMEPARAGRANULARITY.

#### 7.1.3.3.9.8.1 Access Time on Read Access

In asynchronous read mode, for single and paged accesses, GPMC\_CONFIG5\_i[20-16] RDACCESSTIME field defines the number of GPMC\_FCLK cycles from start access time to the GPMC\_FCLK rising edge used for the first data capture. RDACCESSTIME must be programmed to the rounded greater value (in GPMC\_FCLK cycles) of the read access time of the attached memory device.

In synchronous read mode, for single or burst accesses, RDACCESSTIME defines the number of GPMC\_FCLK cycles from start access time to the GPMC\_FCLK rising edge corresponding to the GPMC\_CLK rising edge used for the first data capture.

GPMC\_CLK which is sent to the memory device for synchronization with the GPMC controller, is internally retimed to correctly latch the returned data. GPMC\_CONFIG5\_i[4-0] RDCYCLETIME must be greater than RDACCESSTIME in order to let the GPMC latch the last return data using the internally retimed GPMC\_CLK.

The external WAIT signal can be used in conjunction with RDACCESSTIME to control the effective GPMC data-capture GPMC\_FCLK edge on read access in both asynchronous mode and synchronous mode. For details about wait monitoring, see [Section 7.1.3.3.8.1](#).

#### 7.1.3.3.9.8.2 Access Time on Write Access

In asynchronous write mode, the GPMC\_CONFIG6\_i[[28-24] WRACCESSTIME timing parameter is not used to define the effective write access time. Instead, it is used as a WAIT invalid timing window, and must be set to a correct value so that the gpmc\_wait pin is at a valid state two GPMC\_CLK cycles before WRACCESSTIME completes. For details about wait monitoring, see [Section 7.1.3.3.8.1](#).

In synchronous write mode, for single or burst accesses, WRACCESSTIME defines the number of GPMC\_FCLK cycles from start access time to the GPMC\_CLK rising edge used by the memory device for the first data capture.

The external WAIT signal can be used in conjunction with WRACCESSTIME to control the effective memory device data capture GPMC\_CLK edge for a synchronous write access. For details about wait monitoring, see [Section 7.1.3.3.8.1](#).

#### 7.1.3.3.9.9 Page Burst Access Time (PAGEBURSTACCESSTIME)

GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME bit field can be set with a granularity of 1 or 2 through the GPMC\_CONFIG1\_i[[4] TIMEPARAGRANULARITY.

#### 7.1.3.3.9.9.1 Page Burst Access Time on Read Access

In asynchronous page read mode, the delay between successive word captures in a page is controlled through the PAGEBURSTACCESSTIME bit field. The PAGEBURSTACCESSTIME parameter must be programmed to the rounded greater value (in GPMC\_FCLK cycles) of the read access time of the attached device.

In synchronous burst read mode, the delay between successive word captures in a burst is controlled through the PAGEBURSTACCESSTIME field.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective GPMC data capture GPMC\_FCLK edge on read access. For details about wait monitoring, see [Section 7.1.3.3.8.1](#).

#### 7.1.3.3.9.9.2 Page Burst Access Time on Write Access

Asynchronous page write mode is not supported. PAGEBURSTACCESSTIME is irrelevant in this case.

In synchronous burst write mode, PAGEBURSTACCESSTIME controls the delay between successive memory device word captures in a burst.

The external WAIT signal can be used in conjunction with PAGEBURSTACCESSTIME to control the effective memory-device data capture GPMC\_CLK edge in synchronous write mode. For details about wait monitoring, see [Section 7.1.3.3.8.1](#).

#### 7.1.3.3.9.10 Bus Keeping Support

At the end-cycle time of a read access, if no other access is pending, the GPMC drives the bus with the last data read after RDCYCLETIME completion time to prevent bus floating and reduce power consumption.

After a write access, if no other access is pending, the GPMC keeps driving the data bus after WRCYCLETIME completes with the same data to prevent bus floating and power consumption.

### 7.1.3.3.10 NOR Access Description

For each chip-select configuration, the read access can be specified as either asynchronous or synchronous access through the GPMC\_CONFIG1\_i[29] READTYPE bit. For each chip-select configuration, the write access can be specified as either synchronous or asynchronous access through the GPMC\_CONFIG1\_i[27] WRITETYPE bit.

Asynchronous and synchronous read and write access time and related control signals are controlled through timing parameters that refer to GPMC\_FCLK. The primary difference of synchronous mode is the availability of a configurable clock interface (GPMC\_CLK) to control the external device. Synchronous mode also affects data-capture and wait-pin monitoring schemes in read access.

For details about asynchronous and synchronous access, see the descriptions of GPMC\_CLK, RdAccessTime, WrAccessTime, and wait-pin monitoring.

For more information about timing-parameter settings, see the sample timing diagrams in this chapter.

The address bus and BE[1:0]n are fixed for the duration of a synchronous burst read access, but they are updated for each beat of an asynchronous page-read access.

#### 7.1.3.3.10.1 Asynchronous Access Description

This section describes:

- Asynchronous single read operation on an address/data multiplexed device
- Asynchronous single write operation on an address/data-multiplexed device
- Asynchronous single read operation on an AAD-multiplexed device
- Asynchronous single write operation on an AAD-multiplexed device
- Asynchronous multiple (page) read operation on a non-multiplexed device

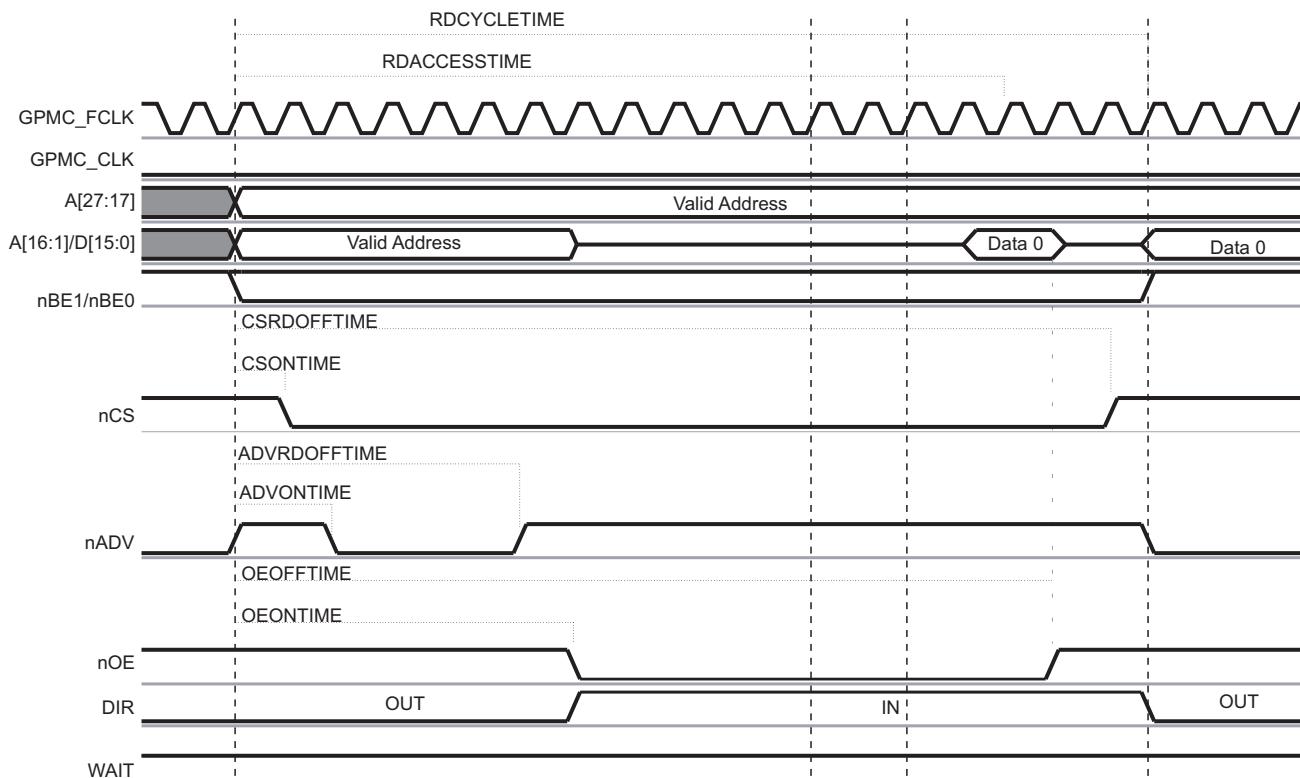
In asynchronous operations GPMC\_CLK is not provided outside the GPMC and is kept low.

### 7.1.3.3.10.1.1 Access on Address/Data Multiplexed Devices

#### 7.1.3.3.10.1.1.1 Asynchronous Single-Read Operation on an Address/Data Multiplexed Device

Figure 7-12 shows an asynchronous single read operation on an address/data-multiplexed device.

**Figure 7-12. Asynchronous Single Read Operation on an Address/Data Multiplexed Device**



#### 7.1.3.3.10.1.1.2 Asynchronous Single Read on an Address/Data-Multiplexed Device

See [Section 7.1.3.9.1](#) for formulas to calculate timing parameters.

[Table 7-41](#) lists the timing bit fields to set up in order to configure the GPMC in asynchronous single read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until OEn assertion time. For details, see [Section 7.1.3.3.8.2.3](#).

Address bits (A[16:1] from a GPMC perspective, A[15:0] from an external device perspective) are placed on the address/data bus, and the remaining address bits GPMC\_A[25:16] are placed on the address bus. The address phase ends at OEn assertion, when the DIR signal goes from OUT to IN.

- Chip-select signal CSn
  - CSn assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME field. It controls the address setup time to CSn assertion.
  - CSn deassertion time is controlled by the GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME field. It controls the address hold time from CSn deassertion
- Address valid signal ADVn
  - ADVn assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADVn deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME field.

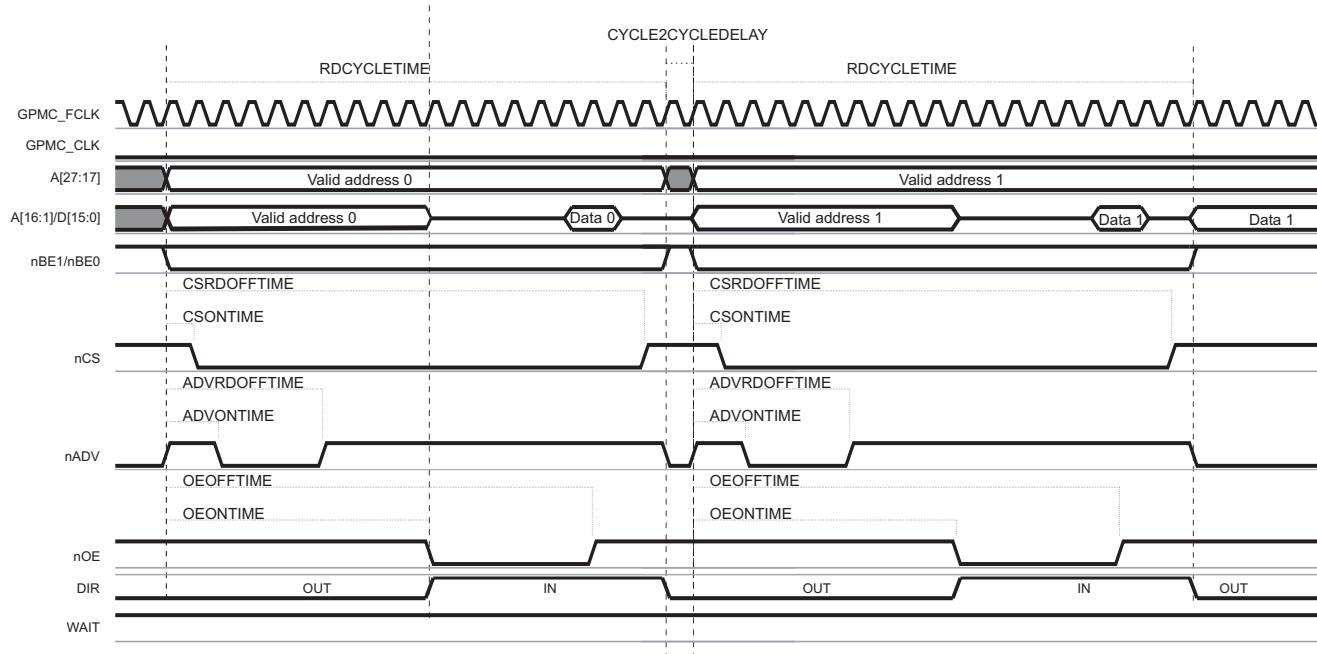
- Output enable signal OEn
  - OEn assertion indicates a read cycle.
  - OEn assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME field.
  - OEn deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFETIME field.
- Read data is latched when RDACCESSTIME completes. Access time is defined in the GPMC\_CONFIG5\_i[20-16] RDACCESSTIME field.
- Direction signal DIR: DIR goes from OUT to IN at the same time that OEn is asserted.
- The end of the access is defined by the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME parameter.

In the GPMC, when a 16-bit wide device is attached to the controller, a 32-bit word write access is split into two 16-bit word write accesses. For more information about GPMC access size and type adaptation, see [Section 7.1.3.3.10.5](#). Between two successive accesses, if a OEn pulse is needed:

- The GPMC\_CONFIG6\_i[11-8] CYCLE2CYCLEDELAY field can be programmed with GPMC\_CONFIG6\_i[7] CYCLE2CYCLESAMECSEN enabled.
- The CSWROFFTIME and CSONTIME parameters also allow a chip-select pulse, but this affects all other types of access.

[Figure 7-13](#) shows two asynchronous single-read accesses on an address/data-multiplexed device.

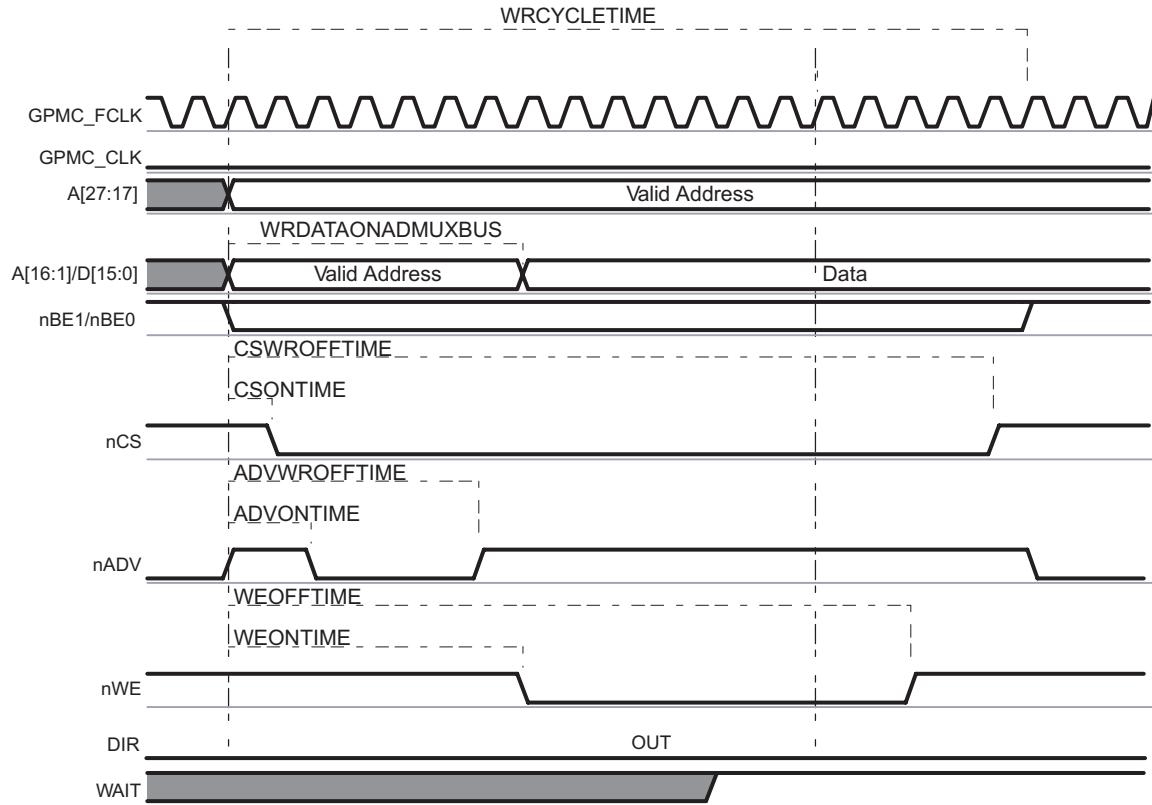
**Figure 7-13. Two Asynchronous Single Read Accesses on an Address/Data Multiplexed Device (32-Bit Read Split Into 2 × 16-Bit Read)**



### 7.1.3.3.10.1.1.3 Asynchronous Single Write Operation on an Address/Data-Multiplexed Device

Figure 7-14 shows an asynchronous single write operation on an address/data-multiplexed device.

**Figure 7-14. Asynchronous Single Write on an Address/Data-Multiplexed Device**



### 7.1.3.3.10.1.1.4 Asynchronous Single Write on an Address/Data-Multiplexed Device

See [Section 7.1.3.9.1](#) for formulas to calculate timing parameters.

[Table 7-41](#) lists the timing bit fields to set up in order to configure the GPMC in asynchronous single write mode. When the GPMC generates a write access to an address/data-multiplexed device, it drives the address bus until WEn assertion time. For more information, see [Section 7.1.3.3.8.2.3](#).

The CSn and ADVn signals are controlled in the same way as for asynchronous single read operation on an address/data-multiplexed device.

- Write enable signal WEn
  - WEn assertion indicates a write cycle.
  - WEn assertion time is controlled by the GPMC\_CONFIG4\_i[19-16] WEONTIME field.
  - WEn deassertion time is controlled by the GPMC\_CONFIG4\_i[28-24] WEOFETIME field.
- Direction signal DIR: DIR signal is OUT during the entire access.
- The end of the access is defined by the GPMC\_CONFIG5\_i[12-8] WRCYCLETIME parameter.

Address bits A[16:1] (GPMC point of view) are placed on the address/data bus at the start of cycle time, and the remaining address bits A[26:17] are placed on the address bus.

Data is driven on the address/data bus at a GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS time.

Write multiple access in asynchronous mode is not supported. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

After a write operation, if no other access (read or write) is pending, the data bus keeps its previous value. See [Section 7.1.3.3.9.10](#).

#### **7.1.3.3.10.1.1.5 Asynchronous Multiple (Page) Write Operation on an Address/Data-Multiplexed Device**

Write multiple (page) access in asynchronous mode is not supported for address/data-multiplexed devices. If GPMC\_CONFIG1\_i[28] WRITEMULTIPLE is enabled (1) with GPMC\_CONFIG1\_i[27] WRITETYPE as asynchronous (0), the GPMC processes single asynchronous accesses.

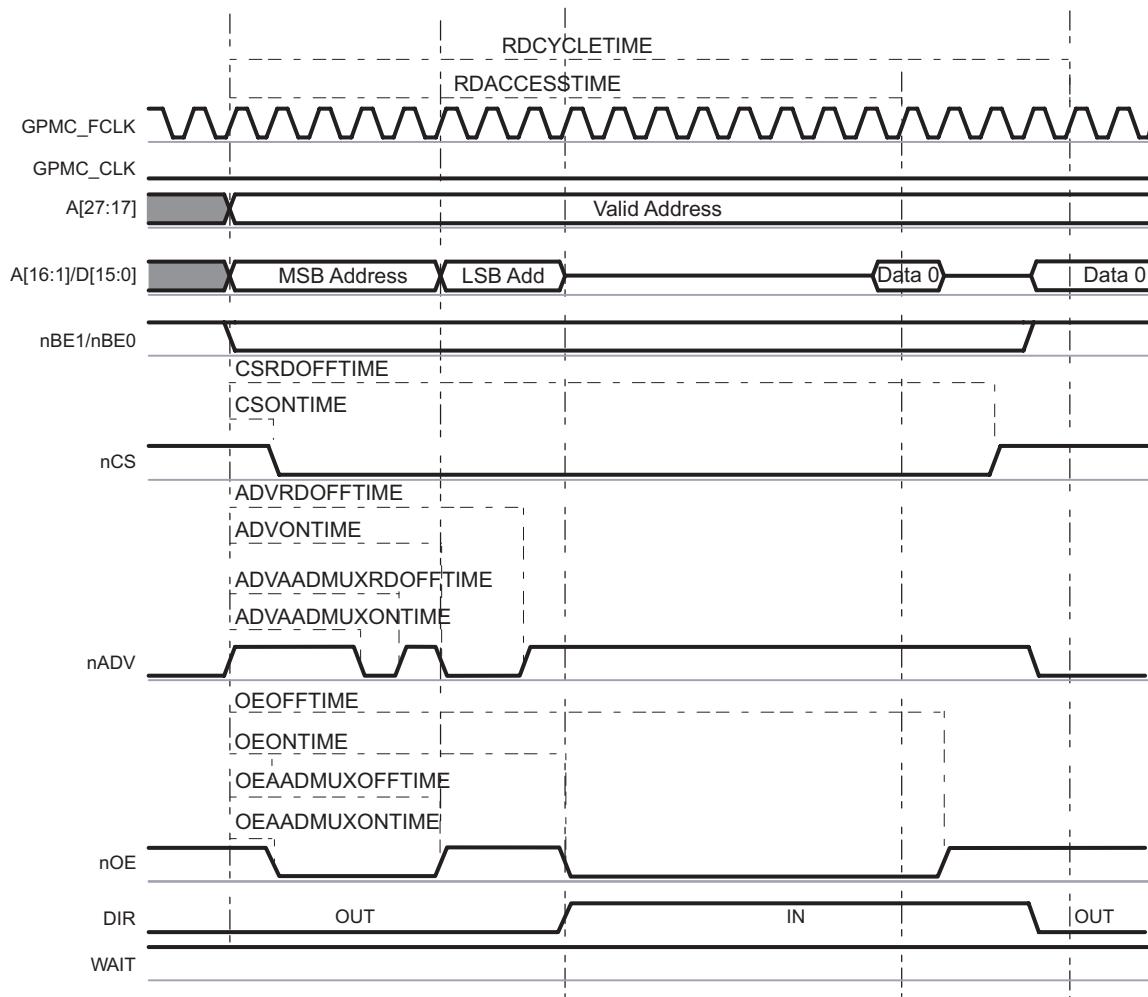
For accesses on non-multiplexed devices, see [Section 7.1.3.3.10.3](#).

#### **7.1.3.3.10.1.2 Access on Address/Address/Data (AAD) Multiplexed Devices**

##### **7.1.3.3.10.1.2.1 Asynchronous Single Read Operation on an AAD-Multiplexed Device**

[Figure 7-15](#) shows an asynchronous single read operation on an AAD-multiplexed device.

**Figure 7-15. Asynchronous Single-Read on an AAD-Multiplexed Device**



### 7.1.3.3.10.1.2.2 Asynchronous Single Read on an AAD-Multiplexed Device

See [Section 7.1.3.9.1](#) for formulas to calculate timing parameters.

[Table 7-41](#) lists the timing bit fields to set up in order to configure the GPMC in asynchronous single write mode.

When the GPMC generates a read access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with OEn driven low. The first address phase ends at the first OEn deassertion time. The second phase for LSB address is qualified with OEn driven high. The second address phase ends at the second OEn assertion time, when the DIR signal goes from OUT to IN.

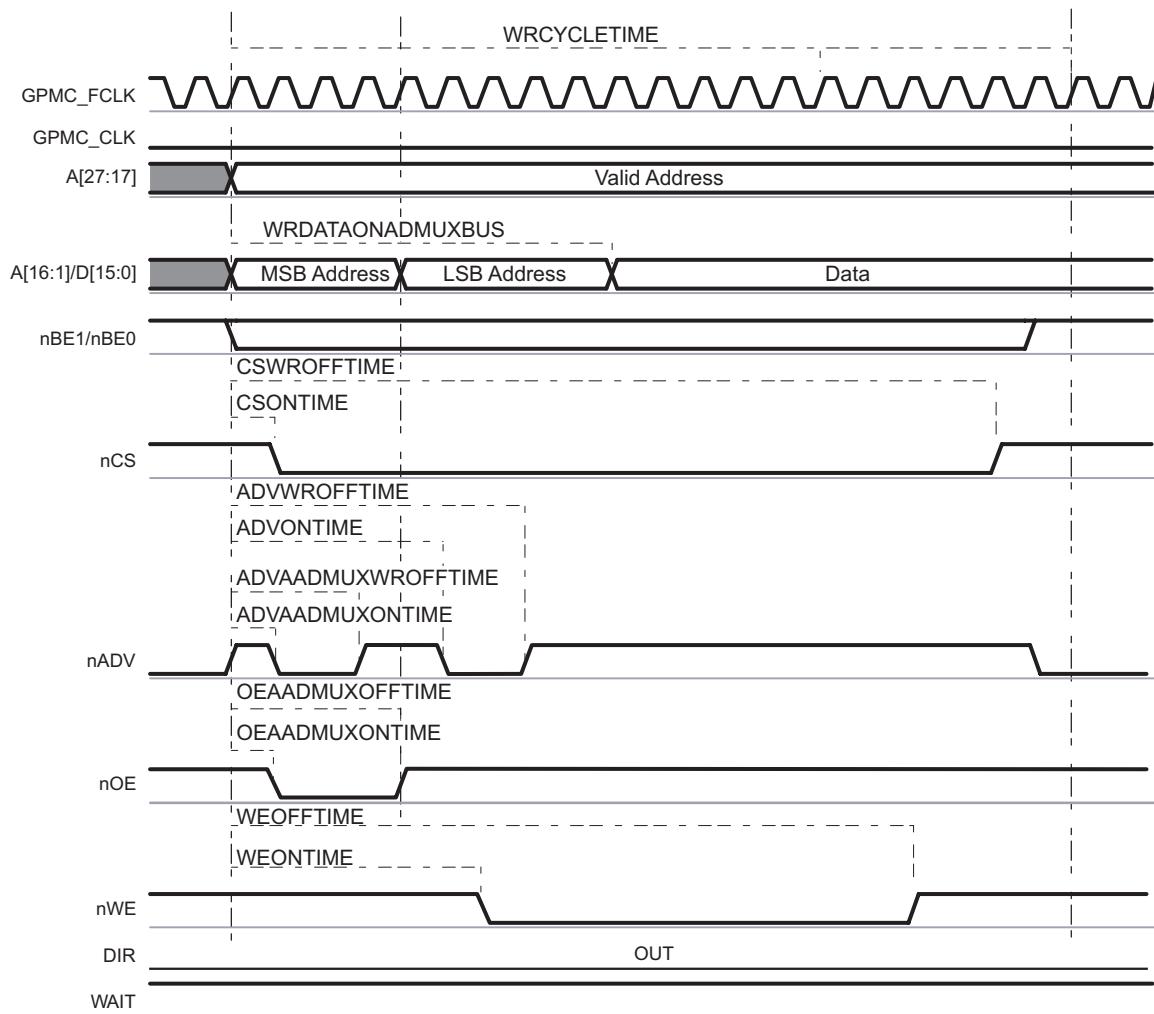
The CSn and DIR signals are controlled in the same way as for asynchronous single read operation on an address/data-multiplexed device.

- Address valid signal ADVn. ADVn is asserted and deasserted twice during a read transaction:
  - ADVn first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME field.
  - ADVn first deassertion time is controlled by the GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME field.
  - ADVn second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADVn second deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME field.
- Output Enable signal OEn. OEn is asserted and deasserted twice during a read transaction (OEn second assertion indicates a read cycle):
  - OEn first assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME field.
  - OEn first deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME field.
  - OEn second assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME field.
  - OEn second deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFETIME field.

### 7.1.3.3.10.1.2.3 Asynchronous Single Write Operation on an AAD-Multiplexed Device

Figure 7-16 shows an asynchronous single write operation on an AAD-multiplexed device.

**Figure 7-16. Asynchronous Single Write on an AAD-Multiplexed Device**



See [Section 7.1.3.9.1](#) for formulas to calculate timing parameters.

[Table 7-41](#) lists the timing bit fields to set up to configure the GPMC in asynchronous single write mode.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with OEn driven low. The second phase for LSB address is qualified with OEn driven high. The address phase ends at WEn assertion time.

The CSn, WEn, and DIR signals are controlled in the same way as for asynchronous single write operation on an address/data-multiplexed device.

- Address valid signal ADVn is asserted and deasserted twice during a write transaction
  - ADVn first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME field.
  - ADVn first deassertion time is controlled by the GPMC\_CONFIG3\_i[30-28] ADVAADMUXWROFFTIME field.
  - ADVn second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADVn second deassertion time is controlled by the GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME field.
- Output Enable signal OEn is asserted during the address phase of a write transaction
  - OEn assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME field.
  - OEn deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME field.

The address bits for the first address phase are driven onto the data bus until OEn deassertion. Data is driven onto the address/data bus at the clock edge defined by the GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS parameter.

#### 7.1.3.3.10.1.2.4 Asynchronous Multiple (Page) Read Operation on an AAD-Multiplexed Device

Write multiple (page) access in asynchronous mode is not supported for AAD-multiplexed devices.

If GPMC\_CONFIG1\_i[28] WRITEMULTIPLE is enabled (1) with GPMC\_CONFIG1\_i[27] WRITETYPE as asynchronous (0), the GPMC processes single asynchronous accesses.

For accesses on non-multiplexed devices, see [Section 7.1.3.3.10.3](#).

#### 7.1.3.3.10.2 Synchronous Access Description

This section details read and write synchronous accesses on address/data multiplexed. All information in this section can be applied to any type of memory - non-multiplexed, address and data multiplexed or AAD-multiplexed - with a difference limited to the address phase. For accesses on non-multiplexed devices, see [Section 7.1.3.3.10.3](#).

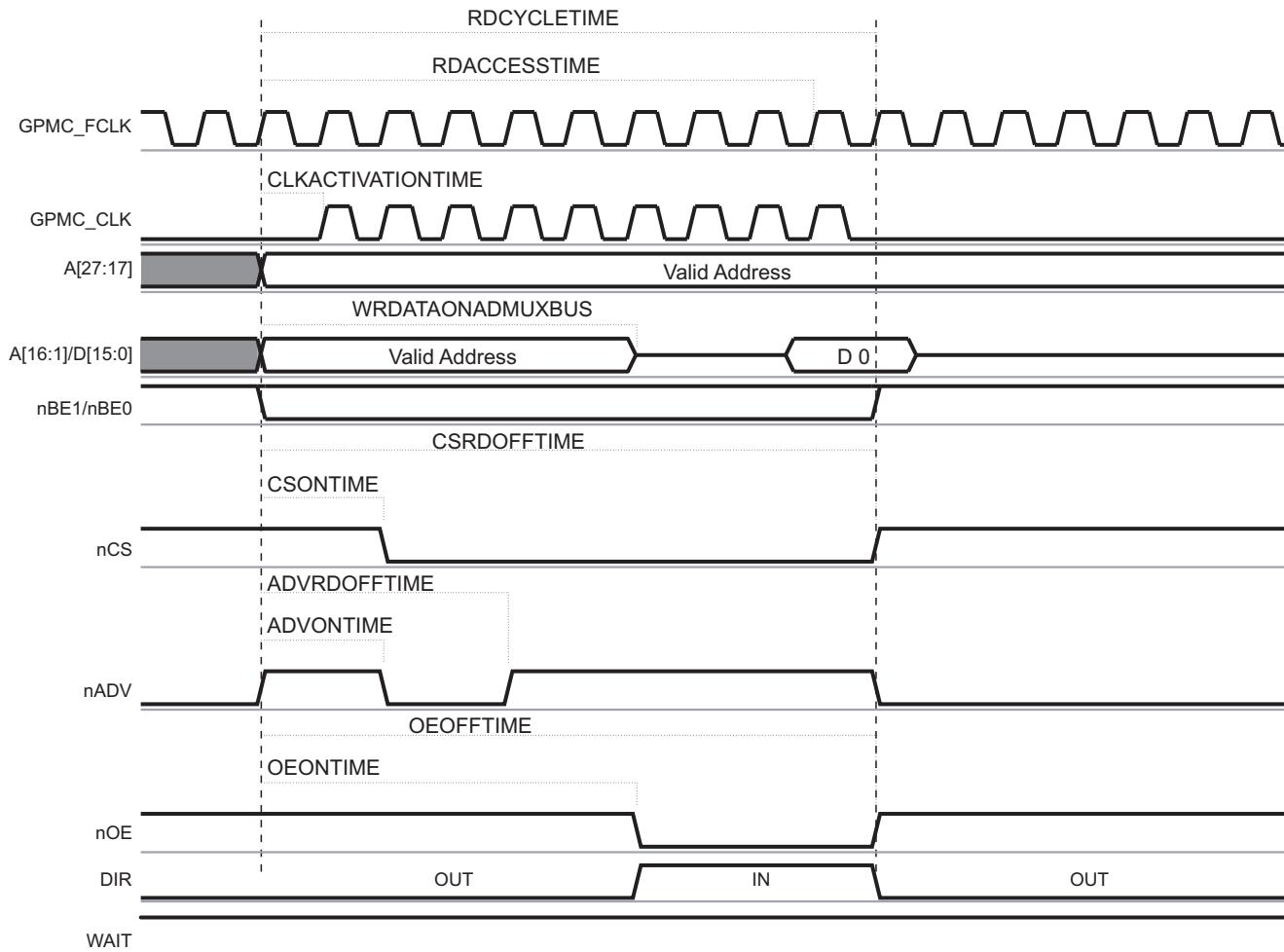
In synchronous operations:

- The GPMC\_CLK clock is provided outside the GPMC when accessing the memory device.
- The GPMC\_CLK clock is derived from the GPMC\_FCLK clock using the GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER field. In the following section, i stands for the chip-select number, i = 0 to 3.
- The GPMC\_CONFIG1\_i[26-25] CLKACTIVATIONTIME field specifies that the GPMC\_CLK is provided outside the GPMC 0, 1, or 2 GPMC\_FCLK cycles after start access time until RDCYCLETIME or WRCYCLETIME completion.

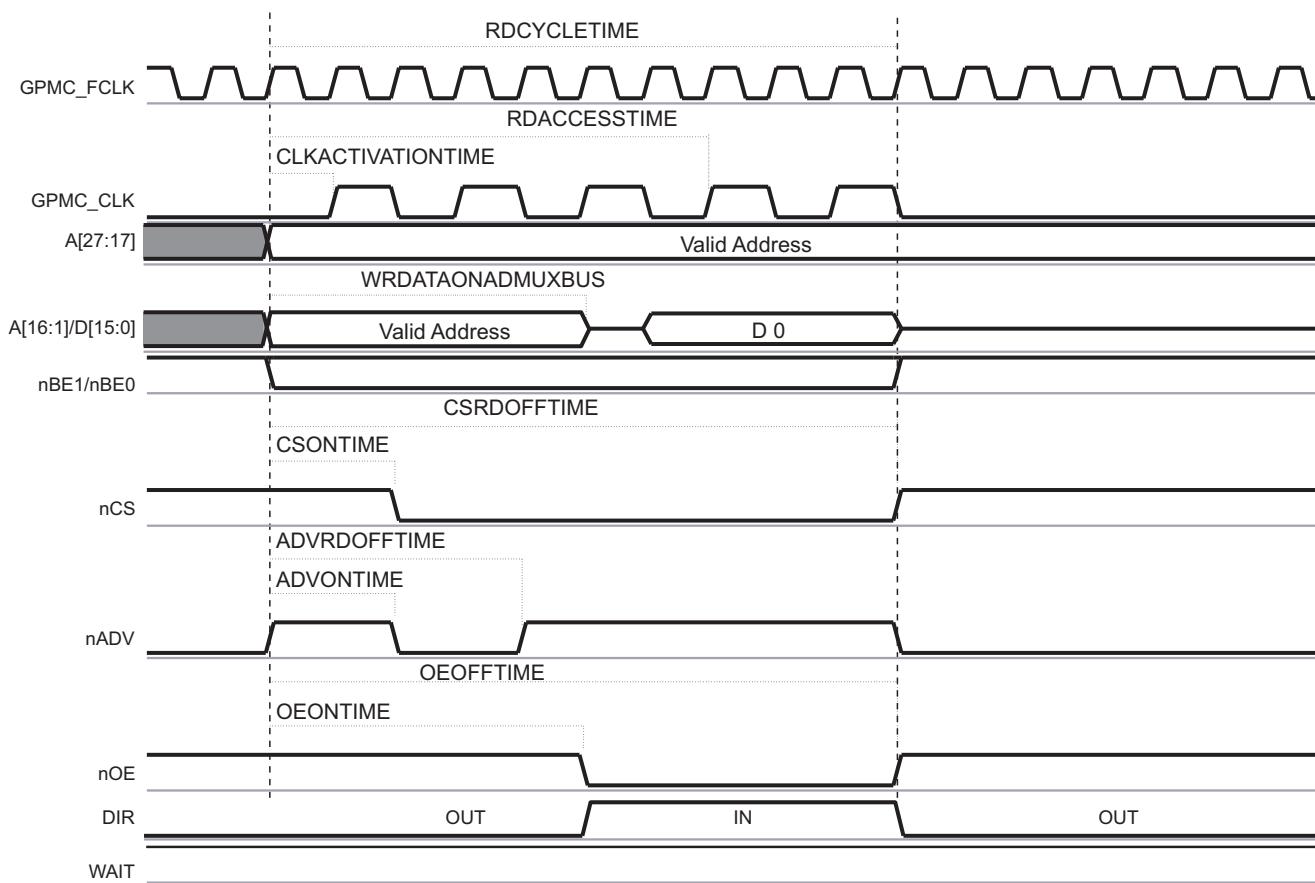
### 7.1.3.3.10.2.1 Synchronous Single Read

Figure 7-17 and Figure 7-18 show a synchronous single-read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.

**Figure 7-17. Synchronous Single Read (GPMCFCLKDIVIDER = 0)**



**Figure 7-18. Synchronous Single Read (GPMCFCLKDIVIDER = 1)**



See [Section 7.1.3.9.1](#) for formulas to calculate timing parameters.

[Table 7-41](#) lists the timing bit fields to set up in order to configure the GPMC in asynchronous single read mode.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until OEn assertion time. For details, see [Section 7.1.3.3.8.2.3](#).

- Chip-select signal CSn
  - CSn assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME field and ensures address setup time to CSn assertion.
  - CSn deassertion time is controlled by the GPMC\_CONFIG2\_i[12-8] CSRDOFFTIME field and ensures address hold time to CSn deassertion.
- Address valid signal ADVn
  - ADVn assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADVn deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME field.
- Output enable signal OEn
  - OEn assertion indicates a read cycle.
  - OEn assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME field.
  - OEn deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFETIME field.
- Initial latency for the first read data is controlled by GPMC\_CONFIG5\_i[20-16] RDACCESSTIME or by monitoring the WAIT signal.
- Total access time (GPMC\_CONFIG5\_i[4-0] RDCYCLETIME) corresponds to RDACCESSTIME plus the address hold time from CSn deassertion, plus time from RDACCESSTIME to CSRDOFFTIME.

- Direction signal DIR: DIR goes from OUT to IN at the same time as OEn assertion.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with OEn driven low. The second phase for LSB address is qualified with OEn driven high. The address phase ends at WEn assertion time.

The CSn and DIR signals are controlled in the same way as for synchronous single read operation on an address/data-multiplexed device.

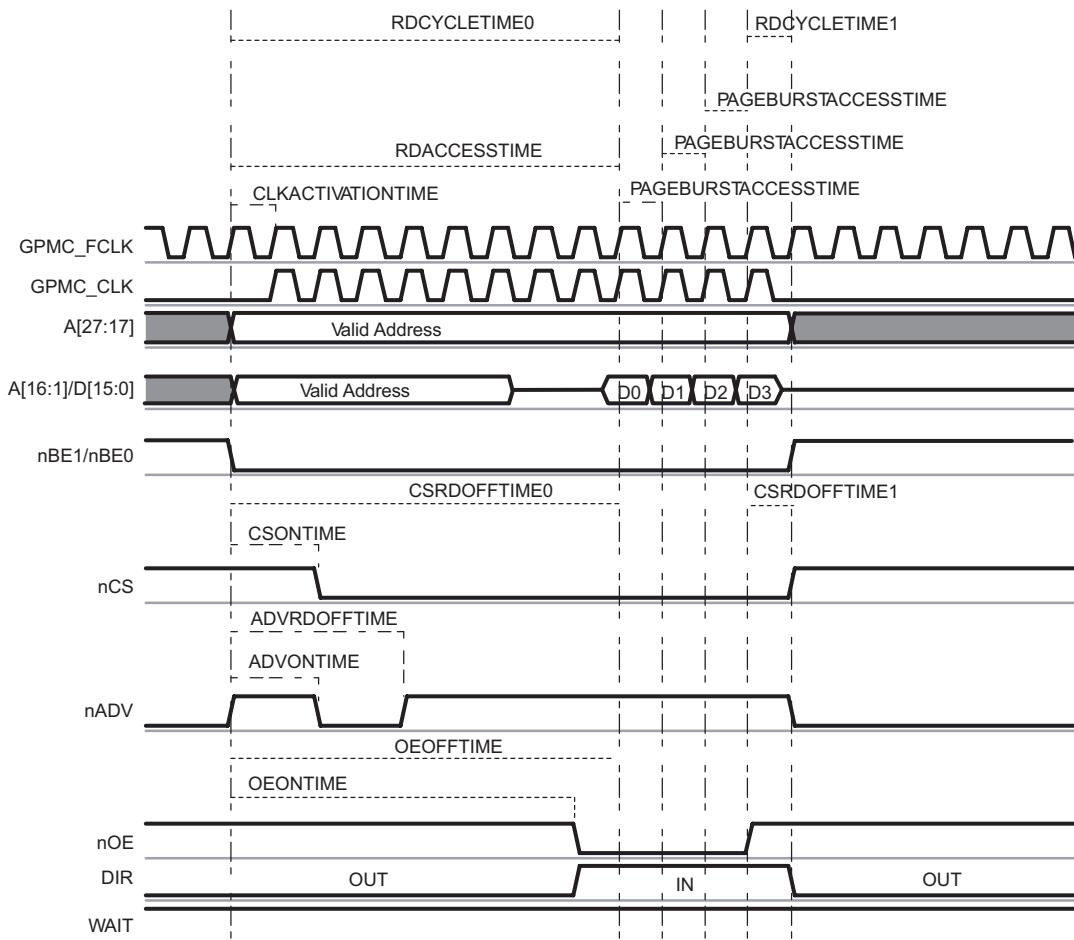
- Address valid signal ADVn is asserted and deasserted twice during a read transaction
  - ADVn first assertion time is controlled by the GPMC\_CONFIG3\_i[6-4] ADVAADMUXONTIME field.
  - ADVn first deassertion time is controlled by the GPMC\_CONFIG3\_i[26-24] ADVAADMUXRDOFFTIME field.
  - ADVn second assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADVn second deassertion time is controlled by the GPMC\_CONFIG3\_i[12-8] ADVRDOFFTIME field.
- Output Enable signal OEn is asserted and deasserted twice during a read transaction (OEn second assertion indicates a read cycle)
  - OEn first assertion time is controlled by the GPMC\_CONFIG4\_i[6-4] OEAADMUXONTIME field.
  - OEn first deassertion time is controlled by the GPMC\_CONFIG3\_i[15-13] OEAADMUXOFFTIME field.
  - OEn second assertion time is controlled by the GPMC\_CONFIG4\_i[3-0] OEONTIME field.
  - OEn second deassertion time is controlled by the GPMC\_CONFIG4\_i[12-8] OEOFETIME field.

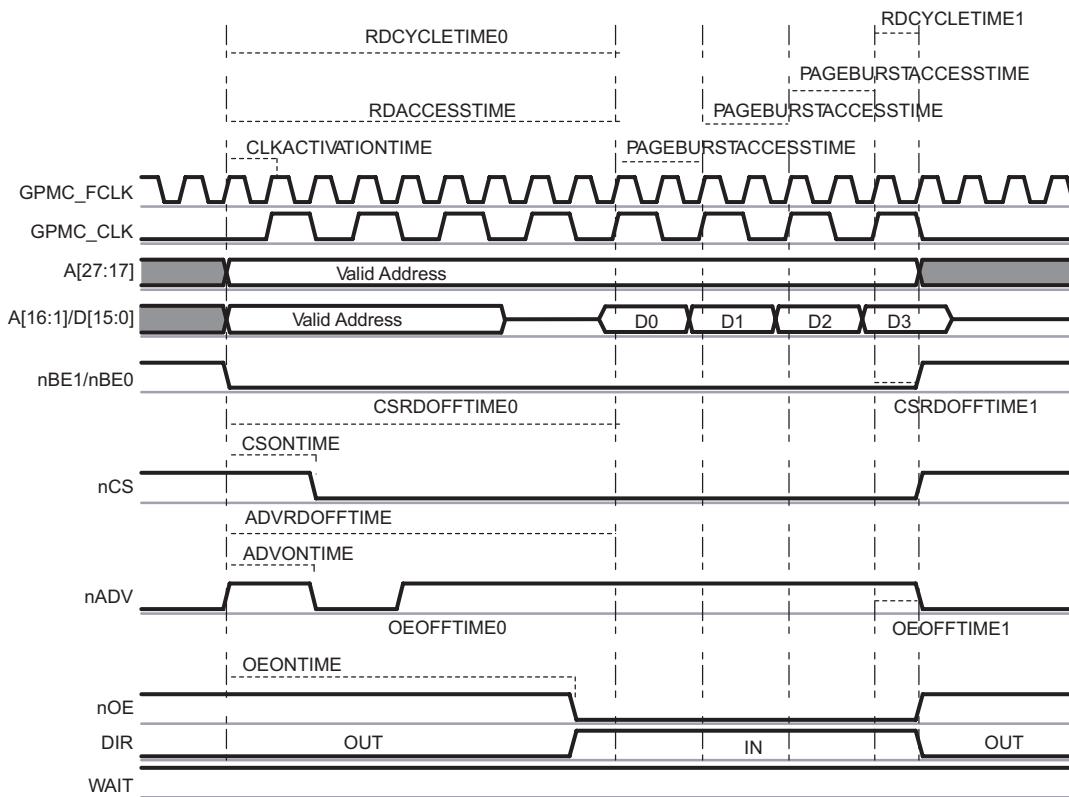
After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See [Section 7.1.3.3.9.10](#).

### 7.1.3.3.10.2.2 Synchronous Multiple (Burst) Read (4-, 8-, 16-Word16 Burst With Wraparound Capability)

Figure 7-19 and Figure 7-20 show a synchronous multiple read operation with GPMCFCLKDIVIDER equal to 0 and 1, respectively.

**Figure 7-19. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 0)**



**Figure 7-20. Synchronous Multiple (Burst) Read (GPMCFCLKDIVIDER = 1)**


When GPMC\_CONFIG5\_i[20-16] RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

The CSn, ADVn, OEn and DIR signals are controlled in the same way as for synchronous single read operation. See [Section 7.1.3.3.10.2.1](#).

Initial latency for the first read data is controlled by RDACCESSTIME or by monitoring the WAIT signal. Successive read data are provided by the memory device each one or two GPMC\_CLK cycles. The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER and the memory-device internal configuration. Depending on the device page length, the GPMC checks device page crossing during a new burst request and purposely insert initial latency (of RDACCESSTIME) when required.

Total access time GPMC\_CONFIG5\_i[4-0] RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time from CSn deassertion. In [Figure 7-19](#), RDCYCLETIME programmed value equals to RDCYCLETIME0 + RDCYCLETIME1.

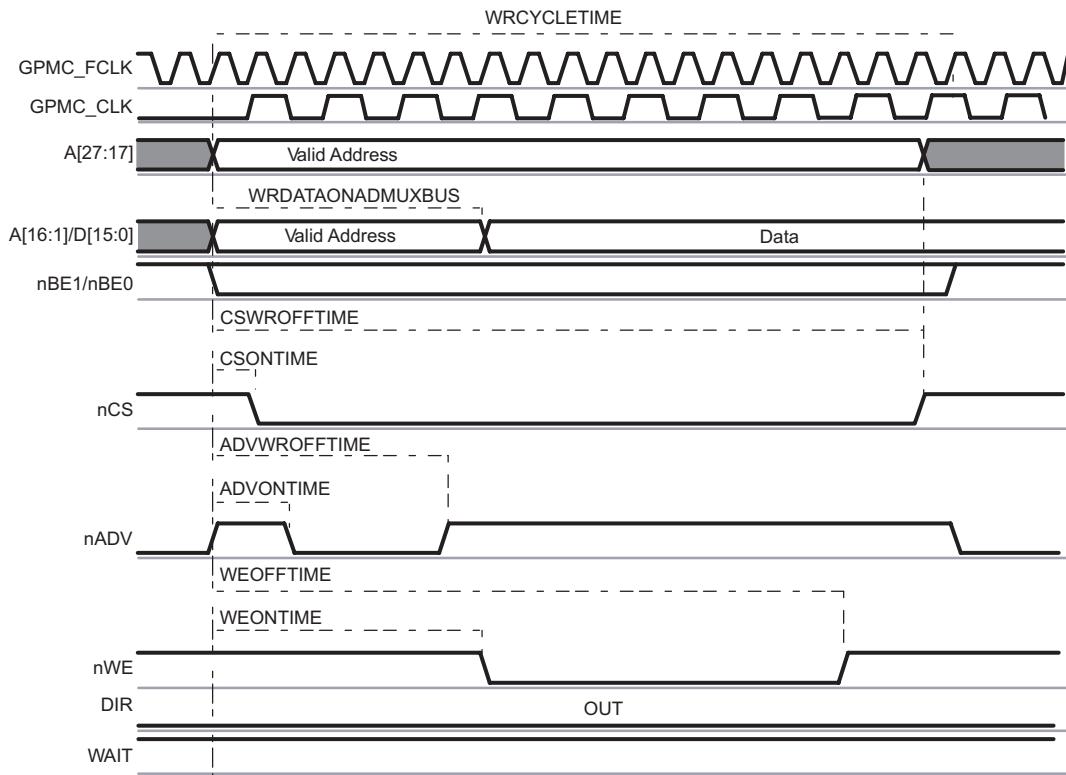
After a read operation, if no other access (read or write) is pending, the data bus is driven with the previous read value. See [Section 7.1.3.3.9.10](#).

Burst wraparound is enabled through the GPMC\_CONFIG1\_i[31] WRAPBURST bit and allows a 4-, 8-, or 16-Word16 linear burst access to wrap within its burst-length boundary through GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH.

### 7.1.3.3.10.2.3 Synchronous Single Write

Burst write mode is used for synchronous single or burst accesses (see [Figure 7-21](#)).

**Figure 7-21. Synchronous Single Write on an Address/Data-Multiplexed Device**

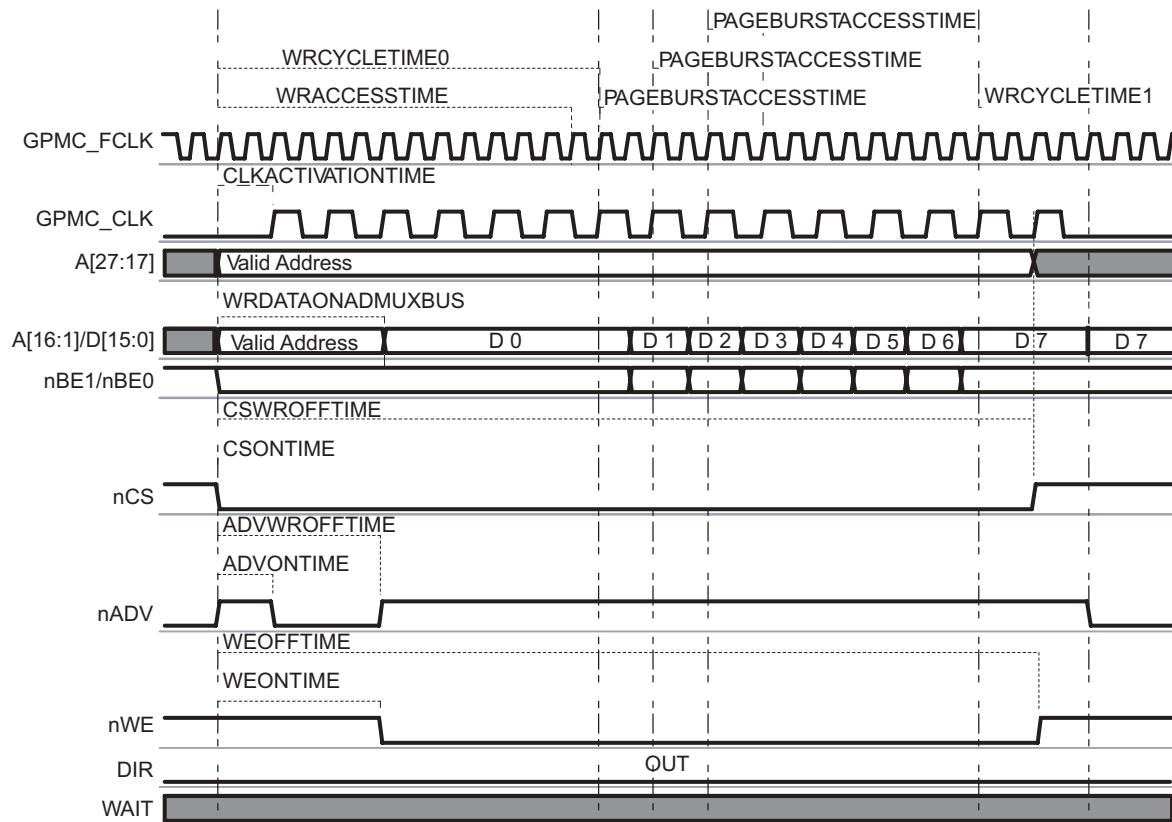


When the GPMC generates a write access to an address/data-multiplexed device, it drives the data bus (with address bits A[16:1]) until [19:16] WRDATAONADMUXBUS time. First data of the burst is driven on the address/data bus at WRDATAONADMUXBUS time.

#### 7.1.3.3.10.2.4 Synchronous Multiple (Burst) Write

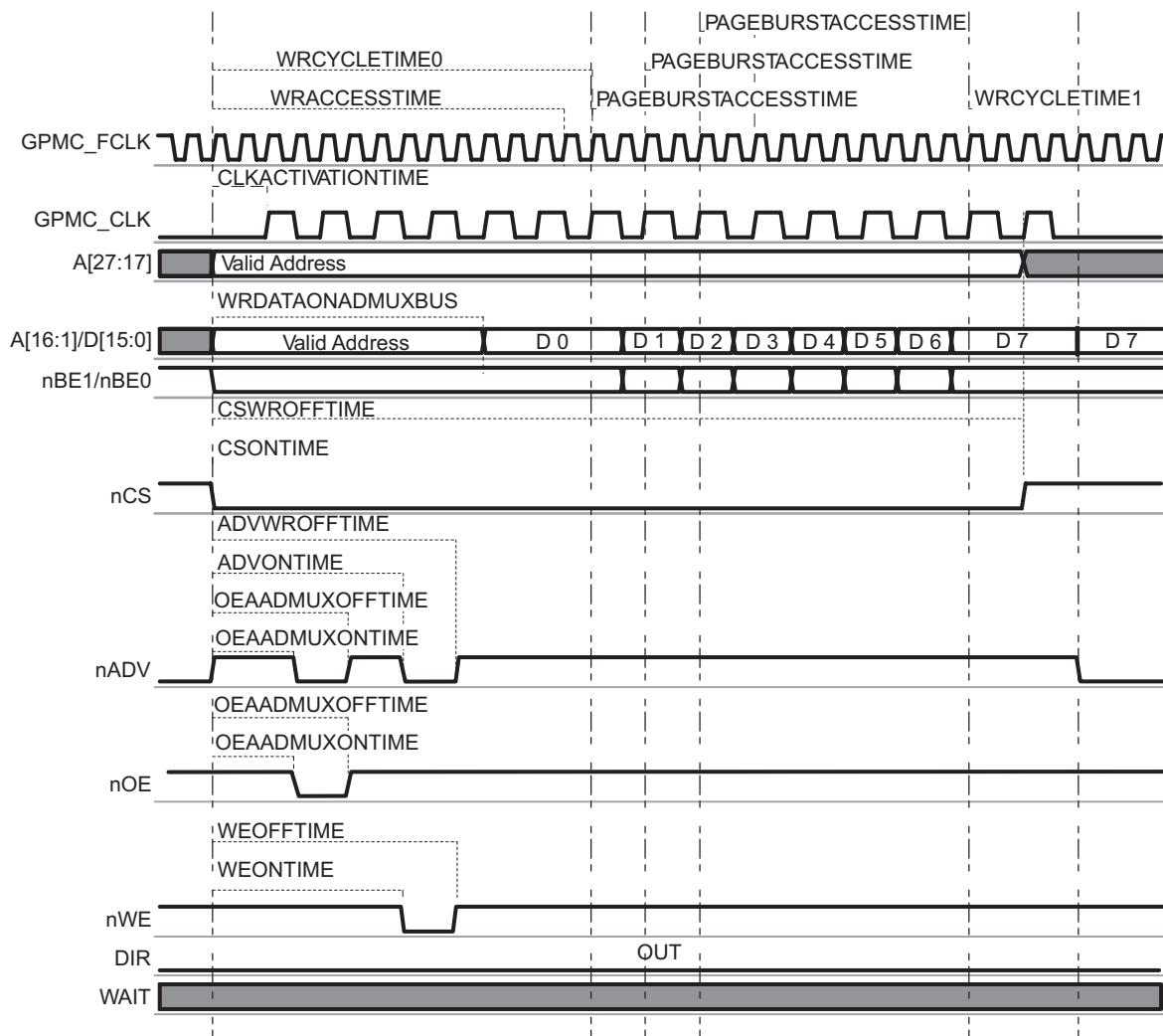
Synchronous burst write mode provides synchronous single or consecutive accesses. [Figure 7-22](#) shows a synchronous burst write access when the chip-select is configured in address/data-multiplexed mode.

**Figure 7-22. Synchronous Multiple Write (Burst Write) in Address/Data-Multiplexed Mode**



[Figure 7-23](#) shows the same synchronous burst write access when the chip-select is configured in address/address/data-multiplexed (AAD-multiplexed) mode.

**Figure 7-23. Synchronous Multiple Write (Burst Write) in Address/Address/Data-Multiplexed Mode**



The first data of the burst is driven on the A/D bus at GPMC\_CONFIG6\_i[19:16] WRDATAONADMUXBUS.

When WRACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to the GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

When the GPMC generates a read access to an address/data-multiplexed device, it drives the address bus until OE<sub>n</sub> assertion time. For details, see [Section 7.1.3.3.8.2.3](#).

- Chip-select signal CS<sub>n</sub>
  - CS<sub>n</sub> assertion time is controlled by the GPMC\_CONFIG2\_i[3-0] CSONTIME field and ensures address setup time to CS<sub>n</sub> assertion.
  - CS<sub>n</sub> deassertion time controlled by the GPMC\_CONFIG2\_i[20-16] CSWROFFTIME field and ensures address hold time to CS<sub>n</sub> deassertion.
- Address valid signal ADV<sub>n</sub>
  - ADV<sub>n</sub> assertion time is controlled by the GPMC\_CONFIG3\_i[3-0] ADVONTIME field.
  - ADV<sub>n</sub> deassertion time is controlled by the GPMC\_CONFIG3\_i[20-16] ADVWROFFTIME field.

- Write enable signal WEn
  - WEn assertion indicates a read cycle.
  - WEn assertion time is controlled by the GPMC\_CONFIG4\_i[19-16] WEONTIME field.
  - WEn deassertion time is controlled by the GPMC\_CONFIG4\_i[28-24] WEOFETIME field.

The WEn falling edge must not be used to control the time when the burst first data is driven in the address/data bus because some new devices require the WEn signal at low during the address phase.

- Direction signal DIR is OUT during the entire access.

When the GPMC generates a write access to an AAD-multiplexed device, all address bits are driven onto the address/data bus in two separate phases. The first phase is used for the MSB address and is qualified with OEn driven low. The second phase for LSB address is qualified with OEn driven high. The address phase ends at WEn assertion time.

The CSn and DIR signals are controlled as detailed above.

- Address valid signal ADVn is asserted and deasserted twice during a read transaction
  - ADVn first assertion time is controlled by the GPMC\_CONFIG3\_i[[6-4] ADVAADMUXONTIME field.
  - ADVn first deassertion time is controlled by the GPMC\_CONFIG3\_i[[26-24] ADVAADMUXRDOFFTIME field.
  - ADVn second assertion time is controlled by the GPMC\_CONFIG3\_i[[3-0] ADVONTIME field.
  - ADVn second deassertion time is controlled by the GPMC\_CONFIG3\_i[[12-8] ADVRDOFFTIME field.
- Output Enable signal OEn is asserted and deasserted twice during a read transaction (OEn second assertion indicates a read cycle)
  - OEn first assertion time is controlled by the GPMC\_CONFIG4\_i[[6-4] OEAADMUXONTIME field.
  - OEn first deassertion time is controlled by the GPMC\_CONFIG4\_i[[15-13] OEAADMUXOFFTIME field.
  - OEn second assertion time is controlled by the GPMC\_CONFIG4\_i[[3-0] OEONTIME field.
  - OEn second deassertion time is controlled by the GPMC\_CONFIG4\_i[[12-8] OEOFETIME field.

First write data is driven by the GPMC at GPMC\_CONFIG6\_i[19-16] WRDATAONADMUXBUS, when in address/data mux configuration. The next write data of the burst is driven on the bus at WRACCESSTIME + 1 during GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME GPMC\_FCLK cycles. The last data of the synchronous burst write is driven until GPMC\_CONFIG5\_i[12-8] WRCYCLETIME completes.

- WRACCESSTIME is defined in the GPMC\_CONFIG6\_i[28-24] register.
- The PAGEBURSTACCESSTIME parameter must be set accordingly with GPMCFCLKDIVIDER and the memory-device internal configuration.

Total access time GPMC\_CONFIG5\_i[12-8] WRCYCLETIME corresponds to WRACCESSTIME plus the address hold time from CSn deassertion. In [Figure 7-23](#) the WRCYCLETIME programmed value equals WRCYCLETIME0 + WRCYCLETIME1. WRCYCLETIME0 and WRCYCLETIME1 delays are not actual parameters and are only a graphical representation of the full WRCYCLETIME value.

After a write operation, if no other access (read or write) is pending, the data bus keeps the previous value. See [Section 7.1.3.3.9.10](#).

#### **7.1.3.3.10.3 Asynchronous and Synchronous Accesses in Nonmultiplexed Mode**

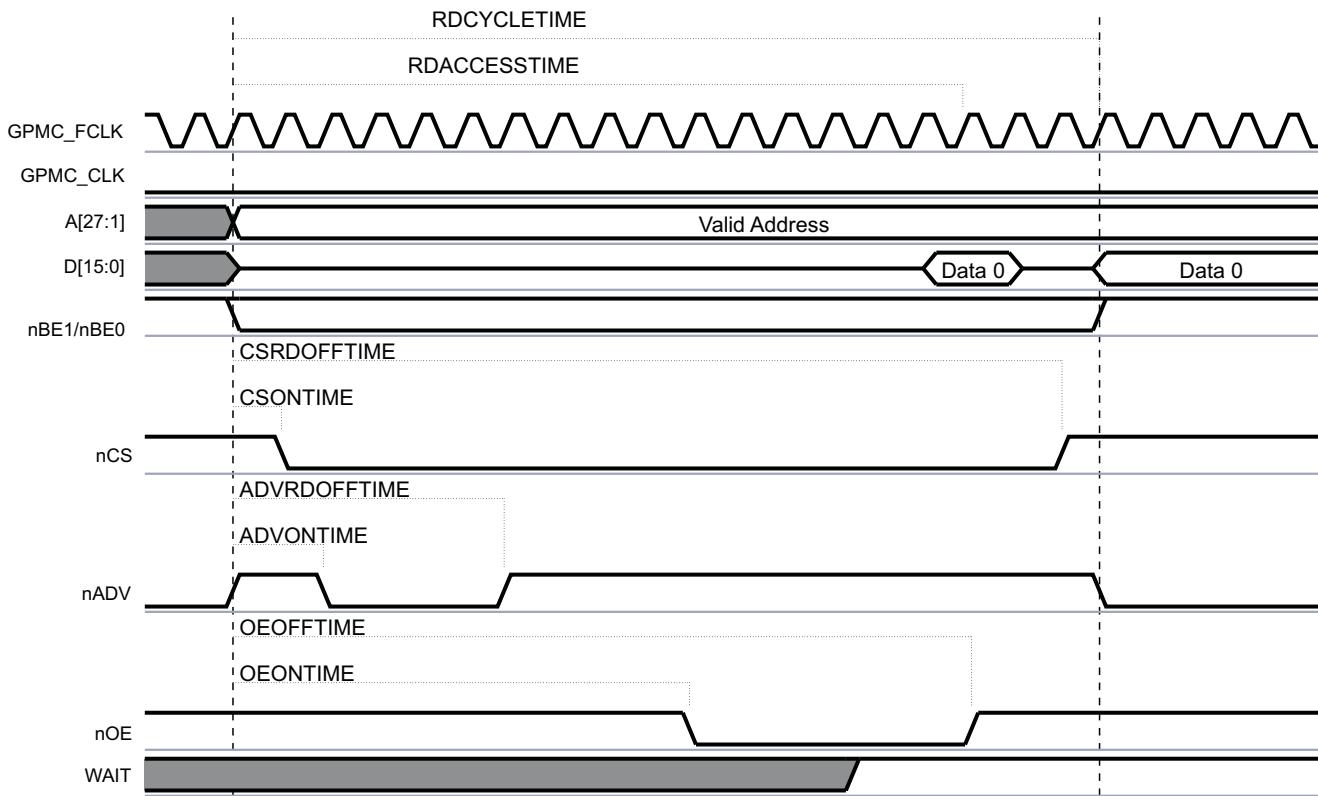
Page mode is only available in non-multiplexed mode.

- Asynchronous single read operation on a nonmultiplexed device
- Asynchronous single write operation on a nonmultiplexed device
- Asynchronous multiple (page mode) read operation on a nonmultiplexed device
- Synchronous operations on a nonmultiplexed device

### 7.1.3.3.10.3.1 Asynchronous Single Read Operation on a Nonmultiplexed Device

Figure 7-24 shows an asynchronous single read operation on a nonmultiplexed device.

**Figure 7-24. Asynchronous Single Read on an Address/Data-Nonmultiplexed Device**



The 27-bit address is driven onto the address bus A[27:1] and the 16-bit data is driven onto the data bus D[15:0].

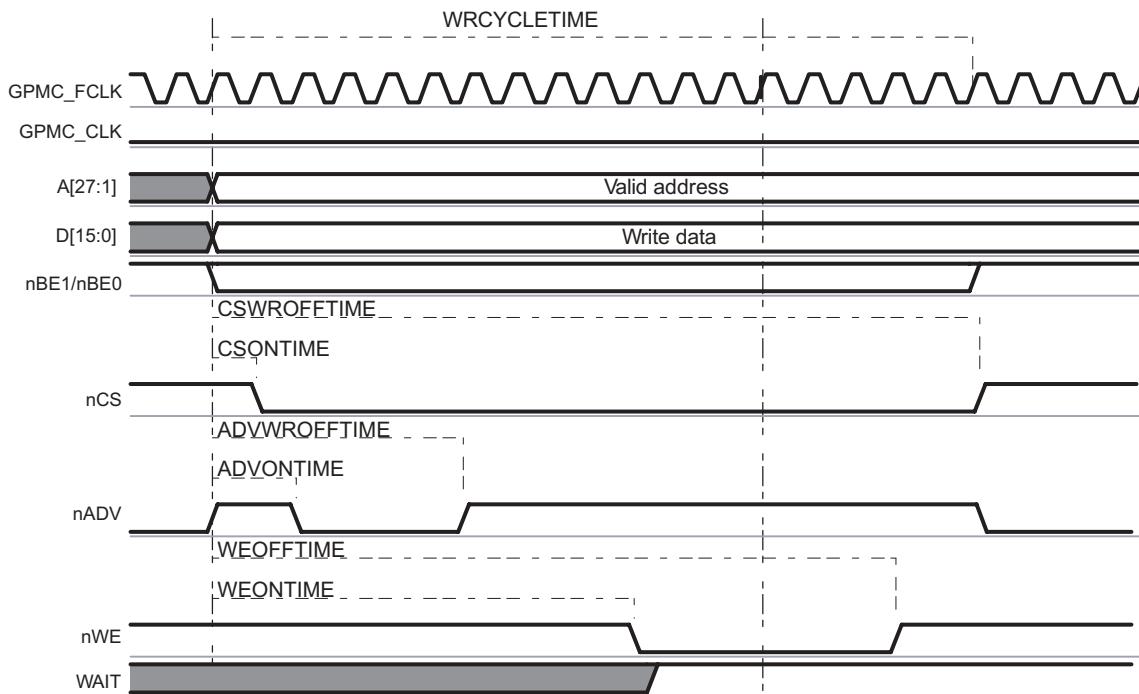
Read data is latched at GPMC\_CONFIG1\_5[20-16] RDACCESSTIME completion time. The end of the access is defined by the GPMC\_CONFIG1\_5[4-0] RDCYCLETIME parameter.

CSn, ADVn, OEn and DIR signals are controlled in the same way as address/data multiplexed accesses, see [Section 7.1.3.3.10.1.1.2](#).

### 7.1.3.3.10.3.2 Asynchronous Single Write Operation on a Nonmultiplexed Device

Figure 7-25 shows an asynchronous single write operation on a nonmultiplexed device.

**Figure 7-25. Asynchronous Single Write on an Address/Data-Nonmultiplexed Device**



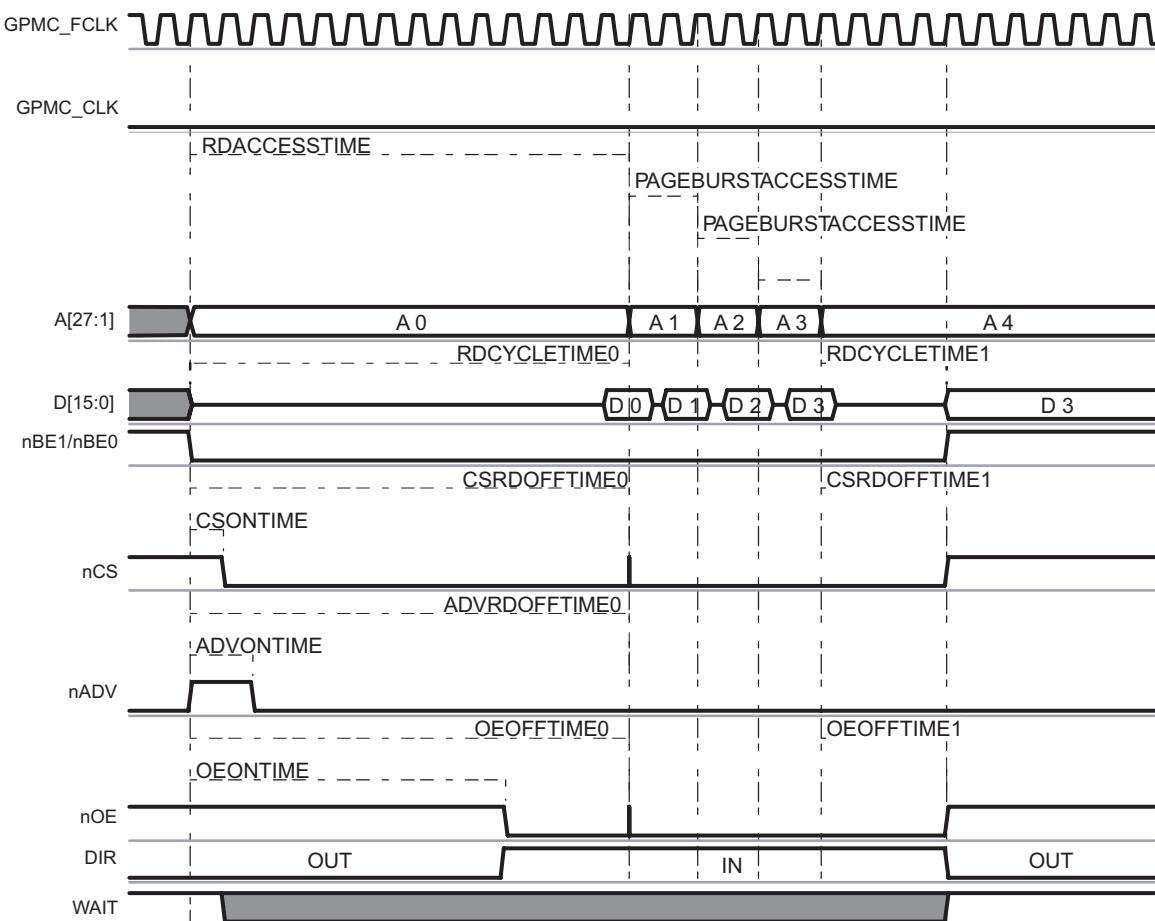
The 27-bit address is driven onto the address bus A[27:1] and the 16-bit data is driven onto the data bus D[15:0].

CSn, ADVn, WEn and DIR signals are controlled in the same way as address/data multiplexed accesses, see [Section 7.1.3.3.10.1.1.3](#).

### 7.1.3.3.10.3.3 Asynchronous Multiple (Page Mode) Read Operation on a Nonmultiplexed Device

Figure 7-26 shows an asynchronous multiple read operation on a Nonmultiplexed Device, in which two word32 host read accesses to the GPMC are split into one multiple (page mode of 4 word16) read access to the attached device.

**Figure 7-26. Asynchronous Multiple (Page Mode) Read**



The WAIT signal is active low.

CSn, ADVn, OEn and DIR signals are controlled in the same way as address/data multiplexed accesses, see [Section 7.1.3.3.10.1.1.2](#).

When RDACCESSTIME completes, control-signal timings are frozen during the multiple data transactions, corresponding to PAGEBURSTACCESSTIME multiplied by the number of remaining data transactions.

Read data is latched at GPMC\_CONFIG5\_i[20-16] RDACCESSTIME completion time. The end of the access is defined by the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME parameter.

During consecutive accesses, the GPMC increments the address after each data read completes.

Delay between successive read data in the page is controlled by the GPMC\_CONFIG5\_i[27-24] PAGEBURSTACCESSTIME parameter. Depending on the device page length, the GPMC can control device page crossing during a burst request and insert initial RDACCESSTIME latency. Note that page crossing is only possible with a new burst access, meaning a new initial access phase is initiated.

Total access time RDCYCLETIME corresponds to RDACCESSTIME plus the address hold time starting from the CSn deassertion.

- The read cycle time is defined in the GPMC\_CONFIG5\_i[4-0] RDCYCLETIME field.
- In [Figure 7-26](#), the RDCYCLETIME programmed value equals RDCYCLETIME0 (before paged accesses) + RDCYCLETIME1 (after paged accesses).

#### **7.1.3.3.10.3.4 Synchronous Operations on a Nonmultiplexed Device**

All information for this section is equivalent to similar operations for address/data- or AAD-multiplexed accesses. The only difference resides in the address phase. See [Section 7.1.3.6](#).

#### **7.1.3.3.10.4 Page and Burst Support**

Each chip-select can be configured to process system single or burst requests into successive single accesses or asynchronous page/synchronous burst accesses, with appropriate access size adaptation.

Depending on the external device page or burst capability, read and write accesses can be independently configured through the GPMC. The GPMC\_CONFIG1\_i[30] READMULTIPLE and GPMC\_CONFIG1\_i[28] WRITMULTIPLE bits are associated with the READTYPE and WRITETYPE parameters.

- Asynchronous write page mode is not supported.
- 8-bit wide device support is limited to nonburstable devices (READMULTIPLE and WRITEMULTIPLE are ignored).
- Not applicable to NAND device interfacing.

#### **7.1.3.3.10.5 System Burst Versus External Device Burst Support**

The device system can issue the following requests to the GPMC:

- Byte, 16-bit word, 32-bit word requests (byte enable controlled). This is always a single request from the interconnect point of view.
- Incrementing fixed-length bursts of two words, four words, and eight words
- Wrapped (critical word access first) fixed-length burst of two, four, or eight words

To process a system request with the optimal protocol, the READMULTIPLE (and READTYPE) and WRITEMULTIPLE (and WRITETYPE) parameters must be set according to the burstable capability (synchronous or asynchronous) of the attached device.

The GPMC access engine issues only fixed-length burst. The maximum length that can be issued is defined per CS by the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH field. When the ATTACHEDDEVICEPAGELENGTH value is less than the system burst request length (including the appropriate access size adaptation according to the device width), the GPMC splits the system burst request into multiple bursts. Within the specified 4-, 8-, or 16-word value, the ATTACHEDDEVICEPAGELENGTH field value must correspond to the maximum-length burst supported by the memory device configured in fixed-length burst mode (as opposed to continuous burst mode).

To get optimal performance from memory devices that natively support 16 Word16-length-wrapping burst capability (critical word access first), the ATTACHEDDEVICEPAGELENGTH parameter must be set to 16 words and the GPMC\_CONFIG1\_i[31] WRAPBURST bit must be set to 1. Similarly DEVICEPAGELENGTH is set to 4 and 8 for memories supporting respectively 4 and 8 Word16-length-wrapping burst.

When the memory device does not offer (or is not configured to offer) native 16 Word16-length-wrapping burst, the WRAPBURST parameter must be cleared, and the GPMC access engine emulates the wrapping burst by issuing the appropriate burst sequences according to the ATTACHEDDEVICEPAGELENGTH value.

When the memory device does not support native-wrapping burst, there is usually no difference in behavior between a fixed burst length mode and a continuous burst mode configuration (except for a potential power increase from a memory-speculative data prefetch in a continuous burst read). However, even though continuous burst mode is compatible with GPMC behavior, because the GPMC access engine issues only fixed-length burst and does not benefit from continuous burst mode, it is best to configure the memory device in fixed-length burst mode.

The memory device maximum-length burst (configured in fixed-length burst wrap or nonwrap mode) usually corresponds to the memory device data buffer size. Memory devices with a minimum of 16 half-word buffers are the most appropriate (especially with wrap support), but memory devices with smaller buffer size (4 or 8) are also supported, assuming that the GPMC\_CONFIG1\_i[24-23] ATTACHEDDEVICEPAGELENGTH field is set accordingly to 4 or 8 words.

The device system issues only requests with addresses or starting addresses for nonwrapping burst requests; that is, the request size boundary is aligned. In case of an eight-word-wrapping burst, the wrapping address always occurs on the eight-words boundary. As a consequence, all words requested must be available from the memory data buffer when the buffer size is equal to or greater than the ATTACHEDDEVICEPAGELENGTH value. This usually means that data can be read from or written to the buffer at a constant rate (number of cycles between data) without wait states between data accesses. If the memory does not behave this way (nonzero wait state burstable memory), wait-pin monitoring must be enabled to dynamically control data-access completion within the burst.

When the system burst request length is less than the ATTACHEDDEVICEPAGELENGTH value, the GPMC proceeds with the required accesses.

#### **7.1.3.3.11 pSRAM Access Specificities**

pSRAM devices are SRAM-pin-compatible low-power memories that contain a self-refreshed DRAM memory array. The GPMC\_CONFIG1\_i[[11-10] DEVICETYPE field shall be cleared to 0b00.

The pSRAM devices uses the NOR protocol. It support the following operations:

- Asynchronous single read
- Asynchronous page read
- Asynchronous single write
- Synchronous single read and write
- Synchronous burst read
- Synchronous burst write (not supported by NOR Flash memory)

pSRAM devices must be powered up and initialized in a predefined manner according to the specifications of the attached device.

pSRAM devices can be programmed to use either mode: fixed or variable latency. pSRAM devices can either automatically schedule autorefresh operations, which force the GPMC to use its WAIT signal capability when read or write operations occur during an internal self-refresh operation, or pSRAM devices automatically include the autorefresh operation in the access time. These devices do not require additional WAIT signal capability or a minimum CSn high pulse width between consecutive accesses to ensure that the correct internal refresh operation is scheduled.

#### **7.1.3.3.12 NAND Access Description**

NAND (8-bit and 16-bit) memory devices using a standard NAND asynchronous address/data-multiplexing scheme can be supported on any chip-select with the appropriate asynchronous configuration settings

As for any other type of memory compatible with the GPMC interface, accesses to a chip-select allocated to a NAND device can be interleaved with accesses to chip-selects allocated to other external devices. This interleaved capability limits the system to *chip enable don't care* NAND devices, because the chip-select allocated to the NAND device must be de-asserted if accesses to other chip-selects are requested.

##### **7.1.3.3.12.1 NAND Memory Device in Byte or 16-Bit Word Stream Mode**

NAND devices require correct command and address programming before data array read or write accesses. The GPMC does not include specific hardware to translate a random address system request into a NAND-specific multiphase access. In that sense, GPMC NAND support, as opposed to random memory-map device support, is data-stream-oriented (byte or 16-bit word).

The GPMC NAND programming model relies on a software driver for address and command formatting with the correct data address pointer value according to the block and page structure. Because of NAND structure and protocol interface diversity, the GPMC does not support automatic command and address phase programming, and software drivers must access the NAND device ID to ensure that correct command and address formatting are used for the identified device.

NAND device data read and write accesses are achieved through an asynchronous read or write access. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Any chip-select region can be qualified as a NAND region to constrain the ADVn\_ALE signal as Address Latch Enable (ALE active high, default state value at low) during address program access, and the BE0n\_CLE signal as Command Latch Enable (CLE active high, default state value at low) during command program access. GPMC address lines are not used (the previous value is not changed) during NAND access.

#### **7.1.3.3.12.1.1 Chip-Select Configuration for NAND Interfacing in Byte or Word Stream Mode**

The GPMC\_CONFIG7\_i register associated with a NAND device region interfaced in byte or word stream mode can be initialized with a minimum size of 16 Mbytes, because any address location in the chip-select memory region can be used to access a NAND data array. The NAND Flash protocol specifies an address sequence where address bits are passed through the data bus in a series of write accesses with the ALE pin asserted. After this address phase, all operations are streamed and the system requests address is irrelevant.

To allow correct command, address, and data-access controls, the GPMC\_CONFIG1\_i register associated with a NAND device region must be initialized in asynchronous read and write modes with the parameters shown in [Table 7-11](#). Failure to comply with these settings corrupts the NAND interface protocol.

The GPMC\_CONFIG1\_i to GPMC\_CONFIG4\_i register associated with a NAND device region must be initialized with the correct control-signal timing value according to the NAND device timing parameters.

**Table 7-11. Chip-Select Configuration for NAND Interfacing**

| Bit Field                | Register       | Value        | Comments   |
|--------------------------|----------------|--------------|--|
| WRAPBURST                | GPMC_CONFIG1_i | 0            | No wrap  |
| READMULTIPLE             | GPMC_CONFIG1_i | 0            | Single access                                    |
| READTYPE                 | GPMC_CONFIG1_i | 0            | Asynchronous mode                                |
| WRITEMULTIPLE            | GPMC_CONFIG1_i | 0            | Single access                                    |
| WRITETYPE                | GPMC_CONFIG1_i | 0            | Asynchronous mode                                |
| CLKACTIVATIONTIME        | GPMC_CONFIG1_i | 0b00         |  |
| ATTACHEDDEVICEPAGELENGTH | GPMC_CONFIG1_i | Don't care   | Single-access mode                               |
| WAITREADMONITORING       | GPMC_CONFIG1_i | 0            | Wait not monitored by GPMC access engine         |
| WAITWRITEMONITORING      | GPMC_CONFIG1_i | 0            | Wait not monitored by GPMC access engine         |
| WAITMONITORINGTIME       | GPMC_CONFIG1_i | Don't care   | Wait not monitored by GPMC access engine         |
| WAITPINSELECT            | GPMC_CONFIG1_i |              | Select which wait is monitored by edge detectors |
| DEVICESIZE               | GPMC_CONFIG1_i | 0b00 or 0b01 | 8- or 16-bit interface                           |
| DEVICETYPE               | GPMC_CONFIG1_i | 0b10         | NAND device in stream mode                       |
| MUXADDDATA               | GPMC_CONFIG1_i | 0b00         | Nonmultiplexed mode                              |
| TIMEPARAGRANULARITY      | GPMC_CONFIG1_i | 0            | Timing achieved with best GPMC clock granularity |
| GPMCFCLKDIVIDER          | GPMC_CONFIG1_i | Don't care   | Asynchronous mode                                |

#### **7.1.3.3.12.1.2 NAND Device Command and Address Phase Control**

NAND devices require multiple address programming phases. The MPU software driver is responsible for issuing the correct number of command and address program accesses, according to the device command set and the device address-mapping scheme.

NAND device-command and address-phase programming is achieved through write requests to the GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i register locations with the correct command and address values. These locations are mapped in the associated chip-select register region. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

Command and address values are not latched during the access and cannot be read back at the register location.

- Only write accesses must be issued to these locations, but the GPMC does not discard any read access. Accessing a NAND device with OEn and CLE or ALE asserted (read access) can produce undefined results.
- Write accesses to the GPMC\_NAND\_COMMAND\_i register location and to the GPMC\_NAND\_ADDRESS\_i register location must be posted for faster operations. The GPMC\_CONFIG[0] NANDFORCEPOSTEDWRITE bit enables write accesses to these locations as posted, even if they are defined as nonposted.

A write buffer is used to store write transaction information before the external device is accessed:

- Up to eight consecutive posted write accesses can be accepted and stored in the write buffer.
- For nonposted write, the pipeline is one deep.
- A GPMC\_STATUS[0] EMPTYWRITEBUFFERSTATUS bit stores the empty status of the write buffer.

The GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i registers are 32-bit word locations, which means any 32-bit word or 16-bit word access is split into 4- or 2-byte accesses if an 8-bit wide NAND device is attached. For multiple-command phase or multiple-address phase, the software driver can use 32-bit word or 16-bit word access to these registers, but it must account for the splitting and little-endian ordering scheme. When only one byte command or address phase is required, only byte write access to a GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i can be used, and any of the four byte locations of the registers are valid.

The same applies to GPMC\_NAND\_COMMAND\_i and GPMC\_NAND\_ADDRESS\_i 32-bit word write access to a 16-bit wide NAND device (split into two 16-bit word accesses). In the case of a 16-bit word write access, the MSByte of the 16-bit word value must be set according to the NAND device requirement (usually 0). Either 16-bit word location or any one of the four byte locations of the registers is valid

### 7.1.3.3.12.1.3 Command Latch Cycle

Writing data at the GPMC\_NAND\_COMMAND\_i location places the data as the NAND command value on the bus, using a regular asynchronous write access.

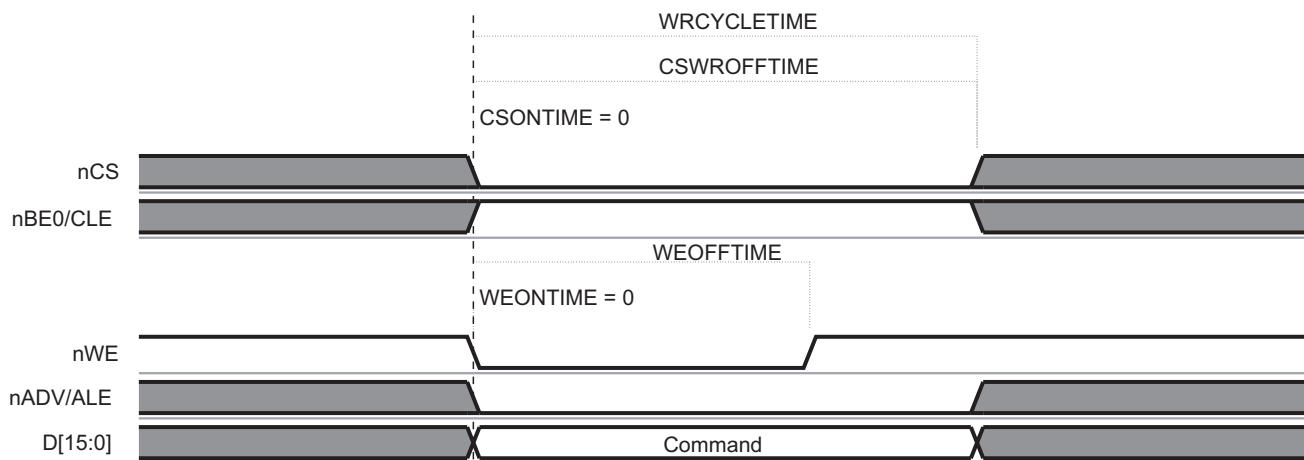
- CSn[i] is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- CLE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- WE is controlled by the WEONTIME and WEOFETIME timing parameters.
- ALE and REn (OEn) are maintained inactive.

[Figure 7-27](#) shows the NAND command latch cycle.

CLE is shared with the BE0n output signal and has an inverted polarity from BE0n. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, BE0n (also BE1n) must not toggle, because it is shared with CLE.

NAND Flash memories do not use byte enable signals at all.

**Figure 7-27. NAND Command Latch Cycle**



#### 7.1.3.3.12.1.4 Address Latch Cycle

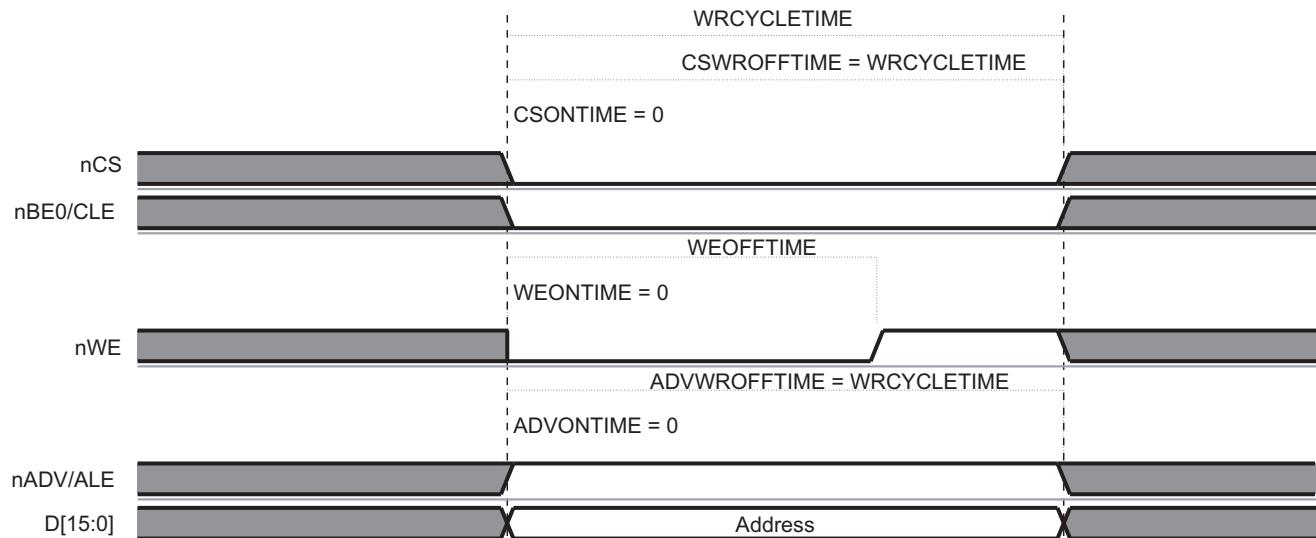
Writing data at the GPMC\_NAND\_ADDRESS\_i location places the data as the NAND partial address value on the bus, using a regular asynchronous write access.

- CSn is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- ALE is controlled by the ADVONTIME and ADVWROFFTIME timing parameters.
- WEn is controlled by the WEONTIME and WEOFETIME timing parameters.
- CLE and REn (OEn) are maintained inactive.

Figure 7-28 shows the NAND address latch cycle.

ALE is shared with the ADVn output signal and has an inverted polarity from ADVn. The NAND qualifier deals with this. During the asynchronous NAND data access cycle, ALE is kept stable.

**Figure 7-28. NAND Address Latch Cycle**



### 7.1.3.3.12.1.5 NAND Device Data Read and Write Phase Control in Stream Mode

NAND device data read and write accesses are achieved through a read or write request to the chip-select-associated memory region at any address location in the region or through a read or write request to the GPMC\_NAND\_DATA\_i location mapped in the chip-select-associated control register region.

GPMC\_NAND\_DATA\_i is not a true register, but an address location to enable REn or WEn signal control. The associated chip-select signal timing control must be programmed according to the NAND device timing specification.

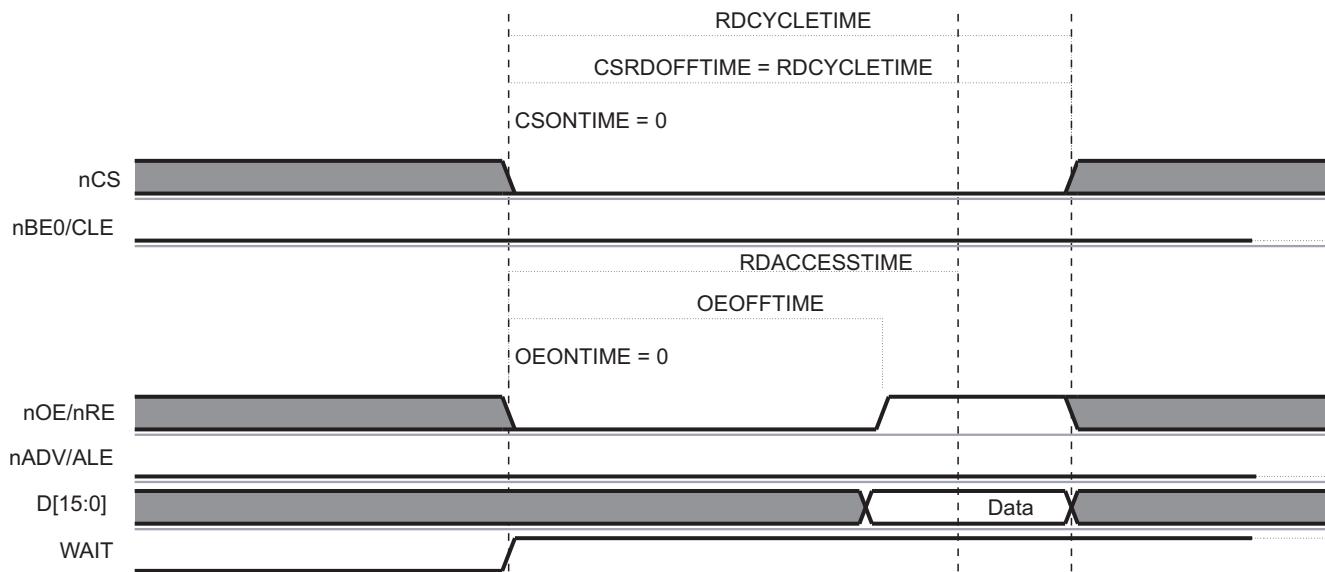
Reading data from the GPMC\_NAND\_DATA\_i location or from any location in the associated chip-select memory region activates an asynchronous read access.

- CSn is controlled by the CSONTIME and CSRDOFFTIME timing parameters.
- REn is controlled by the OEONTIME and OEOFETIME timing parameters.
- To take advantage of REn high-to-data invalid minimum timing value, the RDACCESSTIME can be set so that data are effectively captured after REn deassertion. This allows optimization of NAND read access cycle time completion. For optimal timing parameter settings, see the NAND device and the device IC timing parameters.

ALE, CLE, and WEn are maintained inactive.

Figure 7-29 shows the NAND data read cycle.

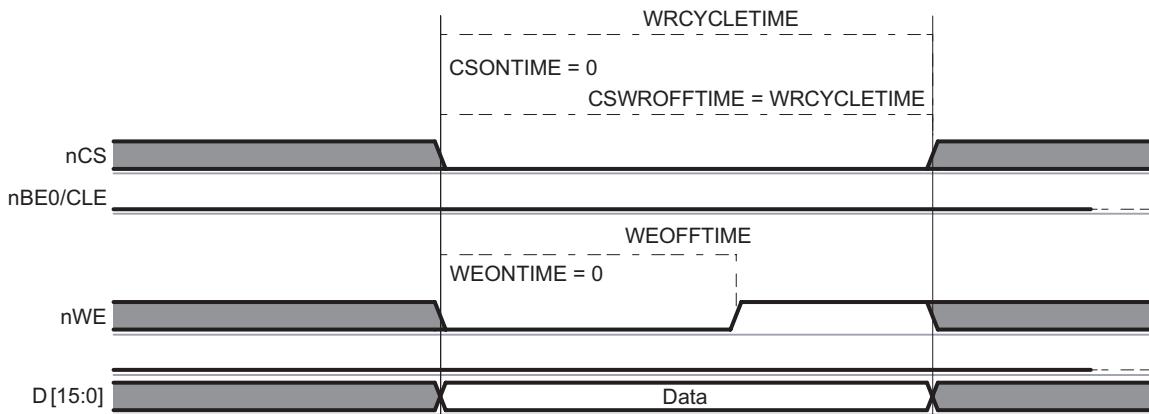
**Figure 7-29. NAND Data Read Cycle**



Writing data to the GPMC\_NAND\_DATA\_i location or to any location in the associated chip-select memory region activates an asynchronous write access.

- CSn is controlled by the CSONTIME and CSWROFFTIME timing parameters.
- WEn is controlled by the WEONTIME and WEOFETIME timing parameters.
- ALE, CLE, and REn (OEn) are maintained inactive.

Figure 7-30 shows the NAND data write cycle.

**Figure 7-30. NAND Data Write Cycle**


#### 7.1.3.3.12.1.6 NAND Device General Chip-Select Timing Control Requirement

For most NAND devices, read data access time is dominated by CSn-to-data-valid timing and has faster REn-to-data-valid timing. Successive accesses with CSn deassertions between accesses are affected by this timing constraint. Because accesses to a NAND device can be interleaved with other chip-select accesses, there is no certainty that CSn always stays low between two accesses to the same chip-select. Moreover, an CSn deassertion time between the same chip-select NAND accesses is likely to be required as follows: the CSn deassertion requires programming CYCLETIME and RDACCESSTIME according to the CSn-to-data-valid critical timing.

To get full performance from NAND read and write accesses, the prefetch engine can dynamically reduce RDCYCLETIME, WRCYCLETIME, RDACCESSTIME, WRACCESSTIME, CSRDOFFTIME, CSWROFFTIME, ADVRDOFFTIME, ADVWROFFTIME, OEOFETIME, and WEOFFTIME on back-to-back NAND accesses (to the same memory) and suppress the minimum CSn high pulse width between accesses. For more information about optimal prefetch engine access, see [Section 7.1.3.3.12.4](#).

Some NAND devices require minimum write-to-read idle time, especially for device-status read accesses following status-read command programming (write access). If such write-to-read transactions are used, a minimum CSn high pulse width must be set. For this, CYCLE2CYCLESAMECSEN and CYCLE2CYCLEDELAY must be set according to the appropriate timing requirement to prevent any timing violation.

NAND devices usually have an important REn high to data bus in tristate mode. This requires a bus turnaround setting (BUSTURNAROUND = 1), so that the next access to a different chip-select is delayed until the BUSTURNAROUND delay completes. Back-to-back NAND read accesses to the same NAND Flash are not affected by the programmed bus turnaround delay.

#### 7.1.3.3.12.1.7 Read and Write Access Size Adaptation

##### 7.1.3.3.12.1.7.1 8-Bit Wide NAND Device

Host 16-bit word and 32-bit word read and write access requests to a chip-select associated with an 8-bit wide NAND device are split into successive read and write byte accesses to the NAND memory device. Byte access is ordered according to little-endian organization. A NAND 8-bit wide device must be interfaced on the D0D7 interface bus lane. GPMC data accesses are justified on this bus lane when the chip-select is associated with an 8-bit wide NAND device.

##### 7.1.3.3.12.1.7.2 16-Bit Wide NAND Device

Host 32-bit word read and write access requests to a chip-select associated with a 16-bit wide NAND device are split into successive read and write 16-bit word accesses to the NAND memory device. 16-bit word access is ordered according to little-endian organization.

Host byte read and write access requests to a 16-bit wide NAND device are completed as 16-bit accesses on the device itself, because there is no byte-addressing capability on 16-bit wide NAND devices. This means that the NAND device address pointer is incremented on a 16-bit word basis and not on a byte basis. For a read access, only the requested byte is given back to the host, but the remaining byte is not stored or saved by the GPMC, and the next byte or 16-bit word read access gets the next 16-bit word NAND location. For a write access, the invalid byte part of the 16-bit word is driven to FF, and the next byte or 16-bit word write access programs the next 16-bit word NAND location.

Generally, byte access to a 16-bit wide NAND device should be avoided, especially when ECC calculation is enabled. 8-bit or 16-bit ECC-based computations are corrupted by a byte read to a 16-bit wide NAND device, because the nonrequested byte is considered invalid on a read access (not captured on the external data bus; FF is fed to the ECC engine) and is set to FF on a write access.

Host requests (read/write) issued in the chip-select memory region are translated in successive single or split accesses (read/write) to the attached device. Therefore, incrementing 32-bit burst requests are translated in multiple 32-bit sequential accesses following the access adaptation of the 32-bit to 8- or 16-bit device.

#### **7.1.3.12.2 NAND Device-Ready Pin**

The NAND memory device provides a ready pin to indicate data availability after a block/page opening and to indicate that data programming is complete. The ready pin can be connected to one of the WAIT GPMC input pins; data read accesses must not be tried when the ready pin is sampled inactive (device is not ready) even if the associated chip-select WAITREADMONITORING bit field is set. The duration of the NAND device busy state after the block/page opening is so long (up to 50  $\mu$ s) that accesses occurring when the ready pin is sampled inactive can stall GPMC access and eventually cause a system time-out.

If a read access to a NAND flash is done using the wait monitoring mode, the device is blocked during a page opening, and so is the GPMC. If the correct settings are used, other chip-selects can be used while the memory processes the page opening command.

To avoid a time-out caused by a block/page opening delay in NAND flash, disable the wait pin monitoring for read and write accesses (that is, set the GPMC\_CONFIG1\_i[21] WAITWRITEMONITORING and GPMC\_CONFIG1\_i[22] WAITREADMONITORING bits to 0 and use one of the following methods instead:

- Use software to poll the WAITnSTATUS bit ( $n = 0$  to 1) of the GPMC\_STATUS register.
- Configure an interrupt that is generated on the WAIT signal change (through the GPMC\_IRQENABLE [11-8]bits).

Even if the READWAITMONITORING bit is not set, the external memory nR/B pin status is captured in the programmed WAIT bit in the GPMC\_STATUS register.

The READWAITMONITORING bit method must be used for other memories than NAND flash, if they require the use of a WAIT signal.

#### **7.1.3.12.2.1 Ready Pin Monitored by Software Polling**

The ready signal state can be monitored through the GPMC\_STATUS WAITxSTATUS bit ( $x = 0$  or 1). The software must monitor the ready pin only when the signal is declared valid. Refer to the NAND device timing parameters to set the correct software temporization to monitor ready only after the invalid window is complete from the last read command written to the NAND device.

#### **7.1.3.12.2.2 Ready Pin Monitored by Hardware Interrupt**

Each gpmc\_wait input pin can generate an interrupt when a wait-to-no-wait transition is detected. Depending on whether the GPMC\_CONFIG WAITxPINPOLARITY bits ( $x = 0$  or 1) is active low or active high, the wait-to-no-wait transition is a low-to-high external WAIT signal transition or a high-to-low external WAIT signal transition, respectively.

The wait transition pin detector must be cleared before any transition detection. This is done by writing 1 to the WAITxEDGEDETECTIONSTATUS bit ( $x = 0$  or  $1$ ) of the GPMC\_IRQSTATUS register according to the gpmc\_wait pin used for the NAND device-ready signal monitoring. To detect a wait-to-no-wait transition, the transition detector requires a wait active time detection of a minimum of two GPMC\_FCLK cycles. Software must incorporate precautions to clear the wait transition pin detector before wait (busy) time completes.

A wait-to-no-wait transition detection can issue a GPMC interrupt if the WAITxEDGEDETECTIONENABLE bit in the GPMC\_IRQENABLE register is set and if the WAITxEDGEDETECTIONSTATUS bit field in the GPMC\_IRQSTATUS register is set.

The WAITMONITORINGTIME field does not affect wait-to-no-wait transition time detection.

It is also possible to poll the WAITxEDGEDETECTIONSTATUS bit field in the GPMC\_IRQSTATUS register according to the gpmc\_wait pin used for the NAND device ready signal monitoring.

#### **7.1.3.3.12.3 ECC Calculator**

The General Purpose Memory Controller includes an Error Code Correction (ECC) calculator circuitry that enables on the fly ECC calculation during data read or data program (that is, write) operations. The page size supported by the ECC calculator in one calculation/context is 512 bytes.

The user can choose from two different algorithms with different error correction capabilities through the GPMC\_ECC\_CONFIG[16] ECCALGORITHM bit:

- Hamming code for 1-bit error code correction on 8- or 16-bit NAND Flash organized with page size greater than 512 bytes
- BCH (Bose-Chaudhuri-Hocquenghem) code for 4- to 16-bit error correction

The GPMC does not directly handle the error code correction itself. During writes, the GPMC computes parity bits. During reads, the GPMC provides enough information for the processor to correct errors without reading the data buffer all over again.

The Hamming code ECC is based on a 2-dimensional (row and column) bit parity accumulation. This parity accumulation is either accomplished on the programmed number of bytes or 16-bit words read from the memory device, or written to the memory device in stream mode.

Because the ECC engine includes only one accumulation context, it can be allocated to only one chip-select at a time through the GPMC\_ECC\_CONFIG[3-1] ECCCS bit field. Even if two CS use different ECC algorithms, one the Hamming code and the other a BCH code, they must define separate ECC contexts because some of the ECC registers are common to all types of algorithms.

#### **7.1.3.3.12.3.1 Hamming Code**

All references to Error Code Correction (ECC) in this subsection refer to the 1-bit error correction Hamming code.

The ECC is based on a two-dimensional (row and column) bit parity accumulation known as Hamming Code. The parity accumulation is done for a programmed number of bytes or 16-bit word read from the memory device or written to the memory device in stream mode.

There is no automatic error detection or correction, and it is the software NAND driver responsibility to read the multiple ECC calculation results, compare them to the expected code value, and take the appropriate corrective actions according to the error handling strategy (ECC storage in spare byte, error correction on read, block invalidation).

The ECC engine includes a single accumulation context. It can be allocated to a single designated chip-select at a time and parallel computations on different chip-selects are not possible. Since it is allocated to a single chip-select, the ECC computation is not affected by interleaved GPMC accesses to other chip-selects and devices. The ECC accumulation is sequentially processed in the order of data read from or written to the memory on the designated chip-select. The ECC engine does not differentiate read accesses from write accesses and does not differentiate data from command or status information. It is the software responsibility to make sure only relevant data are passed to the NAND flash memory while the ECC computation engine is active.

The starting NAND page location must be programmed first, followed by an ECC accumulation context reset with an ECC enabling, if required. The NAND device accesses discussed in the following sections must be limited to data read or write until the specified number of ECC calculations is completed.

#### **7.1.3.3.12.3.1.1 ECC Result Register and ECC Computation Accumulation Size**

The GPMC includes up to nine ECC result registers (GPMC\_ECCj\_RESULT, j = 1 to 9) to store ECC computation results when the specified number of bytes or 16-bit words has been computed.

The ECC result registers are used sequentially; one ECC result is stored in one ECC result register on the list, the next ECC result is stored in the next ECC result register on the list, and so forth, until the last ECC computation. The value of the GPMC\_ECCj\_RESULT register value is valid only when the programmed number of bytes or 16-bit words has been accumulated, which means that the same number of bytes or 16-bit words has been read from or written to the NAND device in sequence.

The GPMC\_ECC\_CONTROL[3-0] ECCPOINTER field must be set to the correct value to select the ECC result register to be used first in the list for the incoming ECC computation process. The ECCPointer can be read to determine which ECC register is used in the next ECC result storage for the ongoing ECC computation. The value of the GPMC\_ECCj\_RESULT register (j = 1 to 9) can be considered valid when ECCPOINTER equals j + 1. When the GPMC\_ECCj\_RESULT (where j = 9) is updated, ECCPOINTER is frozen at 10, and ECC computing is stopped (ECCENABLE = 0).

The ECC accumulator must be reset before any ECC computation accumulation process. The GPMC\_ECC\_CONTROL[8] ECCCLEAR bit must be set to 1 (nonpersistent bit) to clear the accumulator and all ECC result registers.

For each ECC result (each register, j = 1 to 9), the number of bytes or 16-bit words used for ECC computing accumulation can be selected from between two programmable values.

The ECCjRESULTSIZE bits (j = 1 to 9) in the GPMC\_ECC\_SIZE\_CONFIG register select which programmable size value (ECCSIZE0 or ECCSIZE1) must be used for this ECC result (stored in GPMC\_ECCj\_RESULT register ).

The ECCSIZE0 and ECCSIZE1 fields allow selection of the number of bytes or 16-bit words used for ECC computation accumulation. Any even values from 2 to 512 are allowed.

Flexibility in the number of ECCs computed and the number of bytes or 16-bit words used in the successive ECC computations enables different NAND page error-correction strategies. Usually based on 256 or 512 bytes and on 128 or 256 16-bit word, the number of ECC results required is a function of the NAND device page size. Specific ECC accumulation size can be used when computing the ECC on the NAND spare byte.

For example, with a 2 Kbyte data page 8-bit wide NAND device, eight ECCs accumulated on 256 bytes can be computed and added to one extra ECC computed on the 24 spare bytes area where the eight ECC results used for comparison and correction with the computed data page ECC are stored. The GPMC then provides nine GPMC\_ECCj\_RESULT registers (j= 1 to 9) to store the results. In this case, ECCSIZE0 is set to 256, and ECCSIZE1 is set to 24; the ECC[1-8]RESULTSIZE bits are cleared to 0, and the ECC9RESULTSIZE bit is set to 1.

#### **7.1.3.3.12.3.1.2 ECC Enabling**

The GPMC\_ECC\_CONFIG[3-1] ECCCS field selects the allocated chip-select. The GPMC\_ECC\_CONFIG[0] ECCENABLE bit enables ECC computation on the next detected read or write access to the selected chip-select.

The ECCPOINTER, ECCCLEAR, ECCSIZE, ECCjRESULTSIZE (where j = 1 to 9), ECC16B, and ECCCS fields must not be changed or cleared while an ECC computation is in progress.

The ECC accumulator and ECC result register must not be changed or cleared while an ECC computation is in progress.

Table 7-12 describes the ECC enable settings.

Table 7-12. ECC Enable Settings

| Bit Field                      | Register             | Value | Comments  |
|--------------------------------|----------------------|-------|---|
| ECCCS                          | GPMC_ECC_CONFIG      | 0-3h  | Selects the chip-select where ECC is computed   |
| ECC16B                         | GPMC_ECC_CONFIG      | 0/1   | Selects column number for ECC calculation   |
| ECCCLEAR                       | GPMC_ECC_CONTROL     | 0-7h  | Clears all ECC result registers   |
| ECCPOINTER                     | GPMC_ECC_CONTROL     | 0-7h  | A write to this bit field selects the ECC result register where the first ECC computation is stored. Set to 1 by default. |
| ECCSIZE1                       | GPMC_ECC_SIZE_CONFIG | 0-FFh | Defines ECCSIZE1  |
| ECCSIZE0                       | GPMC_ECC_SIZE_CONFIG | 0-FFh | Defines ECCSIZE0  |
| ECCjRESULTSIZE (j from 1 to 9) | GPMC_ECC_SIZE_CONFIG | 0/1   | Selects the size of ECCn result register  |
| ECCENABLE                      | GPMC_ECC_CONFIG      | 1     | Enables the ECC computation   |

#### 7.1.3.3.12.3.1.3 ECC Computation

The ECC algorithm is a multiple parity bit accumulation computed on the odd and even bit streams extracted from the byte or Word 16 streams. The parity accumulation is split into row and column accumulations, as shown in Figure 7-31 and Figure 7-32. The intermediate row and column parities are used to compute the upper level row and column parities. Only the final computation of each parity bit is used for ECC comparison and correction.

P1o = bit7 XOR bit5 XOR bit3 XOR bit1 on each byte of the data stream

P1e = bit6 XOR bit4 XOR bit2 XOR bit0 on each byte of the data stream

P2o = bit7 XOR bit6 XOR bit5 XOR bit2 on each byte of the data stream

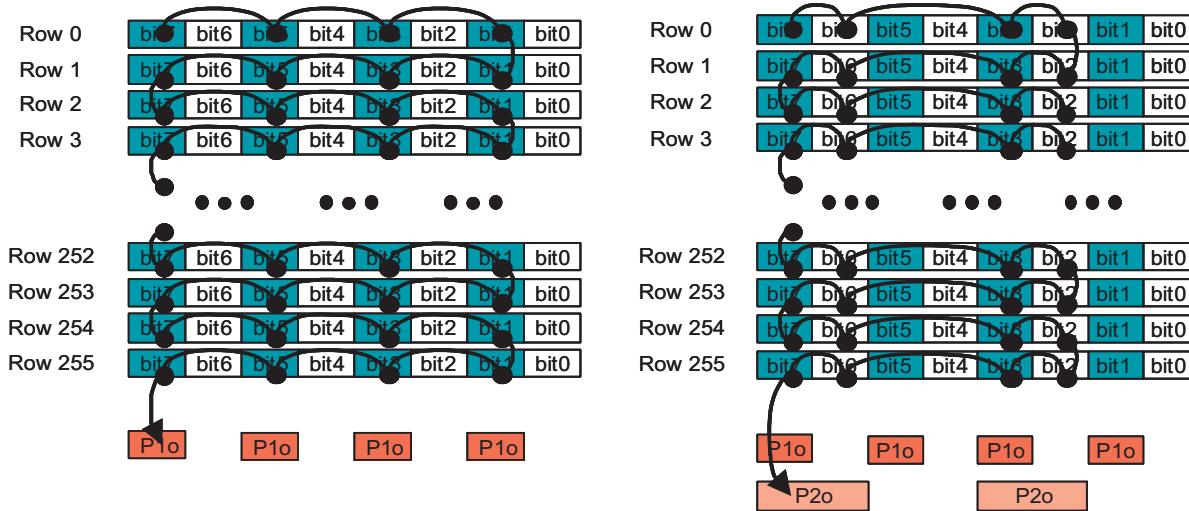
P2e = bit5 XOR bit4 XOR bit1 XOR bit0 on each byte of the data stream

P4o = bit7 XOR bit6 XOR bit5 XOR bit4 on each byte of the data stream

P4e = bit3 XOR bit2 XOR bit1 XOR bit0 on each byte of the data stream

Each column parity bit is XORed with the previous accumulated value.

Figure 7-31. Hamming Code Accumulation Algorithm (1 of 2)



For line parities, the bits of each new data are XORed together, and line parity bits are computed as:

$$P8e = \text{row0} \text{ XOR } \text{row2} \text{ XOR } \text{row4} \text{ XOR } \dots \text{ XOR } \text{row254}$$

$$P8o = \text{row1} \text{ XOR } \text{row3} \text{ XOR } \text{row5} \text{ XOR } \dots \text{ XOR } \text{row255}$$

$$P16e = \text{row0} \text{ XOR } \text{row1} \text{ XOR } \text{row4} \text{ XOR } \text{row5} \text{ XOR } \dots \text{ XOR } \text{row252} \text{ XOR } \text{row253}$$

$$P16o = \text{row2} \text{ XOR } \text{row3} \text{ XOR } \text{row6} \text{ XOR } \text{row7} \text{ XOR } \dots \text{ XOR } \text{row254} \text{ XOR } \text{row255}$$

Unused parity bits in the result registers are cleared to 0.

**Figure 7-32. Hamming Code Accumulation Algorithm (2 of 2)**

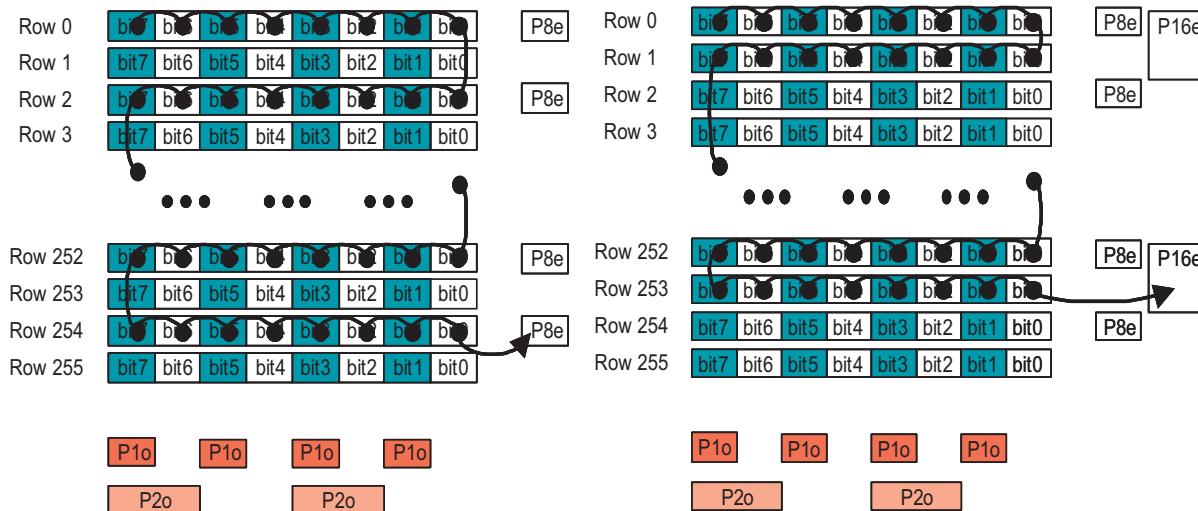


Figure 7-33 shows ECC computation for a 256-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and sixteen row parity bits (P8o-P16o-P32o--P1024o for odd parities, and P8e-P16e-P32e--P1024e for even parities).

**Figure 7-33. ECC Computation for a 256-Byte Data Stream (Read or Write)**

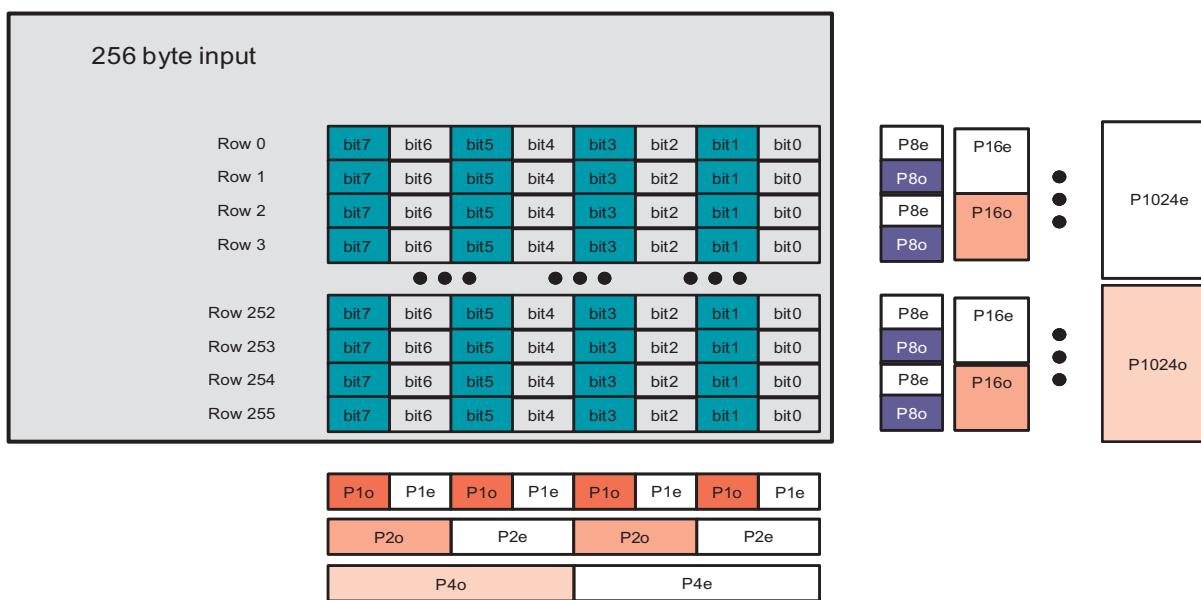
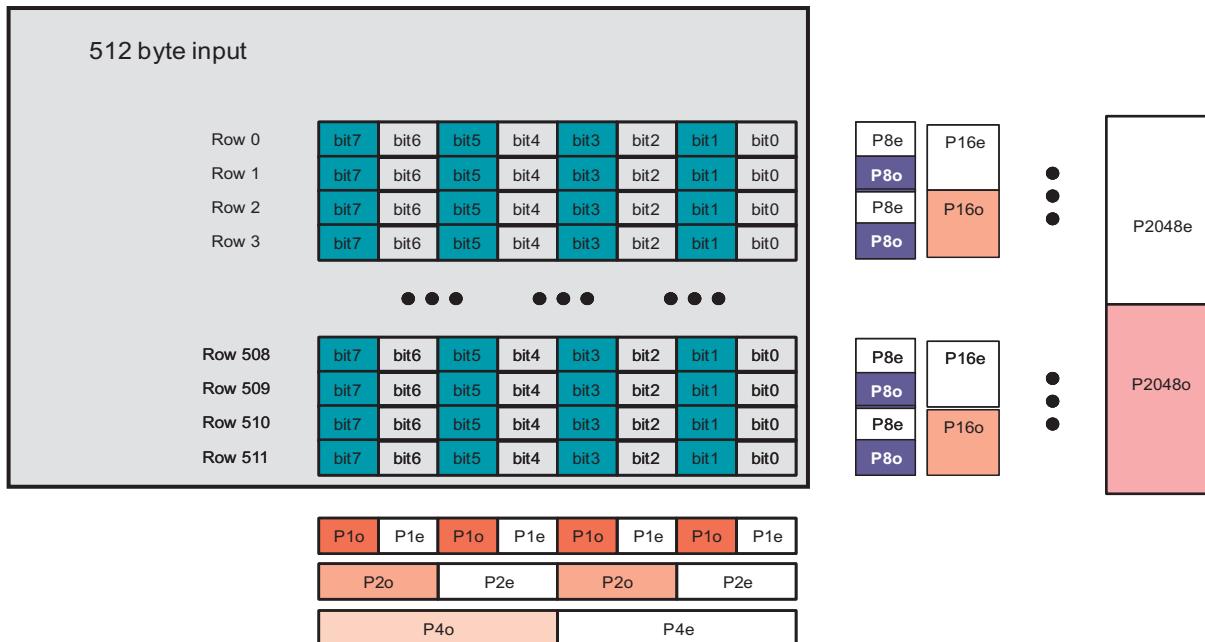


Figure 7-34 shows ECC computation for a 512-byte data stream (read or write). The result includes six column parity bits (P1o-P2o-P4o for odd parities, and P1e-P2e-P4e for even parities) and eighteen row parity bits (P8o-P16o-P32o--P1024o- - P2048o for odd parities, and P8e-P16e-P32e--P1024e- P2048e for even parities).

For a 2 Kbytes page, four 512 bytes ECC calculations plus one for the spare area are required. Results are stored in the GPMC\_ECCj\_RESULT registers ( $j = 1$  to 9).

**Figure 7-34. ECC Computation for a 512-Byte Data Stream (Read or Write)**



#### 7.1.3.3.12.3.1.4 ECC Comparison and Correction

To detect an error, the computed ECC result must be XORed with the parity value stored in the spare area of the accessed page.

- If the result of this logical XOR is all 0s, no error is detected and the read data is correct.
- If every second bit in the parity result is a 1, one bit is corrupted and is located at bit address (P2048o, P1024o, P512o, P256o, P128o, P64o, P32o, P16o, P8o, P4o, P2o, P1o). The software must correct the corresponding bit.
- If only one bit in the parity result is 1, it is an ECC error and the read data is correct.

#### 7.1.3.3.12.3.1.5 ECC Calculation Based on 8-Bit Word

The 8-bit based ECC computation is used for 8-bit wide NAND device interfacing.

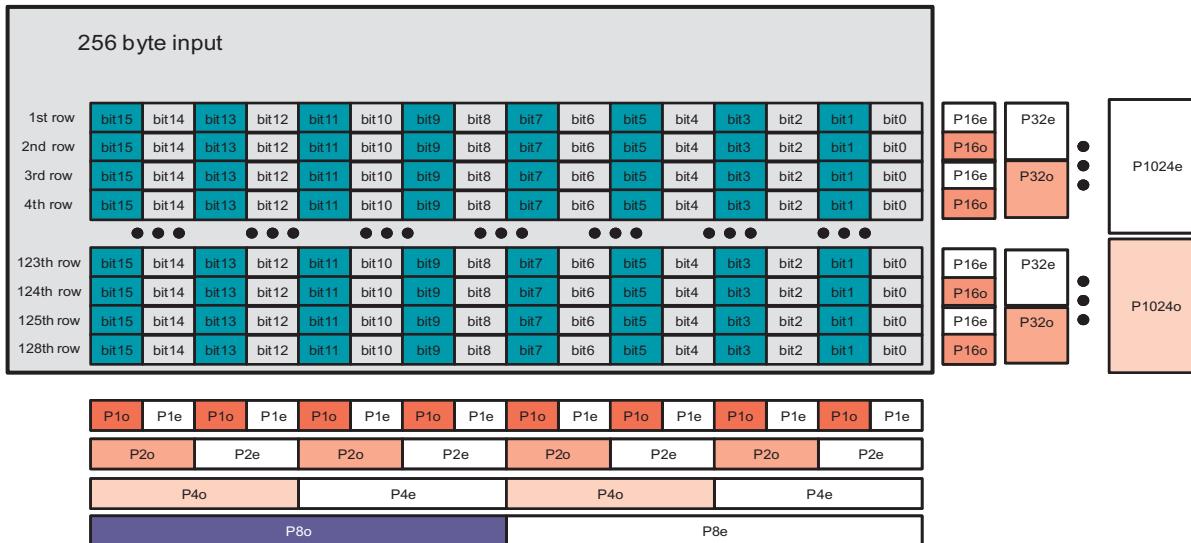
The 8-bit based ECC computation can be used for 16-bit wide NAND device interfacing to get backward compatibility on the error-handling strategy used with 8-bit wide NAND devices. In this case, the 16-bit wide data read from or written to the NAND device is fragmented into 2 bytes. According to little-endian access, the least significant bit (LSB) of the 16-bit wide data is ordered first in the byte stream used for 8-bit based ECC computation.

### 7.1.3.3.12.3.1.6 ECC Calculation Based on 16-Bit Word

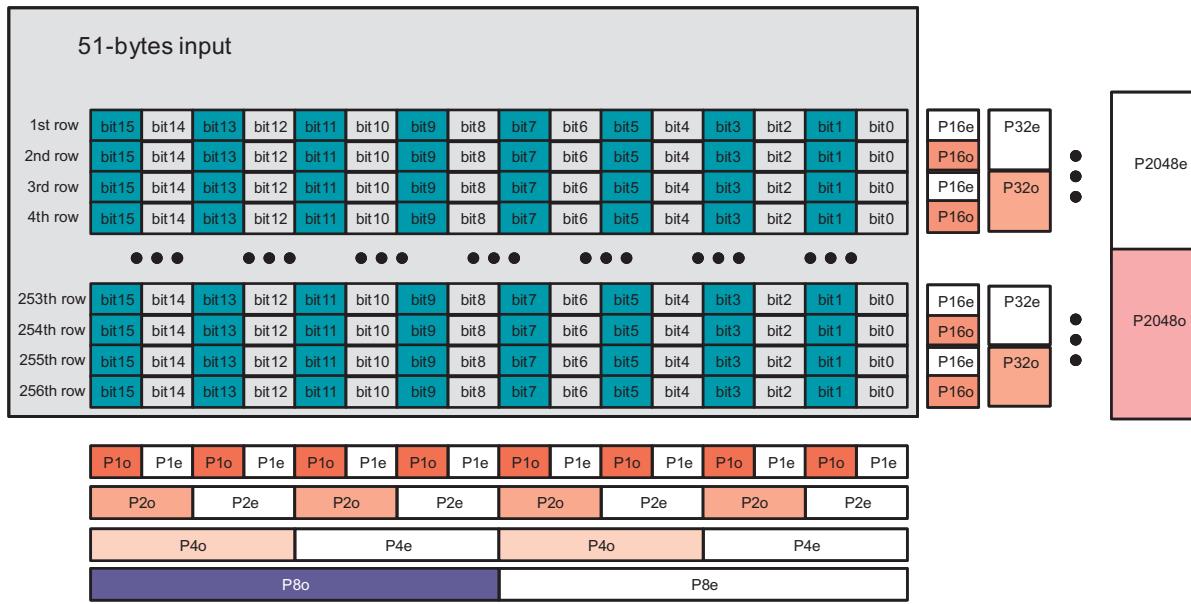
ECC computation based on a 16-bit word is used for 16-bit wide NAND device interfacing. This ECC computation is not supported when interfacing an 8-bit wide NAND device, and the GPMC\_ECC\_CONFIG[7] ECC16B bit must be cleared to 0 when interfacing an 8-bit wide NAND device.

The parity computation based on 16-bit words affects the row and column parity mapping. The main difference is that the odd and even parity bits P8o and P8e are computed on rows for an 8-bit based ECC while there are computed on columns for a 16-bit based ECC. [Figure 7-35](#) and [Figure 7-36](#).

**Figure 7-35. 128 Word16 ECC Computation**



**Figure 7-36. 256 Word16 ECC Computation**



### 7.1.3.3.12.3.2 BCH Code (Bose-Chaudhuri-Hocquenghem)

All references to Error Code Correction (ECC) in this subsection refer to the 4- to 16-bit error correction BCH code.

#### 7.1.3.3.12.3.2.1 Requirements

Read and write accesses to a NAND flash take place by whole pages, in a predetermined sequence: first the data byte page itself, then some spare bytes, including the BCH ECC (and other information). The NAND IC can cache a full page, including spares, for read and write accesses.

Typical page write sequence:

- Sequential write to NAND cache of main data + spare data, for a page. ECC is calculated on the fly. Calculated ECC may be inserted on the fly in the spares, or replaced by dummy accesses.
- When the calculated ECC is replaced by dummy accesses, it must be written to the cache in a second, separate phase. The ECC module is disabled during that time.
- NAND writes its cache line (page) to the array

Typical page read sequence:

- Sequential read of a page. ECC is calculated on the fly.
- ECC module buffers status determines the presence of errors.
- Accesses to several memories may be interleaved by the GPMC, but only one of those memories can be a NAND using the BCH engine at a time; in other words, only one BCH calculation (for example, for a single page) can be on-going at any time. Note also that the sequential nature of NAND accesses guarantees that the data is always written / read out in the same order. BCH-relevant accesses are selected by the GPMCs chip-select.
- Each page may hold up to 4 Kbytes of data, spare bytes not included. This means up to 8 x 512-byte BCH messages. Since all the data is written / read out first, followed by the BCH ECC, this means that the BCH engine must be able to hold 8 104-bit remainders or syndromes (or smaller, 52-bit ones) at the same time.

The BCH module has the capacity to store all remainders internally. After the page start, an internal counter is used to detect the 512-byte sector boundaries. On those boundaries, the current remainder is stored and the divider reset for the next calculation. At the end of the page, the BCH module contains all remainders.

- NAND access cycles hold 8 or 16 bits of data each (1 or 2 bytes); Each NAND cycle takes at least 4 cycles of the GPMCs internal clock. This means the NAND flash timing parameters must define a RDCYCLETIME and a WRCYCLETIME of at least 4 clock cycles after optimization when using the BCH calculator.
- The spare area is assumed to be large enough to hold the BCH ECC, that is, to have at least a message of 13 bytes available per 512-byte sector of data. The zone of unused spare area by the ECC may or may not be protected by the same ECC scheme, by extending the BCH message beyond 512 bytes (maximum codeword is 1023-byte long, ECC included, which leaves a lot of space to cover some spares bytes).

### 7.1.3.3.12.3.2.2 Memory-Mapping of the BCH Codeword

BCH encoding considers a block of data to protect as a polynomial message  $M(x)$ . In our standard case, 512 bytes of data (that is, 2 bits = 4096 bits) are seen as a polynomial of degree  $2 - 1 = 4095$ , with parameters ranging from  $M_0$  to  $M_{4095}$ . For 512 bytes of data, 52 bits are required for 4-bit error correction, and 104 bits are required for 8-bit error correction and 207 bits are required for 16-bit error correction. The ECC is a remainder polynomial  $R(x)$  of degree 103 (or 51, depending on the selected mode). The complete codeword  $C(x)$  is the concatenation of  $M(x)$  and  $R(x)$  as shown in [Table 7-13](#).

**Table 7-13. Flattened BCH Codeword Mapping (512 Bytes + 104 Bits)**

| Bit number | Message $M(x)$ | ECC $R(x)$ |       |      |     |       |
|------------|----------------|------------|-------|------|-----|-------|
|            | M4095          | ...        | $M_0$ | R103 | ... | $R_0$ |

If the message is extended by the addition of spare bytes to be protected by the same ECC, the principle is still valid. For example, a 3-byte extension of the message gives a polynomial message  $M(x)$  of degree  $((512 + 3) \times 8) - 1 = 4119$ , for a total of  $3 + 13 = 16$  spare bytes of spare, all protected as part of the same codeword.

The message and the ECC bits are manipulated and mapped in the GPMC byte-oriented system. The ECC bits are stored in:

- GPMC\_BCH\_RESULT0\_i
- GPMC\_BCH\_RESULT1\_i
- GPMC\_BCH\_RESULT2\_i
- GPMC\_BCH\_RESULT3\_i

### 7.1.3.3.12.3.2.3 Memory Mapping of the Data Message

The data message mapping shall follow the following rules:

- Bit endianness within a byte is little-endian, that is, the bytes LS bit is also the lowest-degree polynomial parameter: a byte b7-b0 (with b0 the LS bit) represents a segment of polynomial  $b_7 * x + b_6 * x + \dots + b_0 * x$
- The message is mapped in the NAND starting with the highest-order parameters, that is, in the lowest addresses of a NAND page.
- Byte endianness within the NANDs 16-bit words is big endian. This means that the same message mapped in 8- and 16-bit memories has the same content at the same byte address.

The BCH module has no visibility over actual addresses. The most important point is the sequence of data word the BCH sees. However, the NAND page is always scanned incrementally in read and write accesses, and this produces the mapping patterns described in the following.

[Table 7-14](#) and [Table 7-15](#) show the mapping of the same 512-byte vector (typically a BCH message) in the NAND memory space. Note that the byte 'address' is only an offset modulo 512 (200h), since the same page may contain several contiguous 512-byte sectors (BCH blocks). The LSB and MSB are respectively the bits  $M_0$  and  $M(2^{12}-1)$  of the codeword mapping given above. In both cases the data vectors are aligned, that is, their boundaries coincide with the RAMs data word boundaries.

**Table 7-14. Aligned Message Byte Mapping in 8-bit NAND**

| Byte Offset | 8-Bit Word            |
|-------------|-----------------------|
| 0           | (msb) Byte 511 (1FFh) |
| 1h          | Byte 510 (1FEh)       |
| :           | :                     |
| 1FFh        | Byte 0 (0) (LSB)      |

**Table 7-15. Aligned Message Byte Mapping in 16-bit NAND**

| Byte Offset | 16-Bit Words MSB | 16-Bit Words LSB      |
|-------------|------------------|-----------------------|
| 0           | Byte 510 (1FEh)  | (msb) Byte 511 (1FFh) |
| 2h          | Byte 508 (1FCh)  | Byte 509 (1FDh)       |
| :           | :                | :                     |
| 1FEh        | Byte 0 (0)       | (lsb) Byte 1 (1)      |

[Table 7-16](#) and [Table 7-17](#) show the mapping in memory of arbitrarily-sized messages, starting on access (byte or 16-bit word) boundaries for more clarity. Note that message may actually start and stop on arbitrary nibbles. A nibble is a 4-bit entity. The unused nibbles are not discarded, and they can still be used by the BCH module, but as part of the next message section (for example, on another sectors ECC).

**Table 7-16. Aligned Nibble Mapping of Message in 8-bit NAND**

| Byte Offset | 8-Bit Word                    |                               |
|-------------|-------------------------------|-------------------------------|
|             | 4-Bit Most Significant Nibble | 4-Bit Less Significant Nibble |
| 1           | (MSB) Nibble S-1              | Nibble S-2                    |
| 2           | Nibble S-3                    | Nibble S-4                    |
| :           | :                             | :                             |
| S/2 - 2     | Nibble 3                      | Nibble 2                      |
| S/2 - 1     | Nibble 1                      | Nibble 0 (LSB)                |

**Table 7-17. Misaligned Nibble Mapping of Message in 8-bit NAND**

| Byte Offset | 8-Bit Word                    |                               |
|-------------|-------------------------------|-------------------------------|
|             | 4-Bit Most Significant Nibble | 4-Bit Less Significant Nibble |
| 1           | (MSB) Nibble S-1              | Nibble S-2                    |
| 2           | Nibble S-3                    | Nibble S-4                    |
| :           | :                             | :                             |
| (S+1)/2 - 2 | Nibble 2                      | Nibble 1                      |
| (S+1)/2 - 1 | Nibble 0 (LSB)                |                               |

**Table 7-18. Aligned Nibble Mapping of Message in 16-bit NAND**

| Byte Offset | 16-Bit Word                   |                |                               |            |  |
|-------------|-------------------------------|----------------|-------------------------------|------------|--|
|             | 4-Bit Most Significant Nibble |                | 4-Bit Less Significant Nibble |            |  |
| 0           | Nibble S-3                    | Nibble S-4     | (MSB) Nibble S-1              | Nibble S-2 |  |
| 2           | Nibble S-7                    | Nibble S-8     | Nibble S-5                    | Nibble S-6 |  |
| :           | :                             | :              | :                             | :          |  |
| S/2 - 4     | Nibble 5                      | Nibble 4       | Nibble 7                      | Nibble 6   |  |
| S/2 - 2     | Nibble 1                      | Nibble 0 (LSB) | Nibble 3                      | Nibble 2   |  |

**Table 7-19. Misaligned Nibble Mapping of Message in 16-bit NAND (1 Unused Nibble)**

| Byte Offset | 16-Bit Word                   |            |                               |            |
|-------------|-------------------------------|------------|-------------------------------|------------|
|             | 4-Bit Most Significant Nibble |            | 4-Bit Less Significant Nibble |            |
| 0           | Nibble S-3                    | Nibble S-4 | (MSB) Nibble S-1              | Nibble S-2 |
| 2           | Nibble S-7                    | Nibble S-8 | Nibble S-5                    | Nibble S-6 |
| :           | :                             | :          | :                             | :          |
| (S+1)/2 - 4 | Nibble 4                      | Nibble 3   | Nibble 6                      | Nibble 5   |
| (S+1)/2 - 2 | Nibble 0 (LSB)                |            | Nibble 2                      | Nibble 1   |

**Table 7-20. Misaligned Nibble Mapping of Message in 16-bit NAND (2 Unused Nibble)**

| Byte Offset | 16-Bit Word                   |            |                               |                |
|-------------|-------------------------------|------------|-------------------------------|----------------|
|             | 4-Bit Most Significant Nibble |            | 4-Bit Less Significant Nibble |                |
| 0           | Nibble S-3                    | Nibble S-4 | (MSB) Nibble S-1              | Nibble S-2     |
| 2           | Nibble S-7                    | Nibble S-8 | Nibble S-5                    | Nibble S-6     |
| :           | :                             | :          | :                             | :              |
| (S+2)/2 - 4 | Nibble 3                      | Nibble 2   | Nibble 5                      | Nibble 4       |
| (S+2)/2 - 2 |                               |            | Nibble 1                      | Nibble 0 (LSB) |

**Table 7-21. Misaligned Nibble Mapping of Message in 16-bit NAND (3 Unused Nibble)**

| Byte Offset | 16-Bit Word                   |            |                               |            |
|-------------|-------------------------------|------------|-------------------------------|------------|
|             | 4-Bit Most Significant Nibble |            | 4-Bit Less Significant Nibble |            |
| 0           | Nibble S-3                    | Nibble S-4 | (MSB) Nibble S-1              | Nibble S-2 |
| 2           | Nibble S-7                    | Nibble S-8 | Nibble S-5                    | Nibble S-6 |
| :           | :                             | :          | :                             | :          |
| (S+3)/2 - 4 | Nibble 2                      | Nibble 1   | Nibble 4                      | Nibble 3   |
| (S+3)/2 - 2 |                               |            | Nibble 0 (LSB)                |            |

Note that many other cases exist than the ones represented above, for example, where the message does not start on a word boundary.

#### 7.1.3.3.12.3.2.4 Memory Mapping of the ECC

The ECC (or remainder) is presented by the BCH module as a single 104-bit (or 52-bit), little-endian vector. It is up to the software to fetch those 13 bytes (or 6 bytes) from the modules interface, then store them to the NANDs spare area (page write) or to an intermediate buffer for comparison with the stored ECC (page read). There are no constraints on the ECC mapping inside the spare area: it is a softwarecontrolled operation.

However, it is advised to maintain a coherence in the respective formats of the message or the ECC remainder once they have been read out of the NAND. The error correction algorithm works from the complete codeword (concatenated message and remainder) once an error has been detected. The creation of this codeword should be made as straightforward as possible.

There are cases where the same NAND access contains both data and the ECC protecting that data. This is the case when the data/ECC boundary (which can be on any nibble) does not coincide with an access boundary. The ECC is calculated on-the-fly following the write. In that case, the write must also contain part of the ECC because it is impossible to insert the ECC on-the-fly. Instead:

- During the initial page write (BCH encoding), the ECC is replaced by dummy bits. The BCH encoder is by definition turned OFF during the ECC section, so the BCH result is unmodified.
  - During a second phase, the ECC is written to the correct location, next to the actual data.
  - The completed line buffer is then written to the NAND array.

### **7.1.3.3.12.3.2.5 Wrapping Modes**

For a given wrapping mode, the module automatically goes through a specific number of sections, as data is being fed into the module. For each section, the BCH core can be enabled (in which case the data is fed to the BCH divider) or not (in which case the BCH simply counts to the end of the section). When enabled, the data is added to the ongoing calculation for a given sector number (for example, number 0).

Wrapping modes are described below. To get a better understanding and see the real-life read and write sequences implemented with each mode, see [Section 7.1.3.3.12.3.3](#).

For each mode:

- A sequence describes the mode in pseudo-language, with for each section the size and the buffer used for ECC processing (if ON). The programmable lengths are size, size0 and size1.
  - A checksum condition is given. If the checksum condition is not respected for a given mode, the modules behavior is unpredictable. S is the number of sectors in the page; size0 and size1 are the section sizes programmed for the mode, in nibbles.

Note that wrapping modes 8, 9, 10, and 11 insert a 1-nibble padding where the BCH processing is OFF. This is intended for t = 4 ECC, where ECC is 6 bytes long and the ECC area is expected to include (at least) 1 unused nibble to remain byte-aligned.

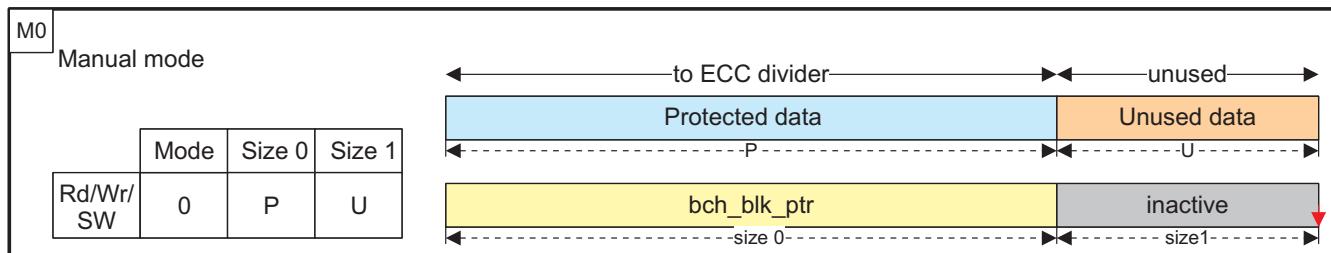
### **7.1.3.3.12.3.2.6 Manual Mode (0x0)**

This mode is intended for short sequences, added manually to a given buffer through the software data port input. A complete page may be built out of several such sequences.

To process an arbitrary sequence of 4-bit nibbles, accesses to the software data port shall be made, containing the appropriate data. If the sequence end does not coincide with an access boundary (for example, to process 5 nibbles = 20 bits in 16-bit access mode) and those nibbles need to be skipped, a number of unused nibbles shall be programmed in size1 (in the same example: 5 nibbles to process + 3 to discard = 8 nibbles = exactly  $2 \times 16$ -bit accesses: we must program size0 = 5, size1 = 3).

[Figure 7-37](#) shows the manual mode sequence and mapping. In this figure, size and size0 are the same parameter.

**Figure 7-37. Manual Mode Sequence and Mapping**



Section processing sequence:

- One time with buffer
  - size0 nibbles of data, processing ON
  - size1 nibbles of unused data, processing OFF

Checksum: size0 + size1 nibbles must fit in a whole number of accesses.

In the following sections, S is the number of sectors in the page.

#### 7.1.3.3.12.3.2.7 Mode 0x1

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

#### 7.1.3.3.12.3.2.8 Mode 0xA (10)

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
  - 1 nibble pad spare, processing OFF
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

#### 7.1.3.3.12.3.2.9 Mode 0x2

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

#### 7.1.3.3.12.3.2.10 Mode 0x3

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

#### **7.1.3.3.12.3.2.11 Mode 0x7**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = size0 + (S - size1)

#### **7.1.3.3.12.3.2.12 Mode 0x8**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time with buffer 0
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1+size1))

#### **7.1.3.3.12.3.2.13 Mode 0x4**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - size1)

#### **7.1.3.3.12.3.2.14 Mode 0x9**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- One time (no buffer used)
  - size0 nibbles spare, processing OFF
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = size0 + (S - (1+size1))

**7.1.3.3.12.3.2.15 Mode 0x5**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + size1)

**7.1.3.3.12.3.2.16 Mode 0xB (11)**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat with buffer 0 to S-1
  - 1 nibble padding spare, processing OFF
  - size1 nibbles spare, processing ON

Checksum: Spare area size (nibbles) = S - (size0 + 1 + size1)

**7.1.3.3.12.3.2.17 Mode 0x6**

Page processing sequence:

- Repeat with buffer 0 to S-1
  - 512-byte data, processing ON
- Repeat with buffer 0 to S-1
  - size0 nibbles spare, processing ON
- Repeat S times (no buffer used)
  - size1 nibbles spare, processing OFF

Checksum: Spare area size (nibbles) = S - (size0 + size1)

### 7.1.3.3.12.3.3 Supported NAND Page Mappings and ECC Schemes

The following rules apply throughout the entire mapping description:

- Main data area (sectors) size is hardcoded to 512 bytes.
- Spare area size is programmable.
- All page sections (of main area data bytes, protected spare bytes, unprotected spare bytes, and ECC) are defined as explained in [Section 7.1.3.3.12.3.2.3](#).

Each one of the following sections shows a NAND page mapping example (per-sector spare mappings, pooled spare mapping, per-sector spare mapping, with ECC separated at the end of the page).

In the mapping diagrams, sections that belong to the same BCH codeword have the same color (blue or green); unprotected sections are not covered (orange) by the BCH scheme.

Below each mapping diagram, a write (encoding) and read (decoding: syndrome generation) sequence is given, with the number of the active buffers at each point in time (yellow). In the inactive zones (grey), no computing is taking place but the data counter is still active.

In [Figure 7-38](#) to [Figure 7-40](#), tables on the left summarize the mode, size0, size1 parameters to program for respectively write and read processing of a page, with the given mapping, where:

- P is the size of spare byte section Protected by the ECC (in nibbles)
- U is the size of spare byte section Unprotected by the ECC (in nibbles)
- E is the size of the ECC itself (in nibbles)
- S is the number of Sectors per page (2 in the current diagrams)

Each time the processing of a BCH block is complete (ECC calculation for write/encoding, syndrome generation for read/decoding, indicated by red arrows), the update pointer is pulsed. Note that the processing for block 0 can be the first or the last to complete, depending on the NAND page mapping and operation (read or write). All examples show a page size of 1kByte + spares, that is, S = 2 sectors of 512 bytes. The same principles can be extended to larger pages by adding more sectors.

The actual BCH codeword size is used during the error location work to restrict the search range: by definition, errors can only happen in the codeword that was actually written to the NAND, and not in the mathematical codeword of  $n = 2 - 1 = 8191$  bits. That codeword (higher-order bits) is all-zero and implicit during computations.

The actual BCH codeword size depends on the mode, on the programmed sizes and on the sector number (all sizes in nibbles):

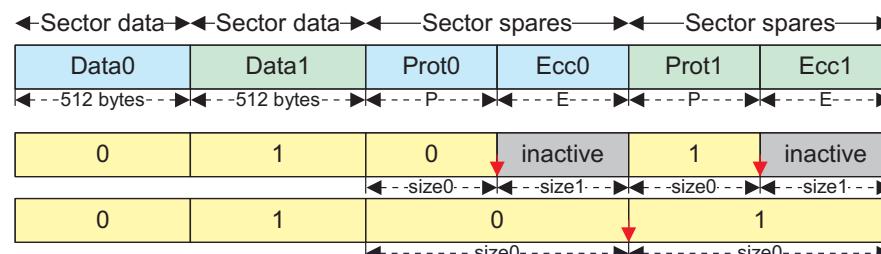
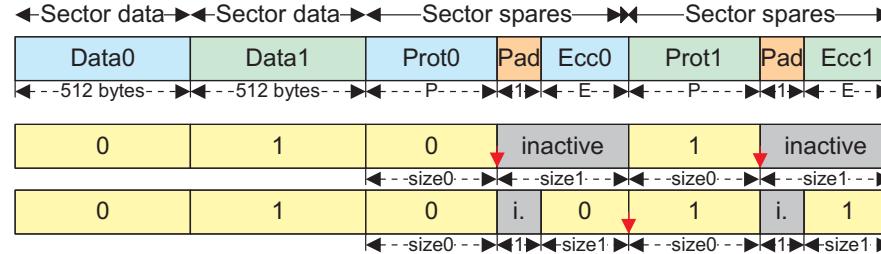
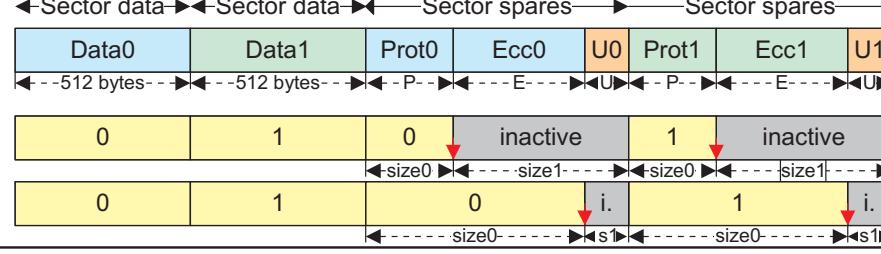
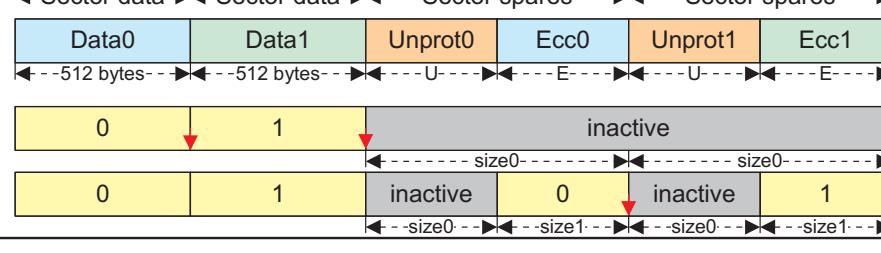
- Spares mapped and protected per sector ([Figure 7-38](#): see M1-M2-M3-M9-M10):
  - all sectors: (512) + P + E
- Spares pooled and protected by sector 0 ([Figure 7-38](#): see M5-M6):
  - sector 0 codeword: (512) + P + E
  - other sectors: (512) + E
- Unprotected spares ([Figure 7-38](#): see M4-M7-M8-M11-M12):
  - all codewords (512) + E

### 7.1.3.3.12.3.3.1 Per-Sector Spare Mappings

In these schemes (Figure 7-38), each 512-byte sector of the main area has its own dedicated section of the spare area. The spare area of each sector is composed of:

- ECC, which must be located after the data it protects
- other data, which may or may not be protected by the sectors ECC

**Figure 7-38. NAND Page Mapping and ECC: Per-Sector Schemes**

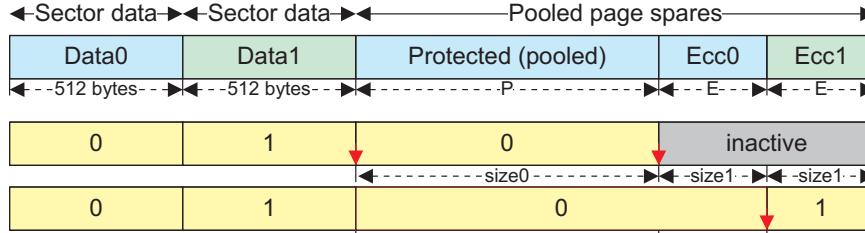
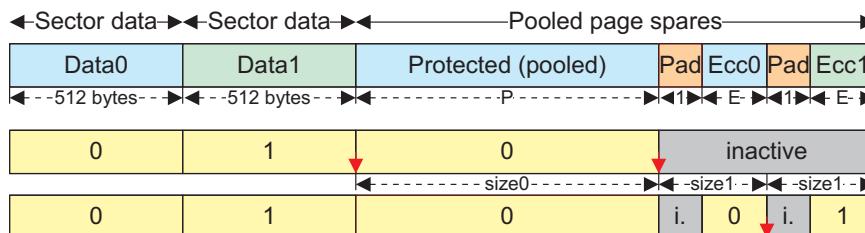
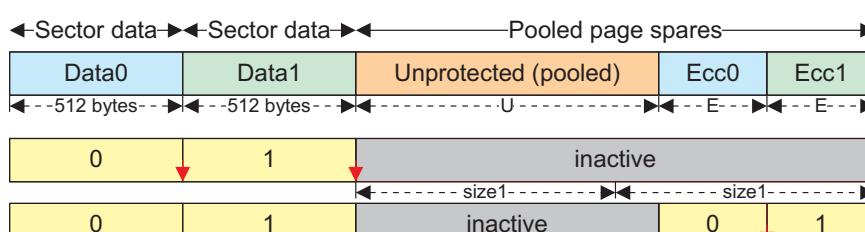
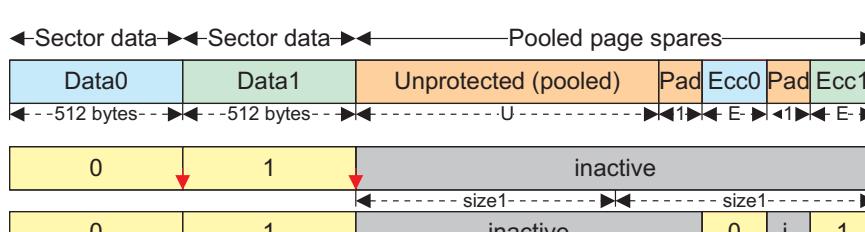
| M1    | Per-sector spares<br>Spares covered by sector ECC per sector ECC mapping.      |  <table border="1"> <thead> <tr> <th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>P</td><td>E</td></tr> <tr> <td>Read</td><td>P+E</td><td>0</td></tr> </tbody> </table>     | Mode | Size0 | Size1 | Write | P   | E   | Read | P+E | 0 |
|-------|--|---|------|-------|-------|-------|-----|-----|------|-----|---|
| Mode  | Size0  | Size1   |      |       |       |       |     |     |      |     |   |
| Write | P  | E   |      |       |       |       |     |     |      |     |   |
| Read  | P+E  | 0   |      |       |       |       |     |     |      |     |   |
| M2    | Per-sector spares<br>Spares covered by sector ECC per sector, left-padded ECC. |  <table border="1"> <thead> <tr> <th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>P</td><td>1+E</td></tr> <tr> <td>Read</td><td>P</td><td>E</td></tr> </tbody> </table>    | Mode | Size0 | Size1 | Write | P   | 1+E | Read | P   | E |
| Mode  | Size0  | Size1   |      |       |       |       |     |     |      |     |   |
| Write | P  | 1+E   |      |       |       |       |     |     |      |     |   |
| Read  | P  | E   |      |       |       |       |     |     |      |     |   |
| M3    | Per-sector spares<br>Spares covered by sector ECC, ECC not right-aligned.      |  <table border="1"> <thead> <tr> <th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>P</td><td>E+U</td></tr> <tr> <td>Read</td><td>P+E</td><td>U</td></tr> </tbody> </table> | Mode | Size0 | Size1 | Write | P   | E+U | Read | P+E | U |
| Mode  | Size0  | Size1   |      |       |       |       |     |     |      |     |   |
| Write | P  | E+U   |      |       |       |       |     |     |      |     |   |
| Read  | P+E  | U   |      |       |       |       |     |     |      |     |   |
| M4    | Per-sector spares<br>Spares not covered by ECC, ECC right-aligned per sector.  |  <table border="1"> <thead> <tr> <th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>U+E</td><td>0</td></tr> <tr> <td>Read</td><td>U</td><td>E</td></tr> </tbody> </table>   | Mode | Size0 | Size1 | Write | U+E | 0   | Read | U   | E |
| Mode  | Size0  | Size1   |      |       |       |       |     |     |      |     |   |
| Write | U+E  | 0   |      |       |       |       |     |     |      |     |   |
| Read  | U  | E   |      |       |       |       |     |     |      |     |   |

### 7.1.3.3.12.3.3.2 Pooled Spare Mapping

In these schemes (Figure 7-39), the spare area is pooled for the page.

- The ECC of each sector is aligned at the end of the spare area.
- The non-ECC spare data may or may not be covered by the ECC of sector 0

**Figure 7-39. NAND Page Mapping and ECC: Pooled Spare Schemes**

| M5    | Pooled spares<br>Spares covered by ECC0.<br>All ECC at the end (of page).       | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>7</td><td>P</td><td>E</td></tr> <tr> <td>Read</td><td>3</td><td>P</td><td>E</td></tr> </tbody> </table>           |          | Mode | Size0 | Size1 | Write | 7 | P | E        | Read | 3 | P | E |
|-------|---|---|----------|------|-------|-------|-------|---|---|----------|------|---|---|---|
|       | Mode  | Size0   | Size1    |      |       |       |       |   |   |          |      |   |   |   |
| Write | 7   | P   | E        |      |       |       |       |   |   |          |      |   |   |   |
| Read  | 3   | P   | E        |      |       |       |       |   |   |          |      |   |   |   |
| M6    | Pooled spares<br>Spares covered by ECC0.<br>All ECC at the end, left-padded.    | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>7</td><td>P</td><td>1+E</td></tr> <tr> <td>Read</td><td>8</td><td>P</td><td>E</td></tr> </tbody> </table>        |          | Mode | Size0 | Size1 | Write | 7 | P | 1+E      | Read | 8 | P | E |
|       | Mode  | Size0   | Size1    |      |       |       |       |   |   |          |      |   |   |   |
| Write | 7   | P   | 1+E      |      |       |       |       |   |   |          |      |   |   |   |
| Read  | 8   | P   | E        |      |       |       |       |   |   |          |      |   |   |   |
| M7    | Pooled spares<br>Spares not covered by ECC.<br>All ECC at the end.              | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>0</td><td>U/S +E</td></tr> <tr> <td>Read</td><td>4</td><td>U</td><td>E</td></tr> </tbody> </table>    |          | Mode | Size0 | Size1 | Write | 6 | 0 | U/S +E   | Read | 4 | U | E |
|       | Mode  | Size0   | Size1    |      |       |       |       |   |   |          |      |   |   |   |
| Write | 6   | 0   | U/S +E   |      |       |       |       |   |   |          |      |   |   |   |
| Read  | 4   | U   | E        |      |       |       |       |   |   |          |      |   |   |   |
| M8    | Pooled spares<br>Spares not covered by ECC.<br>All ECC at the end, left-padded. | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>0</td><td>U/S +1+E</td></tr> <tr> <td>Read</td><td>9</td><td>U</td><td>E</td></tr> </tbody> </table>  |          | Mode | Size0 | Size1 | Write | 6 | 0 | U/S +1+E | Read | 9 | U | E |
|       | Mode  | Size0   | Size1    |      |       |       |       |   |   |          |      |   |   |   |
| Write | 6   | 0   | U/S +1+E |      |       |       |       |   |   |          |      |   |   |   |
| Read  | 9   | U   | E        |      |       |       |       |   |   |          |      |   |   |   |

### 7.1.3.3.12.3.3.3 Per-Sector Spare Mapping, With ECC Separated at the End of the Page

In these schemes (Figure 7-40), each 512-byte sector of the main area is associated with two sections of the spare area.

- ECC section, all aligned at the end of the page
- other data section, aligned before the ECCs, each of which may or may not be protected by its sectors ECC

**Figure 7-40. NAND Page Mapping and ECC: Per-Sector Schemes, with Separate ECC**

| M9    | Per-sector spares, separate ECC<br>Spares covered by sector ECC.<br>All ECC at the end.              | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>P</td><td>E</td></tr> <tr> <td>Read</td><td>5</td><td>P</td><td>E</td></tr> </tbody> </table>      |       | Mode | Size0 | Size1 | Write | 6 | P | E     | Read | 5  | P  | E |
|-------|--|--|-------|------|-------|-------|-------|---|---|-------|------|----|----|---|
|       | Mode   | Size0  | Size1 |      |       |       |       |   |   |       |      |    |    |   |
| Write | 6  | P  | E     |      |       |       |       |   |   |       |      |    |    |   |
| Read  | 5  | P  | E     |      |       |       |       |   |   |       |      |    |    |   |
| M10   | Per-sector spares, separate ECC<br>Spares covered by sector ECC.<br>All ECC at the end, left-padded. | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>P</td><td>1+E</td></tr> <tr> <td>Read</td><td>11</td><td>P</td><td>E</td></tr> </tbody> </table>   |       | Mode | Size0 | Size1 | Write | 6 | P | 1+E   | Read | 11 | P  | E |
|       | Mode   | Size0  | Size1 |      |       |       |       |   |   |       |      |    |    |   |
| Write | 6  | P  | 1+E   |      |       |       |       |   |   |       |      |    |    |   |
| Read  | 11   | P  | E     |      |       |       |       |   |   |       |      |    |    |   |
| M11   | Per-sector spares, separate ECC<br>Spares not covered by ECC.<br>All ECC at the end.                 | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>0</td><td>U+E</td></tr> <tr> <td>Read</td><td>4</td><td>SU</td><td>E</td></tr> </tbody> </table>   |       | Mode | Size0 | Size1 | Write | 6 | 0 | U+E   | Read | 4  | SU | E |
|       | Mode   | Size0  | Size1 |      |       |       |       |   |   |       |      |    |    |   |
| Write | 6  | 0  | U+E   |      |       |       |       |   |   |       |      |    |    |   |
| Read  | 4  | SU   | E     |      |       |       |       |   |   |       |      |    |    |   |
| M12   | Per-sector spares, separate ECC<br>Spares not covered by ECC.<br>All ECC at the end, left-padded.    | <table border="1"> <thead> <tr> <th></th><th>Mode</th><th>Size0</th><th>Size1</th></tr> </thead> <tbody> <tr> <td>Write</td><td>6</td><td>0</td><td>U+1+E</td></tr> <tr> <td>Read</td><td>9</td><td>SU</td><td>E</td></tr> </tbody> </table> |       | Mode | Size0 | Size1 | Write | 6 | 0 | U+1+E | Read | 9  | SU | E |
|       | Mode   | Size0  | Size1 |      |       |       |       |   |   |       |      |    |    |   |
| Write | 6  | 0  | U+1+E |      |       |       |       |   |   |       |      |    |    |   |
| Read  | 9  | SU   | E     |      |       |       |       |   |   |       |      |    |    |   |

#### 7.1.3.3.12.4 Prefetch and Write-Posting Engine

NAND device data access cycles are usually much slower than the MPU system frequency; such NAND read or write accesses issued by the processor will impact the overall system performance, especially considering long read or write sequences required for NAND page loading or programming. To minimize this effect on system performance, the GPMC includes a prefetch and write-posting engine, which can be used to read from or write to any chip-select location in a buffered manner.

The prefetch and write-posting engine is a simplified embedded-access requester that presents requests to the access engine on a user-defined chip-select target. The access engine interleaves these requests with any request coming from the L3 interface; as a default the prefetch and write-posting engine has the lowest priority.

The prefetch and write-posting engine is dedicated to data-stream access (as opposed to random data access); thus, it is primarily dedicated to NAND support. The engine does not include an address generator; the request is limited to chip-select target identification. It includes a 64-byte FIFO associated with a DMA request synchronization line, for optimal DMA-based use.

The prefetch and write-posting engine uses an embedded 64 bytes (32 16-bit word) FIFO to prefetch data from the NAND device in read mode (prefetch mode) or to store host data to be programmed into the NAND device in write mode (write-posting mode). The FIFO draining and filling (read and write) can be controlled either by the MPU through interrupt synchronization (an interrupt is triggered whenever a programmable threshold is reached) or the sDMA through DMA request synchronization, with a programmable request byte size in both prefetch or posting mode.

The prefetch and write-posting engine includes a single memory pool. Therefore, only one mode, read or write, can be used at any given time. In other words, the prefetch and write-posting engine is a single-context engine that can be allocated to only one chip-select at a time for a read prefetch or a write-posting process.

The engine does not support atomic command and address phase programming and is limited to linear memory read or write access. In consequence, it is limited to NAND data-stream access. The engine relies on the MPU NAND software driver to control block and page opening with the correct data address pointer initialization, before the engine can read from or write to the NAND memory device.

Once started, the engine data reads and writes sequencing is solely based on FIFO location availability and until the total programmed number of bytes is read or written.

Any host-concurrent accesses to a different chip-select are correctly interleaved with ongoing engine accesses. The engine has the lowest priority access so that host accesses to a different chip-select do not suffer a large latency.

A round-robin arbitration scheme can be enabled to ensure minimum bandwidth to the prefetch and write-posting engine in the case of back-to-back direct memory requests to a different chip-select. If the GPMC\_PREFETCH\_CONFIG1[23] PFPWENROUNDROBIN bit is enabled, the arbitration grants the prefetch and write posting engine access to the GPMC bus for a number of requests programmed in the GPMC\_PREFETCH\_CONFIG1[19-16] PFPWWEIGHTEDPRIO field.

The prefetch/write-posting engine read or write request is routed to the access engine with the chip-select destination ID. After the required arbitration phase, the access engine processes the request as a single access with the data access size equal to the device size specified in the corresponding chip-select configuration.

The destination chip-select configuration must be set to the NAND protocol-compatible configuration for which address lines are not used (the address bus is not changed from its current value). Selecting a different chip-select configuration can produce undefined behavior.

#### 7.1.3.3.12.4.1 General Facts About the Engine Configuration

The engine can be configured only if the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit is deasserted.

The engine must be correctly configured in prefetch or write-posting mode and must be linked to a NAND chip-select before it can be started. The chip-select is linked using the GPMC\_PREFETCH\_CONFIG1[26-24] ENGINECSSELECTOR field.

In both prefetch and write-posting modes, the engine respectively uses byte or 16-bit word access requests for an 8- or 16-bit wide NAND device attached to the linked chip-select. The FIFOThreshold and TRANSFERCOUNT fields must be programmed accordingly as a number of bytes or a number of 16-bit word.

When the GPMC\_PREFETCH\_CONFIG1[7] ENABLEENGINE bit is set, the FIFO entry on the L3 interconnect port side is accessible at any address in the associated chip-select memory region. When the ENABLEENGINE bit is set, any host access to this chip-select is rerouted to the FIFO input. Directly accessing the NAND device linked to this chip-select from the host is still possible through these registers:

- GPMC\_NAND\_COMMAND\_i
- GPMC\_NAND\_ADDRESS\_i
- GPMC\_NAND\_DATA\_i

The FIFO entry on the L3 interconnect port can be accessed with Byte, 16-bit word, or 32-bit word access size, according to little-endian format, even though the FIFO input is 32-bit wide.

The FIFO control is made easier through the use of interrupts or DMA requests associated with the FIFOThreshold bit field. The GPMC\_PREFETCH\_STATUS[30-24] FIFOPOINTER field monitors the number of available bytes to be read in prefetch mode or the number of free empty slots which can be written in write-posting mode. The GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE field monitors the number of remaining bytes to be read or written by the engine according to the TRANSFERCOUNT value. The FIFOPOINTER and COUNTVALUE bit fields are always expressed as a number of bytes even if a 16-bit wide NAND device is attached to the linked chip-select.

In prefetch mode, when the FIFOPOINTER equals 0, that is, the FIFO is empty, a host read access receives the byte last read from the FIFO as its response. In case of 32-bit word or 16-bit word read accesses, the last byte read from the FIFO is copied the required number of times to fit the requested word size. In write-posting mode, when the FIFOPOINTER equals 0, that is, the FIFO is full, a host write overwrites the last FIFO byte location. There is no underflow or overflow error reporting in the GPMC.

#### 7.1.3.3.12.4.2 Prefetch Mode

The prefetch mode is selected when the GPMC\_PREFETCH\_CONFIG1[0] ACCESSMODE bit is cleared.

The MPU NAND software driver must issue the block and page opening (READ) command with the correct data address pointer initialization before the engine can be started to read from the NAND memory device. The engine is started by asserting the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit. The STARTENGINE bit automatically clears when the prefetch process completes.

If required, the ECC calculator engine must be initialized (i.e., reset, configured, and enabled) before the prefetch engine is started, so that the ECC is correctly computed on all data read by the prefetch engine.

When the GPMC\_PREFETCH\_CONFIG1[3] SYNCHROMODE bit is cleared, the prefetch engine starts requesting data as soon as the STARTENGINE bit is set. If using this configuration, the host must monitor the NAND device-ready pin so that it only sets the STARTENGINE bit when the NAND device is in a ready state, meaning data is valid for prefetching.

When the SYNCHROMODE bit is set, the prefetch engine starts requesting data when an active to inactive wait signal transition is detected. The transition detector must be cleared before any transition detection; see [Section 7.1.3.3.12.2.2](#). The GPMC\_PREFETCH\_CONFIG1[5-4] WAITPINSELECTOR field selects which gpmc\_wait pin edge detector triggers the prefetch engine in this synchronized mode.

If the STARTENGINE bit is set after the NAND address phase (page opening command), the engine is effectively started only after the actual NAND address phase completion. To prevent GPMC stall during this NAND address phase, set the STARTENGINE bit field before NAND address phase completion when in synchronized mode. The prefetch engine will start when an active to inactive wait signal transition is detected. The STARTENGINE bit is automatically cleared on prefetch process completion.

The prefetch engine issues a read request to fill the FIFO with the amount of data specified by GPMC\_PREFETCH\_CONFIG2[13-0] TRANSFERCOUNT field.

[Table 7-22](#) describes the prefetch mode configuration.

**Table 7-22. Prefetch Mode Configuration**

| Bit Field             | Register              | Value   | Comments  |
|-----------------------|-----------------------|---------|---|
| STARTENGINE           | GPMC_PREFETCH_CONTROL | 0       | Prefetch engine can be configured only if STARTENGINE is cleared to 0.                      |
| ENGINECSSELECTOR      | GPMC_PREFETCH_CONFIG1 | 0 to 3h | Selects the chip-select associated with a NAND device where the prefetch engine is active.  |
| ACCESSMODE            | GPMC_PREFETCH_CONFIG1 | 0       | Selects prefetch mode   |
| FIFOTHRESHOLD         | GPMC_PREFETCH_CONFIG1 |         | Selects the maximum number of bytes read or written by the host on DMA or interrupt request |
| TRANSFERCOUNT         | GPMC_PREFETCH_CONFIG1 |         | Selects the number of bytes to be read or written by the engine to the selected chip-select |
| SYNCHROMODE           | GPMC_PREFETCH_CONFIG1 | 0/1     | Selects when the engine starts the access to the chip-select                                |
| WAITPINSELECT         | GPMC_PREFETCH_CONFIG1 | 0 to 1  | Selects wait pin edge detector (if GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE = 1)                |
| ENABLEOPTIMIZEDACCESS | GPMC_PREFETCH_CONFIG1 | 0/1     | See <a href="#">Section 7.1.3.3.12.4.6</a>  |
| CYCLOOPTIMIZATION     | GPMC_PREFETCH_CONFIG1 |         | Number of clock cycle removed to timing parameters  |
| ENABLEENGINE          | GPMC_PREFETCH_CONFIG1 | 1       | Engine enabled  |
| STARTENGINE           | GPMC_PREFETCH_CONFIG1 | 1       | Starts the prefetch engine  |

#### 7.1.3.3.12.4.3 FIFO Control in Prefetch Mode

The FIFO can be drained directly by the MPU or by an eDMA channel.

In MPU draining mode, the FIFO status can be monitored through the GPMC\_PREFETCH\_STATUS[30-24] FIFOPOINTER field or through the GPMC\_PREFETCH\_STATUS[16] FIFOThresholdStatus bit. The FIFOPOINTER indicates the current number of available data to be read; FIFOThresholdStatus set to 1 indicates that at least FIFOThreshold bytes are available from the FIFO.

An interrupt can be triggered by the GPMC if the GPMC\_IRQENABLE[0] FIFOEventEnable bit is set. The FIFO interrupt event is logged, and the GPMC\_IRQSTATUS[0] FIFOEventStatus bit is set. To clear the interrupt, the MPU must read all the available bytes, or at least enough bytes to get below the programmed FIFO threshold, and the FIFOEventStatus bit must be cleared to enable further interrupt events. The FIFOEventStatus bit must always be reset prior to asserting the FIFOEventEnable bit to clear any out-of-date logged interrupt event. This interrupt generation must be enabled after enabling the STARTENGINE bit.

Prefetch completion can be monitored through the GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE field. COUNTVALUE indicates the number of currently remaining data to be requested according to the TRANSFERCOUNT value. An interrupt can be triggered by the GPMC when the prefetch process is complete (that is, COUNTVALUE equals 0) if the GPMC\_IRQENABLE[1] TERMINALCOUNTEventEnable bit is set. At prefetch completion, the TERMINALCOUNT interrupt event is also logged, and the GPMC\_IRQSTATUS[1] TERMINALCOUNTStatus bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTStatus bit. The TERMINALCOUNTStatus bit must always be cleared prior to asserting the TERMINALCOUNTEventEnable bit to clear any out-of-date logged interrupt event.

---

**NOTE:** The COUNTVALUE value is only valid when the prefetch engine is active (started), and an interrupt is only triggered when COUNTVALUE reaches 0, that is, when the prefetch engine automatically goes from an active to an inactive state.

---

The number of bytes to be prefetched (programmed in TRANSFERCOUNT) must be a multiple of the programmed FIFOThreshold to trigger the correct number of interrupts allowing a deterministic and transparent FIFO control. If this guideline is respected, the number of ISR accesses is always required and the FIFO is always empty after the last interrupt is triggered. In other cases, the TERMINALCOUNT interrupt must be used to read the remaining bytes in the FIFO (the number of remaining bytes being lower than the FIFOThreshold value).

In DMA draining mode, the GPMC\_PREFETCH\_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOThreshold bytes are ready to be read from the FIFO. The DMA channel owning this DMA request must be programmed so that the number of bytes programmed in FIFOThreshold is read from the FIFO during the DMA request process. The DMA request is kept active until this number of bytes has effectively been read from the FIFO, and no other DMA request can be issued until the ongoing active request is complete.

In prefetch mode, the TERMINALCOUNT event is also a source of DMA requests if the number of bytes to be prefetched is not a multiple of FIFOThreshold, the remaining bytes in the FIFO can be read by the DMA channel using the last DMA request. This assumes that the number of remaining bytes to be read is known and controlled through the DMA channel programming model.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (the STARTENGINE bit is set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that the out-of-date active DMA request does not trigger spurious DMA transfers.

#### 7.1.3.3.12.4.4 Write-Posting Mode

The write-posting mode is selected when the GPMC\_PREFETCH\_CONFIG1[0] ACCESSMODE bit is set.

The MPU NAND software driver must issue the correct address pointer initialization command (page program) before the engine can start writing data into the NAND memory device. The engine starts when the GPMC\_PREFETCH\_CONTROL[0] STARTENGINE bit is set to 1. The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MPU NAND software driver must issue the second cycle program command and monitor the status for programming process completion (adding ECC handling, if required).

If used, the ECC calculator engine must be started (configured, reset, and enabled) before the posting engine is started so that the ECC parities are properly calculated on all data written by the prefetch engine to the associated chip-select.

In write-posting mode, the GPMC\_PREFETCH\_CONFIG1[3] SYNCHROMODE bit must be cleared so that posting starts as soon as the STARTENGINE bit is set and the FIFO is not empty.

If the STARTENGINE bit is set after the NAND address phase (page program command), the STARTENGINE setting is effective only after the actual NAND command completion. To prevent GPMC stall during this NAND command phase, set the STARTENGINE bit field before the NAND address completion and ensure that the associated DMA channel is enabled after the NAND address phase.

The posting engine issues a write request when valid data are available from the FIFO and until the programmed GPMC\_PREFETCH\_CONFIG2[13-0] TRANSFERCOUNT accesses have been completed.

The STARTENGINE bit clears automatically when posting completes. When all data have been written to the NAND memory device, the MPU NAND software driver must issue the second cycle program command and monitor the status for programming process completion. The closing program command phase must only be issued when the full NAND page has been written into the NAND flash write buffer, including the spare area data and the ECC parities, if used.

**Table 7-23. Write-Posting Mode Configuration**

| Bit Field             | Register              | Value   | Comments   |
|-----------------------|-----------------------|---------|--|
| STARTENGINE           | GPMC_PREFETCH_CONTROL | 0       | Write-posting engine can be configured only if STARTENGINE is cleared to 0.                      |
| ENGINECSSELECTOR      | GPMC_PREFETCH_CONFIG1 | 0 to 3h | Selects the chip-select associated with a NAND device where the prefetch engine is active        |
| ACCESSMODE            | GPMC_PREFETCH_CONFIG1 | 1       | Selects write-posting mode   |
| FIFOTHRESHOLD         | GPMC_PREFETCH_CONFIG1 |         | Selects the maximum number of bytes read or written by the host on DMA or interrupt request      |
| TRANSFERCOUNT         | GPMC_PREFETCH_CONFIG2 |         | Selects the number of bytes to be read or written by the engine from/to the selected chip-select |
| SYNCHROMODE           | GPMC_PREFETCH_CONFIG1 | 0       | Engine starts the access to chip-select as soon as STARTENGINE is set.                           |
| ENABLEOPTIMIZEDACCESS | GPMC_PREFETCH_CONFIG1 | 0/1     | See <a href="#">Section 7.1.3.3.12.4.6</a>   |
| CYCLEOPTIMIZATION     | GPMC_PREFETCH_CONFIG  |         |  |
| ENABLEENGINE          | GPMC_PREFETCH_CONFIG1 | 1       | Engine enabled   |
| STARTENGINE           | GPMC_PREFETCH_CONTROL | 1       | Starts the prefetch engine   |

#### 7.1.3.3.12.4.5 FIFO Control in Write-Posting Mode

The FIFO can be filled directly by the MPU or by an sDMA channel.

In MPU filling mode, the FIFO status can be monitored through the FIFOPOINTER or through the GPMC\_PREFETCH\_STATUS[16] FIFOTHRESHOLDSTATUS bit. FIFOPOINTER indicates the current number of available free byte places in the FIFO, and the FIFOTHRESHOLDSTATUS bit, when set, indicates that at least FIFOTHRESHOLD free byte places are available in the FIFO.

An interrupt can be issued by the GPMC if the GPMC\_IRQENABLE[0] FIFOEVENTENABLE bit is set. When the interrupt is fired, the GPMC\_IRQSTATUS[0] FIFOEVENTSTATUS bit is set. To clear the interrupt, the MPU must write enough bytes to fill the FIFO, or enough bytes to get below the programmed threshold, and the FIFOEVENTSTATUS bit must be cleared to get further interrupt events. The FIFOEVENTSTATUS bit must always be cleared prior to asserting the FIFOEVENTENABLE bit to clear any out-of-date logged interrupt event. This interrupt must be enabled after enabling the STARTENGINE bit.

The posting completion can be monitored through the GPMC\_PREFETCH\_STATUS[13-0] COUNTVALUE field. COUNTVALUE indicates the current number of remaining data to be written based on the TRANSFERCOUNT value. An interrupt is issued by the GPMC when the write-posting process completes (that is, COUNTVALUE equal to 0) if the GPMC\_IRQENABLE[1] TERMINALCOUNTEVENTENABLE bit is set. When the interrupt is fired, the GPMC\_IRQSTATUS[1] TERMINALCOUNTSTATUS bit is set. To clear the interrupt, the MPU must clear the TERMINALCOUNTSTATUS bit. The TERMINALCOUNTSTATUS bit must always be cleared prior to asserting the TERMINALCOUNTEVENTENABLE bit to clear any out-of-date logged interrupt event.

---

**NOTE:** The COUNTVALUE value is only valid if the write-posting engine is active and started, and an interrupt is only issued when COUNTVALUE reaches 0, that is, when the posting engine automatically goes from active to inactive.

---

In DMA filling mode, the DMAMode bit field in the GPMC\_PREFETCH\_CONFIG1[2] DMAMODE bit must be set so that the GPMC issues a DMA hardware request when at least FIFOTHRESHOLD bytes-free places are available in the FIFO. The DMA channel owning this DMA request must be programmed so that a number of bytes equal to the value programmed in the FIFOTHRESHOLD bit field are written into the FIFO during the DMA access. The DMA request remains active until the associated number of bytes has effectively been written into the FIFO, and no other DMA request can be issued until the ongoing active request has been completed.

Any potentially active DMA request is cleared when the prefetch engine goes from inactive to active prefetch (STARTENGINE set to 1). The associated DMA channel must always be enabled by the MPU after setting the STARTENGINE bit so that an out-of-date active DMA request does not trigger spurious DMA transfers.

In write-posting mode, the DMA or the MPU fill the FIFO with no consideration to the associated byte enables. Any byte stored in the FIFO is written into the memory device.

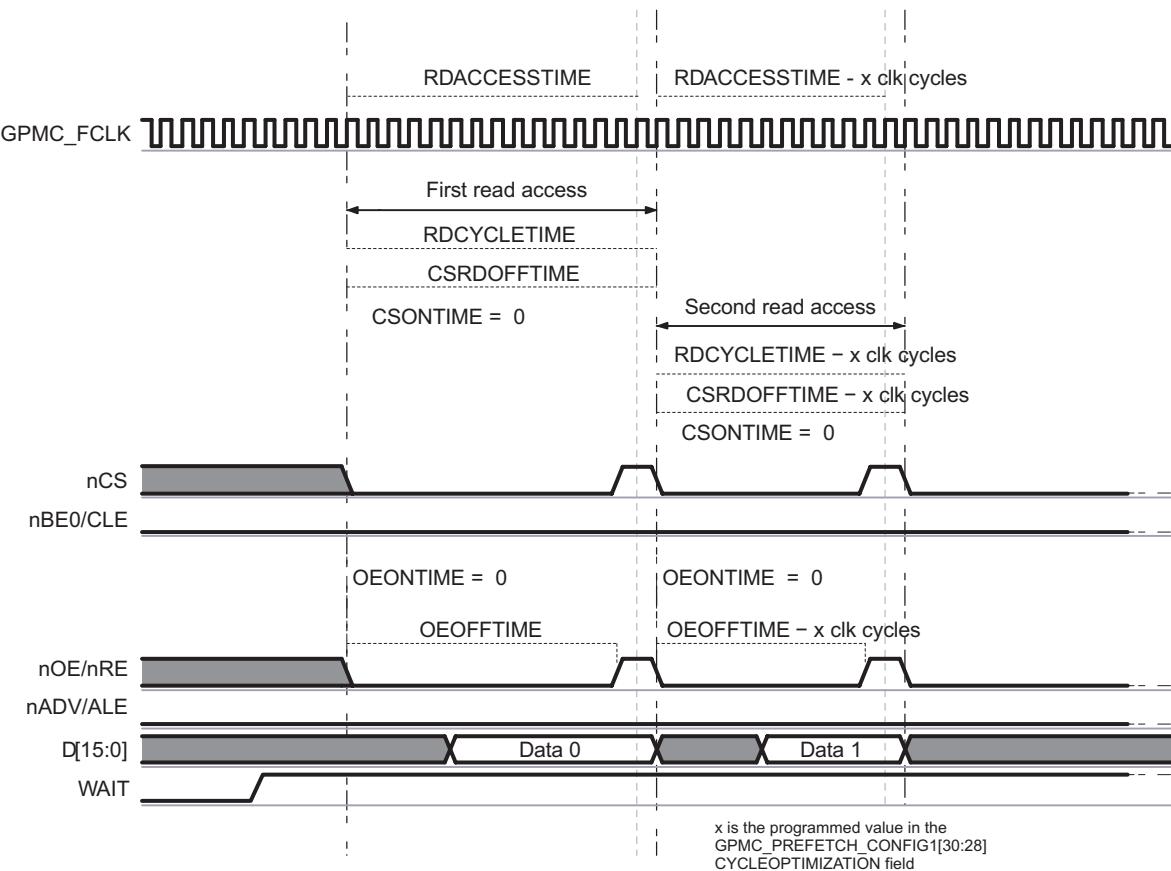
#### 7.1.3.3.12.4.6 Optimizing NAND Access Using the Prefetch and Write-Posting Engine

Access time to a NAND memory device can be optimized for back-to-back accesses if the associated CSn signal is not deasserted between accesses. The GPMC access engine can track prefetch engine accesses to optimize the access timing parameter programmed for the allocated chip-select, if no accesses to other chip-selects (that is, interleaved accesses) occur. Similarly, the access engine also eliminates the CYCLE2CYCLEDELAY even if CYCLE2CYCLESAMECSEN is set. This capability is limited to the prefetch and write-posting engine accesses, and MPU accesses to a NAND memory device (through the defined chip-select memory region or through the GPMC\_NAND\_DATA\_i) are never optimized.

The GPMC\_PREFETCH\_CONFIG1[27] ENABLEOPTIMIZEDACCESS bit must be set to enable optimized accesses. To optimize access time, the GPMC\_PREFETCH\_CONFIG1[30:28] CYCLEOPTIMIZATION field defines the number of GPMC\_FCLK cycles to be suppressed from the RDACYCLETIME, WRACYCLETIME, RDACCESSTIME, WRACCESSTIME, CSOFFTIME, ADVOFFTIME, OEOFETIME, and WEOFETIME timing parameters.

**Figure 7-41**, in the case of back-to-back accesses to the NAND flash through the prefetch engine, CYCLE2CYCLESAMECSEN is forced to 0 when using optimized accesses. The first access uses the regular timing settings for this chip-select. All accesses after this one use settings reduced by x clock cycles, x being defined by the GPMC\_PREFETCH\_CONFIG1[30:28] CYCLEOPTIMIZATION field.

**Figure 7-41. NAND Read Cycle Optimization Timing Description**



#### 7.1.3.3.12.4.7 Interleaved Accesses Between Prefetch and Write-Posting Engine and Other Chip-Selects

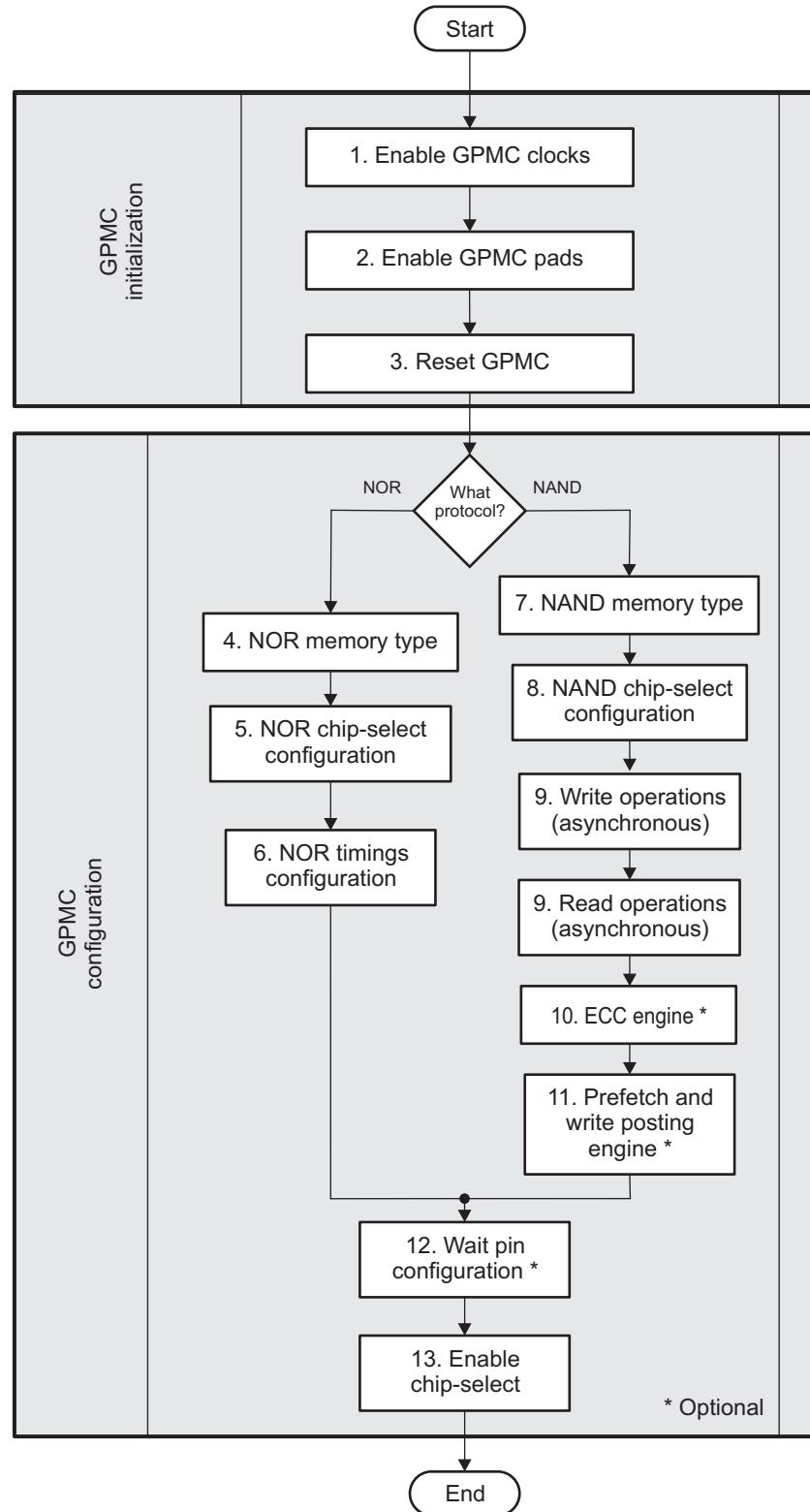
Any on-going read or write access from the prefetch and write-posting engine is completed before an access to any other chip-select can be initiated. As a default, the arbiter uses a fixed-priority algorithm, and the prefetch and write-posting engine has the lowest priority. The maximum latency added to access starting time in this case equals the RDCYCLETIME or WRCYCLETIME (optimized or not) plus the requested BUSTURNAROUND delay for bus turnaround completion programmed for the chip-select to which the NAND device is connected to.

Alternatively, a round-robin arbitration can be used to prioritize accesses to the external bus. This arbitration scheme is enabled by setting the GPMC\_PREFETCH\_CONFIG1[23] PFPWENROUNDROBIN bit. When a request to another chip-select is received while the prefetch and write-posting engine is active, priority is given to the new request. The request processed thereafter is the prefetch and write-posting engine request, even if another interconnect request is passed in the mean time. The engine keeps control of the bus for an additional number of requests programmed in the GPMC\_PREFETCH\_CONFIG1[19-16] PFPWWEIGHTEDPRIO bit field. Control is then passed to the direct interconnect request.

As an example, the round-robin arbitration scheme is selected with PFPWWEIGHTEDPRIO set to 2h. Considering the prefetch and write-posting engine and the interconnect interface are always requesting access to the external interface, the GPMC grants priority to the direct interconnect access for one request. The GPMC then grants priority to the engine for three requests, and finally back to the direct interconnect access, until the arbiter is reset when one of the two initiators stops initiating requests.

#### 7.1.3.4 GPMC High-Level Programming Model Overview

The high-level programming model introduces a top-down approach for users that need to configure the GPMC module. [Figure 7-42](#) shows a programming model top-level diagram for the GPMC. Each block of the diagram is described in one of the following subsections through a set of registers to configure. [Table 7-24](#) and [Table 7-25](#) list each step in the model.

**Figure 7-42. Programming Model Top-Level Diagram**


**Table 7-24. GPMC Configuration in NOR Mode**

| Step                          | Description                    |
|-------------------------------|--------------------------------|
| NOR Memory Type               | See <a href="#">Table 7-27</a> |
| NOR Chip-Select Configuration | See <a href="#">Table 7-28</a> |
| NOR Timings Configuration     | See <a href="#">Table 7-29</a> |
| Wait Pin Configuration        | See <a href="#">Table 7-30</a> |
| Enable Chip-Select            | See <a href="#">Table 7-31</a> |

**Table 7-25. GPMC Configuration in NAND Mode**

| Step                              | Description                    |
|-----------------------------------|--------------------------------|
| NAND Memory Type                  | See <a href="#">Table 7-32</a> |
| NAND Chip-Select Configuration    | See <a href="#">Table 7-33</a> |
| Write Operations (Asynchronous)   | See <a href="#">Table 7-34</a> |
| Read Operations (Asynchronous)    | See <a href="#">Table 7-34</a> |
| ECC Engine                        | See <a href="#">Table 7-35</a> |
| Prefetch and Write-Posting Engine | See <a href="#">Table 7-36</a> |
| Wait Pin Configuration            | See <a href="#">Table 7-37</a> |
| Enable Chip-Select                | See <a href="#">Table 7-38</a> |

### 7.1.3.5 GPMC Initialization

[Table 7-26](#) describes the settings required to reset the GPMC.

**Table 7-26. Reset GPMC**

| Sub-process Name       | Register / Bitfield         | Value |
|------------------------|-----------------------------|-------|
| Start a software reset | GPMC_SYSCONFIG[1] SOFTRESET | 1     |
| Wait until             | GPMC_SYSSTATUS[0] RESETDONE | 1     |

### 7.1.3.6 GPMC Configuration in NOR Mode

This section gives a generic configuration for parameters related to the NOR memory connected to the GPMC. [Table 7-27](#) through [Table 7-31](#) list the steps to configure the GPMC in NOR mode.

**NOTE:** In the tables of this section, 'x' in Value column stands for 'depends on configuration'.

**Table 7-27. NOR Memory Type**

| Sub-process Name  | Register / Bitfield                               | Value |
|---|---|-------|
| Set the NOR protocol  | GPMC_CONFIG1_i[11-10] DEVICETYPE                  | 0     |
| Set a device size   | GPMC_CONFIG1_i[13-12] DEVICESIZE                  | x     |
| Select an address and data multiplexing protocol            | GPMC_CONFIG1_i[9] MUXADDDATA                      | x     |
| Set the attached device page length                         | GPMC_CONFIG1_i[24-23]<br>ATTACHEDDEVICEPAGELENGTH | x     |
| Set the wrapping burst capabilities                         | GPMC_CONFIG1_i[31] WRAPBURST                      | x     |
| Select a timing signals latencies factor                    | GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY             | x     |
| Select an output clock frequency                            | GPMC_CONFIG1_i[1-0] GPMCFCLKDIVIDER               | x     |
| Choose an output clock activation time                      | GPMC_CONFIG1_i[26-25] CLKACTIVATIONTIME           | x     |
| Set a single or multiple access for read operations         | GPMC_CONFIG1_i[30] READMULTIPLE                   | x     |
| Set a synchronous or asynchronous mode for read operations  | GPMC_CONFIG1_i[29] READTYPE                       | x     |
| Set a single or multiple access for write operations        | GPMC_CONFIG1_i[28] WRITEMULTIPLE                  | x     |
| Set a synchronous or asynchronous mode for write operations | GPMC_CONFIG1_i[27] WRITETYPE                      | x     |

**Table 7-28. NOR Chip-Select Configuration**

| Sub-process Name                    | Register / Bitfield              | Value |
|-------------------------------------|----------------------------------|-------|
| Select the chip-select base address | GPMC_CONFIG7_i[5-0] BASEADDRESS  | x     |
| Select the chip-select mask address | GPMC_CONFIG7_i[11-8] MASKADDRESS | x     |

**Table 7-29. NOR Timings Configuration**

| Sub-process Name   | Register / Bitfield                 | Value |
|--|-------------------------------------|-------|
| Configure adequate timing parameters in various memory modes | See <a href="#">Section 7.1.3.9</a> |       |

**Table 7-30. WAIT Pin Configuration**

| Sub-process Name   | Register / Bitfield                         | Value |
|--|---|-------|
| Enable or disable wait pin monitoring for read operations  | GPMC_CONFIG1_i[22] WAITREADMONITORING       | x     |
| Enable or disable wait pin monitoring for write operations | GPMC_CONFIG1_i[21] WAITWRITEMONITORING      | x     |
| Select a wait pin monitoring time                          | GPMC_CONFIG1_i[19-18]<br>WAITMONITORINGTIME | x     |
| Choose the input wait pin for the chip-select              | GPMC_CONFIG1_i[17-16] WAITPINSELECT         | x     |

**Table 7-31. Enable Chip-Select**

| Sub-process Name   | Register / Bitfield       | Value |
|--|---------------------------|-------|
| When all parameters are configured, enable the chip-select | GPMC_CONFIG7_i[6] CSVALID | x     |

### 7.1.3.7 GPMC Configuration in NAND Mode

This section gives a generic configuration for parameters related to NAND memory connected to the GPMC.

**Table 7-32. NAND Memory Type**

| Sub-process Name   | Register / Bitfield                   | Value |
|--|---------------------------------------|-------|
| Set the NAND protocol  | GPMC_CONFIG1_i[11-10] DEVICETYPE      | 2h    |
| Set a device size  | GPMC_CONFIG1_i[13-12] DEVICESIZE      | x     |
| Set the address and data multiplexing protocol to non-multiplexed attached device                    | GPMC_CONFIG1_i[9] MUXADDDATA          | 0     |
| Select a timing signals latencies factor   | GPMC_CONFIG1_i[4] TIMEPARAGRANULARITY | x     |
| Set a synchronous or asynchronous mode and a single or multiple access for read and write operations | See <a href="#">Section 7.1.3.8</a>   | x     |

**Table 7-33. NAND Chip-Select Configuration**

| Sub-process Name                                       | Register / Bitfield              | Value |
|--|----------------------------------|-------|
| Select the chip-select base address                    | GPMC_CONFIG7_i[5-0] BASEADDRESS  | x     |
| Select the chip-select minimum granularity (16M bytes) | GPMC_CONFIG7_i[11-8] MASKADDRESS | x     |

**Table 7-34. Asynchronous Read and Write Operations**

| Sub-process Name   | Register / Bitfield                 | Value |
|--|-------------------------------------|-------|
| Configure adequate timing parameters in asynchronous modes | See <a href="#">Section 7.1.3.9</a> |       |

**Table 7-35. ECC Engine**

| Sub-process Name  | Register / Bitfield   | Value            |
|---|---|------------------|
| Select the ECC result register where the first ECC computation is stored (Only applies to Hamming)    | GPMC_ECC_CONTROL[3-0] ECCPOINTER  | x                |
| Clear all ECC result registers  | GPMC_ECC_CONTROL[8] ECCCLEAR  | Write 1 to clear |
| Define ECCSIZE0 and ECCSIZE1  | GPMC_ECC_SIZE_CONFIG[19-12] ECCSIZE0 and GPMC_ECC_SIZE_CONFIG[29-22] ECCSIZE1     | x                |
| Select the size of each of the 9 result registers (size specified by ECCSIZE0 or ECCSIZE1)            | GPMC_ECC_SIZE_CONFIG[j-1] ECCjRESULTSIZE where j = 1 to 9                         | x                |
| Select the chip-select where ECC is computed  | GPMC_ECC_SIZE_CONFIG[3-1] ECCCS   | x                |
| Select the Hamming code or BCH code ECC algorithm in use  | GPMC_ECC_SIZE_CONFIG[16] ECCALGORITHM   | x                |
| Select word size for ECC calculation  | GPMC_ECC_SIZE_CONFIG[7] ECC16B  | x                |
| If the BCH code is used, Set an error correction capability and Select a number of sectors to process | GPMC_ECC_SIZE_CONFIG[13-12] ECCBCHTSEL and GPMC_ECC_SIZE_CONFIG[6-4] ECCTOPSECTOR | x                |

**Table 7-35. ECC Engine (continued)**

| Sub-process Name           | Register / Bitfield               | Value |
|----------------------------|-----------------------------------|-------|
| Enable the ECC computation | GPMC_ECC_SIZE_CONFIG[0] ECCENABLE | 1     |

**Table 7-36. Prefetch and Write-Posting Engine**

| <b>Sub-process Name</b>  | <b>Register / Bitfield</b>                        | <b>Value</b> |
|--|---|--------------|
| Disable the engine before configuration  | GPMC_PREFETCH_CONTROL[0] STARTENGINE              | 0            |
| Select the chip-select associated with a NAND device where the prefetch engine is active   | GPMC_PREFETCH_CONFIG1[26-24]<br>ENGINECSSELECTOR  | x            |
| Select access direction through prefetch engine, read or write.  | GPMC_PREFETCH_CONFIG1[0] ACCESSMODE               | x            |
| Select the threshold used to issue a DMA request   | GPMC_PREFETCH_CONFIG1[14-8]<br>FIFOThreshold      | x            |
| Select either DMA synchronized mode or SW manual mode.   | GPMC_PREFETCH_CONFIG1[2] DMAMODE                  | x            |
| Select if the engine immediately starts accessing the memory upon STARTENGINE assertion or if hardware synchronization based on a WAIT signal is used. | GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE              | x            |
| Select which wait pin edge detector should start the engine in synchronized mode   | GPMC_PREFETCH_CONFIG1[5-4]<br>WAITPINSELECTOR     | x            |
| Enter a number of clock cycles removed to timing parameters (For all back-to-back accesses to the NAND flash but not the first one)                    | GPMC_PREFETCH_CONFIG1[30-28]<br>CYCLEOPTIMIZATION | x            |
| Enable the prefetch postwrite engine   | GPMC_PREFETCH_CONFIG1[7] ENABLEENGINE             | 1            |
| Select the number of bytes to be read or written by the engine to the selected chip-select   | GPMC_PREFETCH_CONFIG2[13-0]<br>TRANSFERCOUNT      | x            |
| Start the prefetch engine  | GPMC_PREFETCH_CONTROL[0] STARTENGINE              | 1            |

**Table 7-37. WAIT Pin Configuration**

| <b>Sub-process Name</b>  | <b>Register / Bitfield</b>                    | <b>Value</b> |
|--|---|--------------|
| Selects when the engine starts the access to CS                                  | GPMC_PREFETCH_CONFIG1[3] SYNCHROMODE          | x            |
| Select which wait pin edge detector should start the engine in synchronized mode | GPMC_PREFETCH_CONFIG1[5-4]<br>WAITPINSELECTOR | x            |

**Table 7-38. Enable Chip-Select**

| <b>Sub-process Name</b>                                    | <b>Register / Bitfield</b> | <b>Value</b> |
|--|----------------------------|--------------|
| When all parameters are configured, enable the chip-select | GPMC_CONFIG7_i[6] CSVALID  | x            |

### 7.1.3.8 Set Memory Access

This section details the bit field to configure to set the GPMC in various memory modes.

**Table 7-39. Mode Parameters Check List Table**

| Register       | Bit | Bit Field Name | Asynchronous       |                     |                             |                              | Synchronous        |                     |                              |                               |
|----------------|-----|----------------|--------------------|---------------------|-----------------------------|------------------------------|--------------------|---------------------|------------------------------|-------------------------------|
|                |     |                | Single Read Access | Single Write Access | Multiple Read (Page) Access | Multiple Write (Page) Access | Single Read Access | Single Write Access | Multiple Read (Burst) Access | Multiple Write (Burst) Access |
| GPMC_CONFIG1_i | 30  | READMULTIPLE   | 0                  | -                   | 1                           | N/S                          | 0                  | -                   | 1                            | -                             |
| GPMC_CONFIG1_i | 29  | READYTYPE      | 0                  | -                   | 0                           | N/S                          | 1                  | -                   | 1                            | -                             |
| GPMC_CONFIG1_i | 28  | WRITEMULTIPLE  | -                  | 0                   |                             | N/S                          | -                  | 0                   | -                            | 1                             |
| GPMC_CONFIG1_i | 27  | WRITETYPE      | -                  | 0                   |                             | N/S                          | -                  | 1                   | -                            | 1                             |

**Table 7-40. Access Type Parameters Check List Table**

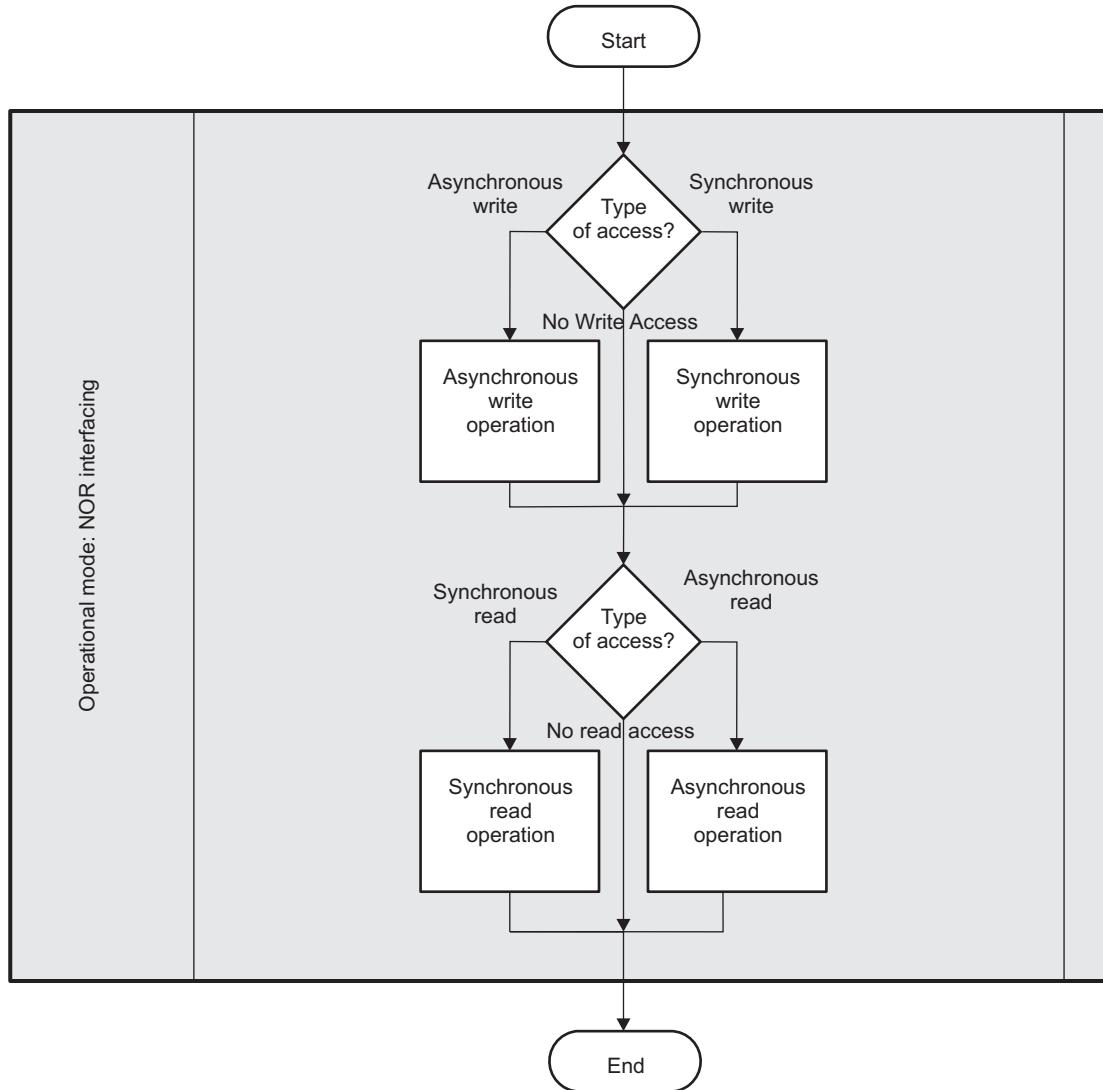
| Register       | Bit | Bit Field Name | Access Type |                  |         |
|----------------|-----|----------------|-------------|------------------|---------|
|                |     |                | Non-Mux     | Address/Data Mux | AAD Mux |
| GPMC_CONFIG1_i | 9-8 | MUXADDATA      | 0           | 2h               | 1       |

### 7.1.3.9 GPMC Timing Parameters

Figure 7-43 shows a programming model diagram for the NOR interfacing timing parameters.

Table 7-41 lists bit fields to configure adequate timing parameter in various memory modes.

**Figure 7-43. NOR Interfacing Timing Parameters Diagram**



**Table 7-41. Timing Parameters**

| Register       | Bit   | Bit Field Name      | Asynchronous       |                     |                             | Synchronous        |                     |                              |                               | Access Type     |                           |                 |
|----------------|-------|---------------------|--------------------|---------------------|-----------------------------|--------------------|---------------------|------------------------------|-------------------------------|-----------------|---------------------------|-----------------|
|                |       |                     | Single Read Access | Single Write Access | Multiple Read (Page) access | Single Read Access | Single Write Access | Multiple Read (Burst) Access | Multiple Write (Burst) Access | Non-multiplexed | Address /Data-multiplexed | AAD-multiplexed |
| GPMC_CONFIG1_i | 9-8   | MUXADDDATA          | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 29    | READTYPE            | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG1_i | 30    | READMULTIPLE        | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG1_i | 27    | WRITETYPE           |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 28    | WRITEMULTIPLE       |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 31    | WRAPBURST           |                    |                     |                             |                    |                     | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 26-25 | CLKACTIVATIONTIME   |                    |                     |                             | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 19-18 | WAITMONITORINGTIME  | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG1_i | 4     | TIMEPARAGRANULARITY | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG2_i | 20-16 | CSWROFFTIME         |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG2_i | 12-8  | CSRDOFFTIME         | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG2_i | 7     | CSEXTRADELAY        | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG2_i | 3-0   | CSONTIME            | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG3_i | 30-28 | ADVAADMUXWROFFTIME  |                    | y                   |                             |                    | y                   |                              | y                             |                 |                           | y               |
| GPMC_CONFIG3_i | 26-24 | ADVAADMUXRDOFFTIME  | y                  |                     | y                           | y                  |                     | y                            |                               |                 |                           | y               |
| GPMC_CONFIG3_i | 6-4   | ADVAADMUXONTIME     | y                  | y                   | y                           | y                  | y                   | y                            | y                             |                 |                           | y               |
| GPMC_CONFIG3_i | 20-16 | ADVWROFFTIME        |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG3_i | 12-8  | ADVRDOFFTIME        | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG3_i | 7     | ADVEXTRADELAY       | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG3_i | 3-0   | ADVONTIME           | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG4_i | 15-13 | OEAADMUXOFFTIME     | y                  | y                   | y                           | y                  | y                   | y                            | y                             |                 |                           | y               |
| GPMC_CONFIG4_i | 6-4   | OEAADMUXONTIME      | y                  | y                   | y                           | y                  | y                   | y                            | y                             |                 |                           | y               |
| GPMC_CONFIG4_i | 28-24 | WEOFFTIME           |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG4_i | 23    | WEEXTRADELAY        |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG4_i | 19-16 | WEONTIME            |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG4_i | 12-8  | OEOFETIME           | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG4_i | 7     | OEEEXTRADELAY       | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG4_i | 3-0   | OEONTIME            | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG5_i | 27-24 | PAGEBURSTACCESSTIME |                    |                     | y                           |                    |                     | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG5_i | 20-16 | RDACCESSTIME        | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG5_i | 12-8  | WRCYCLETIME         |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG5_i | 4-0   | RDCYCLETIME         | y                  |                     | y                           | y                  |                     | y                            |                               | y               | y                         | y               |
| GPMC_CONFIG6_i | 28-24 | WRACCESSTIME        |                    | y                   |                             |                    | y                   |                              | y                             | y               | y                         | y               |
| GPMC_CONFIG6_i | 19-16 | WRDATAONADMUXBUS    |                    | y                   |                             |                    | y                   |                              | y                             |                 | y                         | y               |

Table 7-41. Timing Parameters (continued)

| Register       | Bit  | Bit Field Name      | Asynchronous       |                     |                             | Synchronous        |                     |                              |                               | Access Type     |                           |                 |
|----------------|------|---------------------|--------------------|---------------------|-----------------------------|--------------------|---------------------|------------------------------|-------------------------------|-----------------|---------------------------|-----------------|
|                |      |                     | Single Read Access | Single Write Access | Multiple Read (Page) access | Single Read Access | Single Write Access | Multiple Read (Burst) Access | Multiple Write (Burst) Access | Non-multiplexed | Address /Data-multiplexed | AAD-multiplexed |
| GPMC_CONFIG6_i | 11-8 | CYCLE2CYCLEDELAY    | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG6_i | 7    | CYCLE2CYCLESAMECSEN | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG6_i | 6    | CYCLE2CYCLEIFFCSEN  | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG6_i | 3-0  | BUSTURNAROUND       | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |
| GPMC_CONFIG7_i | 6    | CSVALID             | y                  | y                   | y                           | y                  | y                   | y                            | y                             | y               | y                         | y               |

### 7.1.3.9.1 GPMC Timing Parameters Formulas

This section intends to help the user to calculate the GPMC timing bit fields values. Formulas are not listed exhaustively.

The section details:

- NAND Flash Interface Timing Parameters Formulas
- Synchronous NOR Flash Timing Parameters Formulas
- Asynchronous NOR Flash Timing Parameters Formulas

#### 7.1.3.9.1.1 NAND Flash Interface Timing Parameters Formulas

This section lists formulas to use in order to calculate NAND timing parameters. This is the case when GPMC\_CONFIG1\_i[11:10] DEVICETYPE = 2h. [Table 7-42](#) details NAND timing parameters.

**Table 7-42. NAND Formulas Description Table**

| Configuration Parameter | Unit | Description  |
|-------------------------|------|--|
| A                       | ns   | Pulse duration - GPMC_WEn valid time                                 |
| B                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_WEn valid            |
| C                       | ns   | Delay time - GPMC_BE0n_CLE/GPMC_ADVn_ALE high to GPMC_WEn valid      |
| D                       | ns   | Delay time - GPMC_AD[15:0] valid to GPMC_WEn valid                   |
| E                       | ns   | Delay time - GPMC_WEn invalid to GPMC_AD[15:0] invalid               |
| F                       | ns   | Delay time - GPMC_WEn invalid to GPMC_BE0n_CLE/GPMC_ADVn_ALE invalid |
| G                       | ns   | Delay time - GPMC_WEn invalid to GPMC_CS <sub>n</sub> invalid        |
| H                       | ns   | Cycle time - Write cycle time  |
| I                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_OEn valid            |
| J                       | ns   | Setup time - GPMC_AD[15:0] valid to GPMC_OEn invalid                 |
| K                       | ns   | Pulse duration - GPMC_OEn valid time                                 |
| L                       | ns   | Cycle time - Read cycle time   |
| M                       | ns   | Delay time - GPMC_OEn invalid to GPMC_CS <sub>n</sub> invalid        |

The configuration parameters are calculated through the following formulas.

$$A = (WEOFFTime - WEONTime) * (TimeParaGranularity + 1) * GPMC_FCLK \text{ period}$$

$$B = ((WEONTime - CSOnTime) * (TimeParaGranularity + 1) + 0.5 * (WEExtraDelay - CSEExtraDelay)) * GPMC_FCLK \text{ period}$$

$$C = ((WEONTime - ADVOnTime) * (TimeParaGranularity + 1) + 0.5 * (WEExtraDelay - ADVExtraDelay)) * GPMC_FCLK \text{ period}$$

$$D = (WEONTime * (TimeParaGranularity + 1) + 0.5 * WEExtraDelay) * GPMC_FCLK \text{ period}$$

$$E = (WrCycleTime - WEOFFTime * (TimeParaGranularity + 1) - 0.5 * WEExtraDelay) * GPMC_FCLK \text{ period}$$

$$F = (ADVWrOffTime - WEOFFTime * (TimeParaGranularity + 1) + 0.5 * (ADVExtraDelay - WEExtraDelay)) * GPMC_FCLK \text{ period}$$

$$G = (CSWrOffTime - WEOFFTime * (TimeParaGranularity + 1) + 0.5 * (CSEExtraDelay - WEExtraDelay)) * GPMC_FCLK \text{ period}$$

$$H = WrCycleTime * (1 + TimeParaGranularity) * GPMC_FCLK \text{ period}$$

$$I = ((OEONTime - CSOnTime) * (TimeParaGranularity + 1) + 0.5 * (OEExtraDelay - CSEExtraDelay)) * GPMC_FCLK \text{ period}$$

$$J = ((AccessTime - OEOFFTime) * (TimeParaGranularity + 1) - 0.5 * OEExtraDelay) * GPMC_FCLK \text{ period}$$

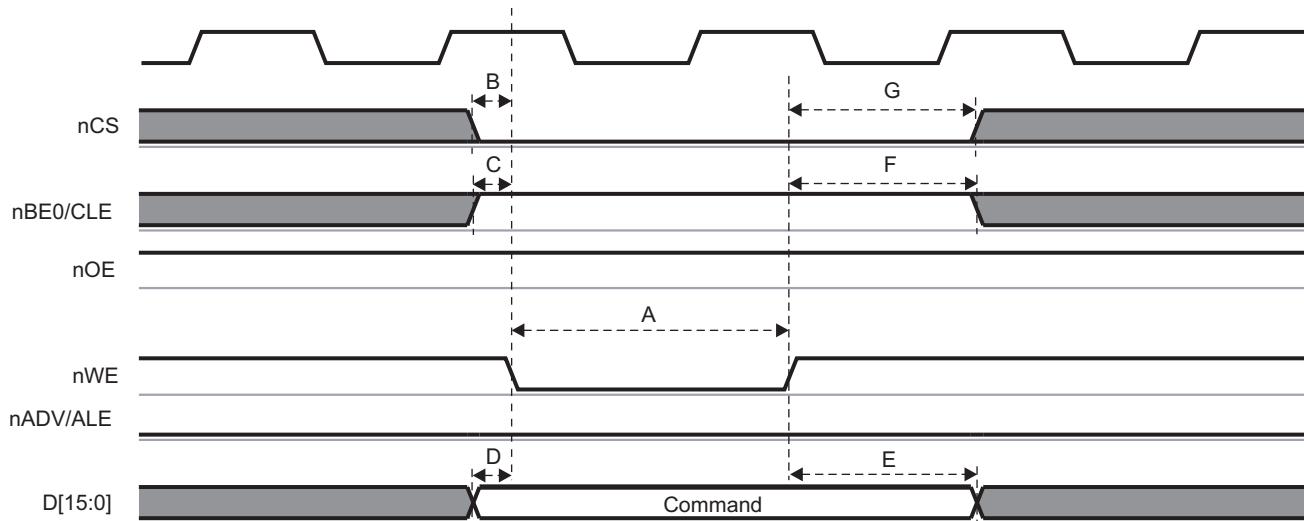
$$K = (\text{OEOffTime} - \text{OEOnTime}) * (1 + \text{TimeParaGranularity}) * \text{GPMC_FCLK period}$$

$$L = \text{RdCycleTime} * (1 + \text{TimeParaGranularity}) * \text{GPMC_FCLK period}$$

$$M = (\text{CSRdOffTime} - \text{OEOffTime} * (\text{TimeParaGranularity} + 1) + 0.5 * (\text{CSEExtraDelay} - \text{OEEExtraDelay}) * \text{GPMC_FCLK period}$$

**Figure 7-44** shows a command latch cycle timing simplified example where formulas are associated with signal waves.

**Figure 7-44. NAND Command Latch Cycle Timing Simplified Example**



#### 7.1.3.9.1.2 Synchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to use in order to calculate synchronous NOR timing parameters. This is the case when GPMC\_CONFIG1\_i[11-10] DEVICETYPE = 0 and when READTYPE or WRITETYPE are set to synchronous mode.

**Table 7-43. Synchronous NOR Formulas Description Table**

| Configuration Parameter | Unit | Description   |
|-------------------------|------|---|
| A                       | ns   | Pulse duration - GPMC_CS <sub>n</sub> low                               |
| B                       | ns   | Delay time - address bus valid to GPMC_CLK first edge                   |
|                         |      | Delay time - GPMC_BE0n_CLE/GPMC_BE1n valid to GPMC_CLK first edge       |
| C                       | ns   | Pulse duration - GPMC_BE0n_CLE/GPMC_BE1n low                            |
| D                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_BE0n_CLE/GPMC_BE1n invalid    |
|                         |      | Delay time - GPMC_CLK rising edge to GPMC_ADVn_ALE invalid              |
| E                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_CS <sub>n</sub> invalid       |
|                         |      | Delay time - GPMC_CLK rising edge to GPMC_OEn invalid                   |
| F                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_CS <sub>n</sub> transition    |
| G                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_ADVn_ALE transition           |
| H                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_OEn transition                |
| I                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_WEn transition                |
| J                       | ns   | Delay time - GPMC_CLK rising edge to GPMC_AD data bus transition        |
|                         |      | Delay time - GPMC_CLK rising edge to GPMC_BE0n_CLE/GPMC_BE1n transition |
| K                       | ns   | Pulse duration - GPMC_ADVn_ALE low                                      |
| L                       | ns   | Delay time - GPMC_WAIT invalid to first data latching GPMC_CLK edge     |

The configuration parameters are calculated through the following formulas.

For single read accesses:

$$\begin{aligned} A &= (\text{CSRDOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ C &= \text{RDCYCLETIME} * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ D &= (\text{RDCYCLETIME} - \text{ACCESSTIME}) * \text{GPMC\_FCLK period} \\ E &= (\text{CSRDOFFTIME} - \text{ACCESSTIME}) * \text{GPMC\_FCLK period} \end{aligned}$$

For burst read accesses (where n is the page burst access number):

$$\begin{aligned} A &= (\text{CSRDOFFTIME} - \text{CSONTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ C &= (\text{RDCYCLETIME} + (n - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ D &= (\text{RDCYCLETIME} - (\text{ACCESSTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME})) * \text{GPMC\_FCLK period} \\ E &= (\text{CSRDOFFTIME} - (\text{ACCESSTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME})) * \text{GPMC\_FCLK period} \end{aligned}$$

For burst write accesses (where n is the page burst access number):

$$\begin{aligned} A &= (\text{CSWROFFTIME} - \text{CSONTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ C &= (\text{WRCYCLETIME} + (n - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period} \\ D &= (\text{WRCYCLETIME} - (\text{ACCESSTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME})) * \text{GPMC\_FCLK period} \\ E &= (\text{CSWROFFTIME} - (\text{ACCESSTIME} + (n - 1) * \text{PAGEBURSTACCESSTIME})) * \text{GPMC\_FCLK period} \end{aligned}$$

For all accesses:

For CSn falling edge (CSn activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$ , when (CLKACTIVATIONTIME and CSONTIME are odd) or (CLKACTIVATIONTIME and CSRDOFFTIME are even)  
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$  otherwise
- Case where GPMCFCLKDIVIDER = 0x2  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$ , when (CSONTIME - CLKACTIVATIONTIME) is a multiple of 3  
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$ , when (CSONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3  
 $F = (2 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$ , when (CSONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For CSn rising edge (CSn de-activated) in reading mode:

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$ , when (CLKACTIVATIONTIME and CSRDOFFTIME are odd) or (CLKACTIVATIONTIME and CSONTIME are even)  
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$  otherwise
- Case where GPMCFCLKDIVIDER = 0x2  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC\_FCLK period}$ , when (CSRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3  
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$ , when (CSRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3  
 $F = (2 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC\_FCLK period}$ , when (CSRDOFFTIME -

CLKACTIVATIONTIME - 2) is a multiple of 3

For CSn rising edge (CSn de-activated) in writing mode:

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC_FCLK period, when (CLKACTIVATIONTIME and CSWROFFTIME are odd) or (CLKACTIVATIONTIME and CSWROFFTIME are even)}$   
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC_FCLK period otherwise}$
- Case where GPMCFCLKDIVIDER = 0x2  
 $F = 0.5 * \text{CSEXTRADELAY} * \text{GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME) is a multiple of 3}$   
 $F = (1 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$   
 $F = (2 + 0.5 * \text{CSEXTRADELAY}) * \text{GPMC_FCLK period, when (CSWROFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$

For ADVn falling edge (ADVn activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVONTIME are odd) or (CLKACTIVATIONTIME and ADVONTIME are even)}$   
 $G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period otherwise}$
- Case where GPMCFCLKDIVIDER = 0x2  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME) is a multiple of 3}$   
 $G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$   
 $G = (2 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period, when (ADVONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$

For ADVn rising edge (ADVn de-activated) in reading mode:

- Case where [1-0] GPMCFCLKDIVIDER = 0x0  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period, when (CLKACTIVATIONTIME and ADVRDOFFTIME are odd) or (CLKACTIVATIONTIME and ADVRDOFFTIME are even)}$   
 $G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period otherwise}$
- Case where GPMCFCLKDIVIDER = 0x2  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME) is a multiple of 3}$   
 $G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$   
 $G = (2 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period, when (ADVRDOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$

For ADVn rising edge (ADVn de-activated) in writing mode:

- Case where [1-0] GPMCFCLKDIVIDER = 0x0  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC_FCLK period, when (CLKACTIVATIONTIME and CSWROFFTIME are odd) or (CLKACTIVATIONTIME and CSWROFFTIME are even)}$   
 $G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC_FCLK period otherwise}$

ADWROFFTIME are odd) or (CLKACTIVATIONTIME and ADWROFFTIME are even)

$G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC\_FCLK}$  period otherwise

- Case where GPMCFCLKDIVIDER = 0x2

$G = 0.5 * \text{ADVEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (ADWROFFTIME - CLKACTIVATIONTIME) is a multiple of 3

$G = (1 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (ADWROFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

$G = (2 + 0.5 * \text{ADVEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (ADWROFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For OEn falling edge (OEn activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period

- Case where GPMCFCLKDIVIDER = 0x1

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (CLKACTIVATIONTIME and OEONTIME are odd) or (CLKACTIVATIONTIME and OEONTIME are even)

$H = (1 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period otherwise

- Case where GPMCFCLKDIVIDER = 0x2

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (OEONTIME - CLKACTIVATIONTIME) is a multiple of 3

$H = (1 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (OEONTIME - CLKACTIVATIONTIME - 1) is a multiple of 3

$H = (2 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (OEONTIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For OEn rising edge (OEn de-activated):

- Case where [1-0] GPMCFCLKDIVIDER = 0x0

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period

- Case where GPMCFCLKDIVIDER = 0x1

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (CLKACTIVATIONTIME and OEOFETIME are odd) or (CLKACTIVATIONTIME and OEOFETIME are even)

$H = (1 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period otherwise

- Case where GPMCFCLKDIVIDER = 0x2

$H = 0.5 * \text{OEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (OEOFETIME - CLKACTIVATIONTIME) is a multiple of 3

$H = (1 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (OEOFETIME - CLKACTIVATIONTIME - 1) is a multiple of 3

$H = (2 + 0.5 * \text{OEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (OEOFETIME - CLKACTIVATIONTIME - 2) is a multiple of 3

For WEn falling edge (WEn activated):

- Case where GPMC\_CONFIG1\_i[1-0] GPMCFCLKDIVIDER = 0x0

$I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK}$  period

- Case where GPMCFCLKDIVIDER = 0x1

$I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (CLKACTIVATIONTIME and WEONTIME are odd) or (CLKACTIVATIONTIME and WEONTIME are even)

$I = (1 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK}$  period otherwise

- Case where GPMCFCLKDIVIDER = 0x2

$I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK}$  period, when (WEONTIME - CLKACTIVATIONTIME) is a multiple of 3

$I = (1 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when (WEONTIME - CLKACTIVATIONTIME -

1) is a multiple of 3

$I = (2 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK}$  period, when  $(\text{WEONTIME} - \text{CLKACTIVATIONTIME} - 2)$  is a multiple of 3

For WEn rising edge (WEn de-activated):

- Case where [1-0] GPMCFCLKDIVIDER = 0x0  
 $I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK period}$
- Case where GPMCFCLKDIVIDER = 0x1  
 $I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK period, when (CLKACTIVATIONTIME and WEOFFTIME are odd) or (CLKACTIVATIONTIME and WEOFFTIME are even)}$   
 $I = (1 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK period otherwise}$
- Case where GPMCFCLKDIVIDER = 0x2  
 $I = 0.5 * \text{WEEXTRADELAY} * \text{GPMC\_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME) is a multiple of 3}$   
 $I = (1 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME - 1) is a multiple of 3}$   
 $I = (2 + 0.5 * \text{WEEXTRADELAY}) * \text{GPMC\_FCLK period, when (WEOFFTIME - CLKACTIVATIONTIME - 2) is a multiple of 3}$

For GPMC\_ADVn low pulse duration:

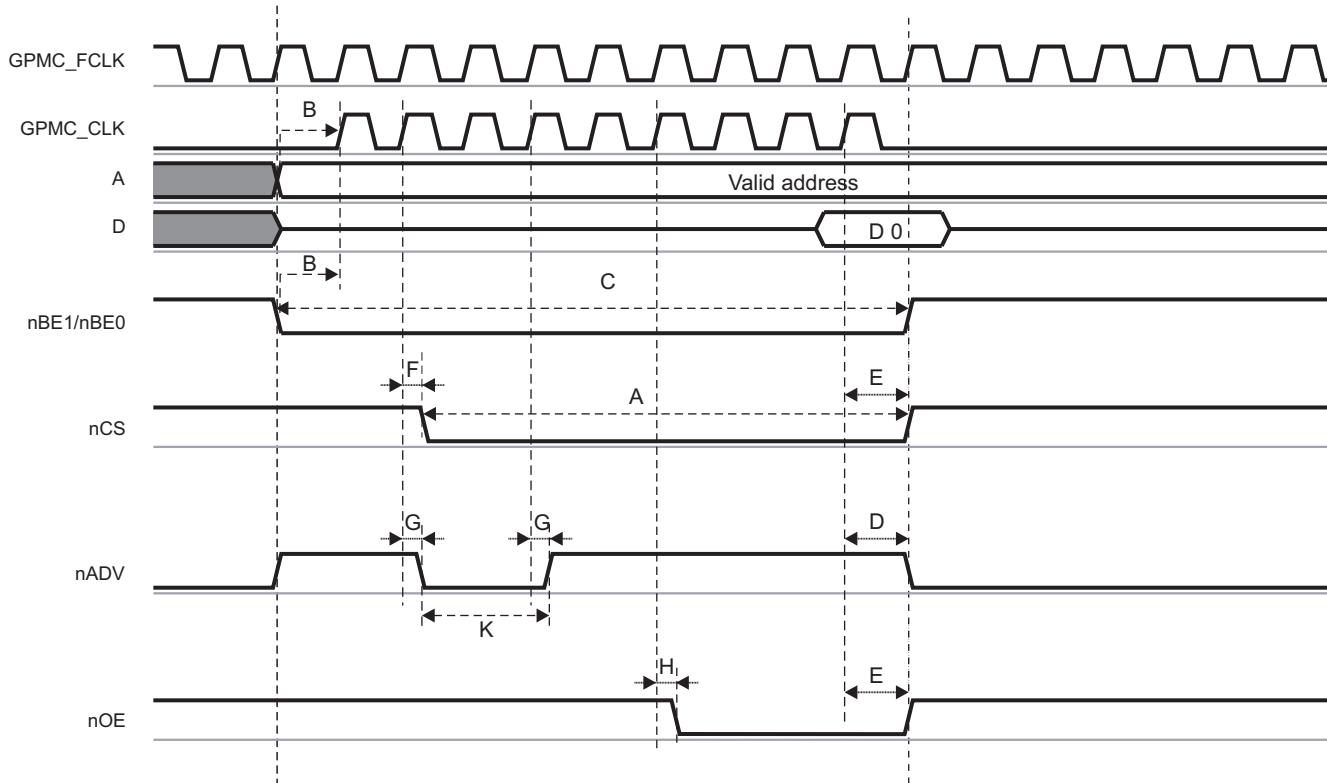
- Read operation  
 $K = (\text{ADVRDOFFTIME} - \text{ADVONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$
- Write operation  
 $K = (\text{ADVWRDOFFTIME} - \text{ADVONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC\_FCLK period}$

For GPMC\_WAIT invalid to first data latching GPMC\_CLK edge:

$$L = \text{WAITMONITORINGTIME} * (\text{GPMCFCLKDIVIDER} + 1) * \text{GPMC\_FCLK period} + \text{GPMC\_CLK period}$$

[Figure 7-45](#) shows a synchronous NOR single read simplified example where formulas are associated with signal waves.

**Figure 7-45. Synchronous NOR Single Read Simplified Example**



### 7.1.3.9.1.3 Asynchronous NOR Flash Timing Parameters Formulas

This section lists all formulas to use in order to calculate asynchronous NOR timing parameters. This is the case when [11-10] DEVICETYPE = 0x0 and when READTYPE or WRITETYPE are set to asynchronous mode.

**Table 7-44. Asynchronous NOR Formulas Description Table**

| Configuration Parameter | Unit | Description   |
|-------------------------|------|---|
| A                       | ns   | Pulse duration - GPMC_CS <sub>n</sub> low                                     |
| B                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_ADV <sub>n</sub> _ALE invalid |
| C                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_OEn invalid (single read)     |
| D                       | ns   | Pulse duration - address bus valid - 2nd, 3rd and 4th accesses                |
| E                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_WEn valid                     |
| F                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_WEn invalid                   |
| G                       | ns   | Address invalid duration between 2 successive R/W accesses                    |
| H                       | ns   | Setup time - read data valid before GPMC_OEn high                             |
| I                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_OEn invalid (burst read)      |
| J                       | ns   | Delay time - address bus valid to GPMC_CS <sub>n</sub> valid                  |
|                         |      | Delay time - data bus valid to GPMC_CS <sub>n</sub> valid                     |
|                         |      | Delay time - GPMC_BE0n_CLE/GPMC_BE1n valid to GPMC_CS <sub>n</sub> valid      |
| K                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_ADV <sub>n</sub> _ALE valid   |
| L                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_OEn valid                     |
| M                       | ns   | Delay time - GPMC_CS valid to first data latching edge                        |
| N                       | ns   | Pulse duration - GPMC_BE0n_CLE/GPMC_BE1n valid time                           |
| O                       | ns   | Delay time - GPMC_CS <sub>n</sub> valid to GPMC_ADV <sub>n</sub> _ALE valid   |

The configuration parameters are calculated through the following formulas. Note that these formulas are not exhaustive.

GPMC\_CS<sub>n</sub> low pulse:

For single read:  $A = (\text{CSRDOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC_FCLK period}$

For burst read:  $A = (\text{CSRDOFFTIME} - \text{CSONTIME} + (N - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC_FCLK period}$ , where N = page burst access number

For single write:  $A = (\text{CSWROFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC_FCLK period}$

For burst write:  $A = (\text{CSWROFFTIME} - \text{CSONTIME} + (N - 1) * \text{PAGEBURSTACCESSTIME}) * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC_FCLK period}$ , where N = page burst access number

GPMC\_CS<sub>n</sub> valid to GPMC\_ADV<sub>n</sub>\_ALE invalid delay:

For reading:  $B = ((\text{ADVRDOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{ADVEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC_FCLK period}$

For writing:  $B = ((\text{ADVVROFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{ADVEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC_FCLK period}$

$C = ((\text{OEOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{OEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC_FCLK period}$

$D = \text{PAGEBURSTACCESSTIME} * (\text{TIMEPARAGRANULARITY} + 1) * \text{GPMC_FCLK period}$

$E = ((\text{WEONTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{WEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC_FCLK period}$

$F = ((\text{WEOFFTIME} - \text{CSONTIME}) * (\text{TIMEPARAGRANULARITY} + 1) + 0.5 * (\text{WEEXTRADELAY} - \text{CSEXTRADELAY})) * \text{GPMC_FCLK period}$

$G = \text{CYCLE2CYCLEDELAY} * \text{GPMC_FCLK period}$

$H = ((OEOFFSETIME - ACCESSTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * OEEXTRADELAY) * GPMC_FCLK$  period

$I = ((OEOFFSETIME + (N - 1) * PAGEBURSTACCESSTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (OEEXTRADELAY - CSEXTRADELAY)) * GPMC_FCLK$  period, where  $N$  = page burst access number

$J = (CSONTIME * (TIMEPARAGRANULARITY + 1) + 0.5 * CSEXTRADELAY) * GPMC_FCLK$  period

$K = ((ADVONTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY - CSEXTRADELAY)) * GPMC_FCLK$  period

$L = ((OEONTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (OEEXTRADELAY - CSEXTRADELAY)) * GPMC_FCLK$  period  $M = ((ACCESSTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) - 0.5 * CSEXTRADELAY) * GPMC_FCLK$  period

GPMC\_BE0n\_CLE/GPMC\_BE1n pulse:

For single read:  $N = RDCYCLETIME * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK$  period

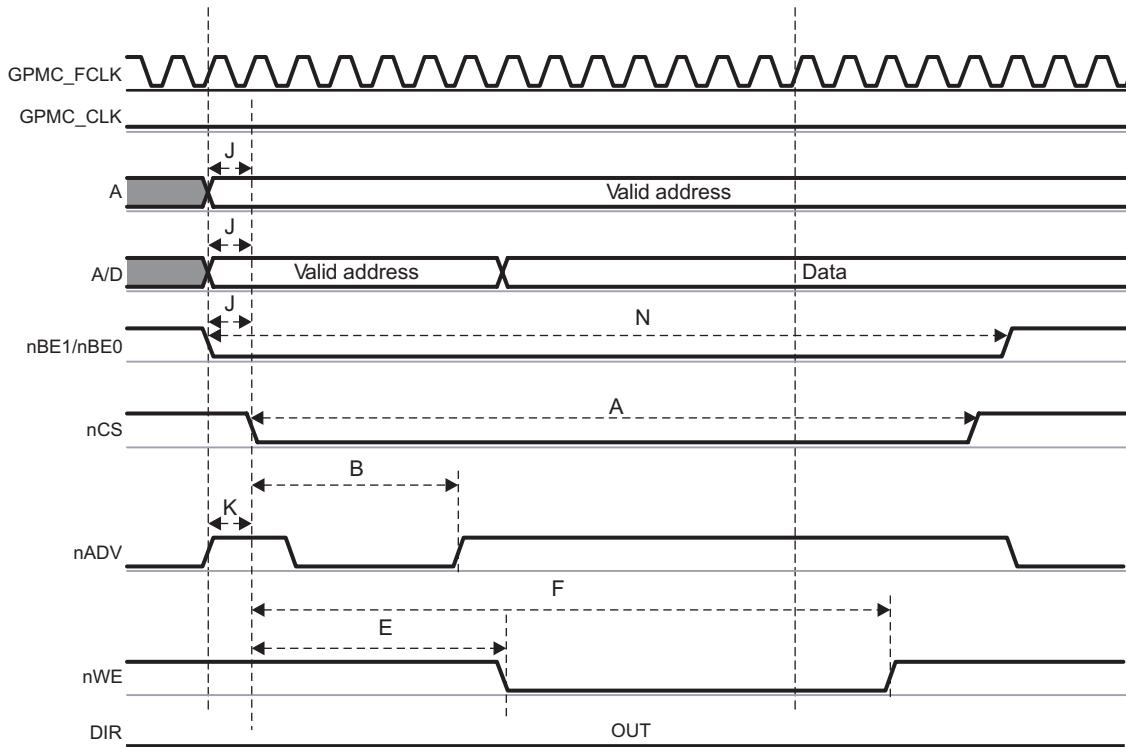
For burst read:  $N = (RDCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK$  period, where  $N$  = page burst access number

For burst write:  $N = (WRCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME) * (TIMEPARAGRANULARITY + 1) * GPMC_FCLK$  period, where  $N$  = page burst access number

$O = ((WRCYCLETIME + (N - 1) * PAGEBURSTACCESSTIME - CSONTIME) * (TIMEPARAGRANULARITY + 1) + 0.5 * (ADVEXTRADELAY - CSEXTRADELAY)) * GPMC_FCLK$  period

Figure 7-46 shows an asynchronous NOR single write simplified example where formulas are associated with signal waves.

**Figure 7-46. Asynchronous NOR Single Write Simplified Example**



Write multiple access is not supported in asynchronous mode. If WRITEMULTIPLE is enabled with WRITETYPE as asynchronous, the GPMC processes single asynchronous accesses.

## 7.1.4 Use Cases

### 7.1.4.1 How to Set GPMC Timing Parameters for Typical Accesses

#### 7.1.4.1.1 External Memory Attached to the GPMC Module

As discussed in the introduction to this chapter, the GPMC module supports the following external memory types:

- Asynchronous or synchronous, 8-bit or 16-bit-width memory or device
- 16-bit address/data-multiplexed or not multiplexed NOR flash device
- 8- or 16-bit NAND flash device

The following examples show how to calculate GPMC timing parameters by showing a typical parameter setup for the access to be performed.

The example is based on a 512-Mb multiplexed NOR flash memory with the following characteristics:

- Type: NOR flash (address/data-multiplexed mode)
- Size: 512M bits
- Data Bus: 16 bits wide
- Speed: 104 MHz clock frequency
- Read access time: 80 ns

#### 7.1.4.1.2 Typical GPMC Setup

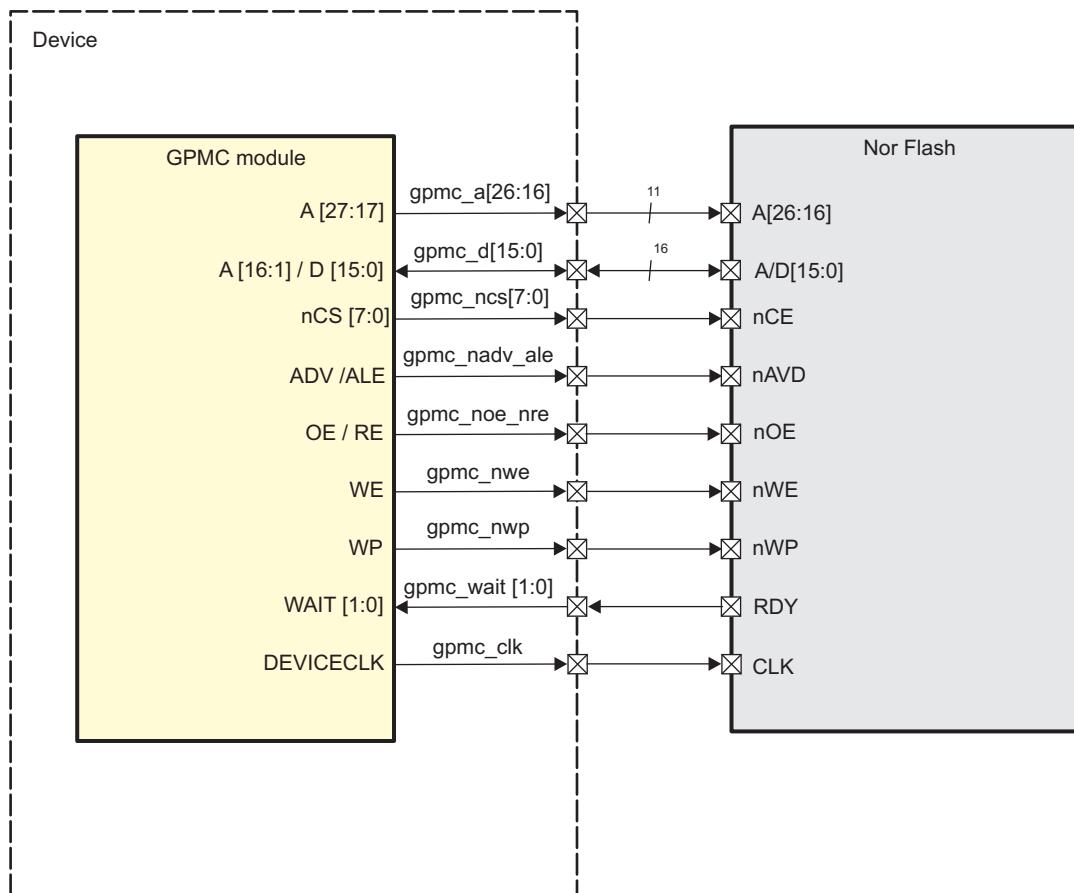
[Table 7-45](#) lists some of the I/Os of the GPMC module.

**Table 7-45. GPMC Signals**

| Signal Name    | I/O      | Description  |
|----------------|----------|--|
| GPMC_FCLK      | Internal | Functional and interface clock. Acts as the time reference.                          |
| GPMC_CLK       | O        | External clock provided to the external device for synchronous operations            |
| GPMC_A[27:17]  | O        | Address  |
| GPMC_AD[15: 0] | I/O      | Data-multiplexed with addresses A[16:1] on memory side                               |
| GPMC_CSxn      | O        | Chip-select (where x = 0, or 1)  |
| GPMC_ADVn_ALE  | O        | Address valid enable   |
| GPMC_OE_REn    | O        | Output enable (read access only)   |
| GPMC_WEn       | O        | Write enable (write access only)   |
| GPMC_WAIT[1:0] | I        | Ready signal from memory device. Indicates when valid burst data is ready to be read |

Figure 7-47 shows the typical connection between the GPMC module and an attached NOR Flash memory.

**Figure 7-47. GPMC Connection to an External NOR Flash Memory**



The following sections demonstrate how to calculate GPMC parameters for three access types:

- Synchronous burst read
- Asynchronous read
- Asynchronous single write

#### 7.1.4.1.3 GPMC Configuration for Synchronous Burst Read Access

The clock runs at 104 MHz ( $f = 104$  MHz;  $T = 9,615$  ns).

[Table 7-46](#) shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

[Table 7-47](#) shows how to calculate timings for the GPMC using the memory parameters.

[Figure 7-48](#) shows the synchronous burst read access.

**Table 7-46. Useful Timing Parameters on the Memory Side**

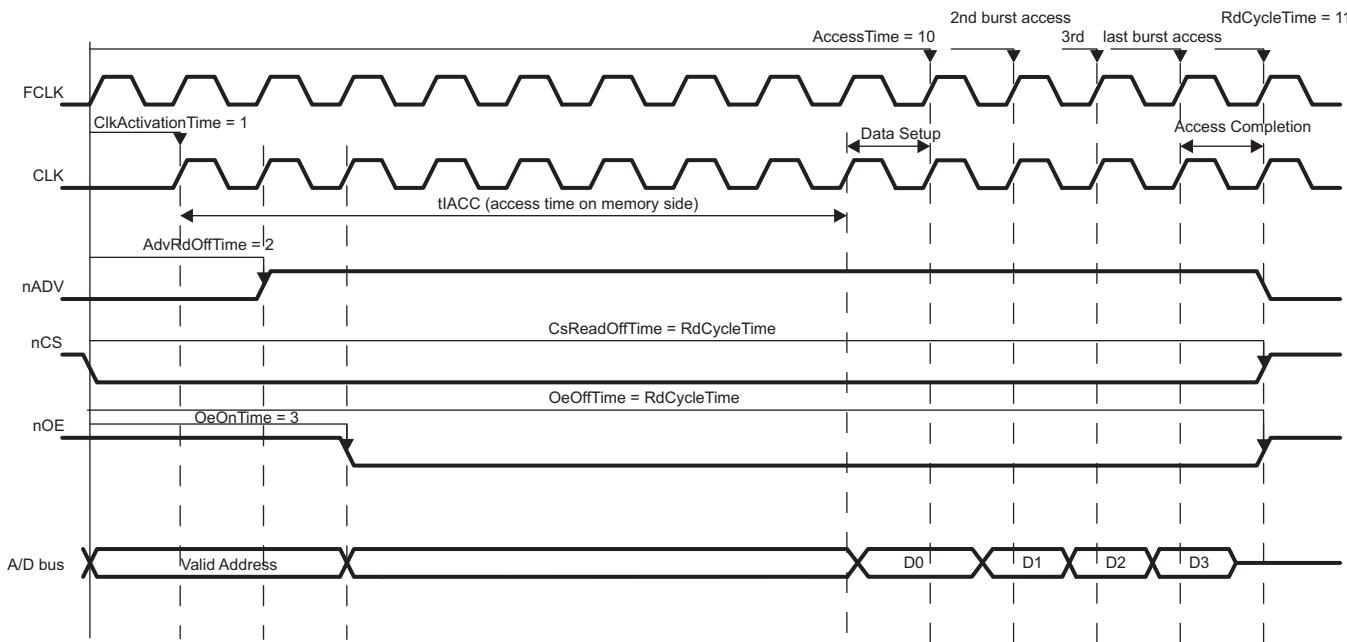
| AC Read Characteristics on the Memory Side | Description                                   | Duration (ns) |
|--|---|---------------|
| tCES                                       | CSn setup time to clock                       | 0             |
| tACS                                       | Address setup time to clock                   | 3             |
| tIACC                                      | Synchronous access time                       | 80            |
| tBACC                                      | Burst access time valid clock to output delay | 5,2           |
| tCEZ                                       | Chip-select to High-Impedance                 | 7             |
| tOEZ                                       | Output enable to High-Impedance               | 7             |
| tAVC                                       | ADVn setup time                               | 6             |
| tAVD                                       | AVDn pulse                                    | 6             |
| tACH                                       | Address hold time from clock                  | 3             |

The following terms, which describe the timing interface between the controller and its attached device, are used to calculate the timing parameters on the GPMC side:

- Read Access time (GPMC side): Time required to activate the clock + read access time requested on the memory side + data setup time required for optimal capture of a burst of data
- Data setup time (GPMC side): Ensures a good capture of a burst of data (as opposed to taking a burst of data out). One word of data is processed in one clock cycle ( $T = 9,615$  ns). The read access time between 2 bursts of data is tBACC = 5,2 ns. Therefore, data setup time is a clock period - tBACC = 4,415 ns of data setup.
- Access completion (GPMC side): (Different from page burst access time) Time required between the last burst access and access completion: CSn/OEn hold time (CSn and OEn must be released at the end of an access. These signals are held to allow the access to complete).
- Read cycle time (GPMC side): Read Access time + access completion
- Write cycle time for burst access: Not supported for NOR flash memory

**Table 7-47. Calculating GPMC Timing Parameters**

| Parameter Name on GPMC Side | Formula   | Duration (ns)               | Number of Clock Cycles (F = 104 MHz) | GPMC Register Configurations |
|-----------------------------|---|-----------------------------|--------------------------------------|------------------------------|
| GPMC FCLK Divider           | -   | -                           | -                                    | GPMCFCLKDIVIDER = 0          |
| ClkActivationTime           | min ( tCES, tACS )  | 3                           | 1                                    | CLKACTIVATIONTIME = 1        |
| RdAccessTime                | roundmax ( ClkActivationTime + tIACC + DataSetupTime )              | 94,03: (9,615 + 80 + 4,415) | 10 : roundmax (94,03 / 9,615)        | ACCESTIME = Ah               |
| PageBurstAccessTime         | roundmax (tBACC)  | roundmax (5,2)              | 1                                    | PAGEBURSTACCESTIME = 1       |
| RdCycleTime                 | AccessTime + max ( tCEZ, tOEZ )                                     | 101, 03: (94, 03 + 7)       | 11                                   | RDCYCLETIME = Bh             |
| CsOnTime                    | tCES  | 0                           | 0                                    | CSONTIME = 0                 |
| CsReadOffTime               | RdCycleTime   | -                           | 11                                   | CSRDOFFTIME = Bh             |
| AdvOnTime                   | tAVC  | 0                           | 0                                    | ADVONTIME = 0                |
| AdvRdOffTime                | tAVD + tAVC   | 12                          | 2                                    | ADVRDOFFTIME = 2h            |
| OeOnTime                    | (ClkActivationTime + tACH) < OeOnTime < (ClkActivationTime + tIACC) | -                           | 3, for instance                      | OEONTIME = 3h                |
| OeOffTime                   | RdCycleTime   | -                           | 11                                   | OEOFETIME = Bh               |

**Figure 7-48. Synchronous Burst Read Access (Timing Parameters in Clock Cycles)**

#### 7.1.4.1.4 GPMC Configuration for Asynchronous Read Access

The clock runs at 104 MHz ( $f = 104$  MHz;  $T = 9,615$  ns).

[Table 7-48](#) shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

[Table 7-49](#) shows how to calculate timings for the GPMC using the memory parameters.

[Figure 7-49](#) shows the asynchronous read access.

**Table 7-48. AC Characteristics for Asynchronous Read Access**

| AC Read Characteristics on the Memory Side | Description                               | Duration (ns) |
|--|---|---------------|
| tCE  | Read Access time from CSn low             | 80            |
| tAAVDS                                     | Address setup time to rising edge of ADVn | 3             |
| tAVDP                                      | ADVn low time                             | 6             |
| tCAS                                       | CSn setup time to ADVn                    | 0             |
| tOE  | Output enable to output valid             | 6             |
| tOEZ                                       | Output enable to High-Impedance           | 7             |

Use the following formula to calculate the RdCycleTime parameter for this typical access:

$$\text{RdCycleTime} = \text{RdAccessTime} + \text{AccessCompletion} = \text{RdAccessTime} + 1 \text{ clock cycle} + \text{tOEZ}$$

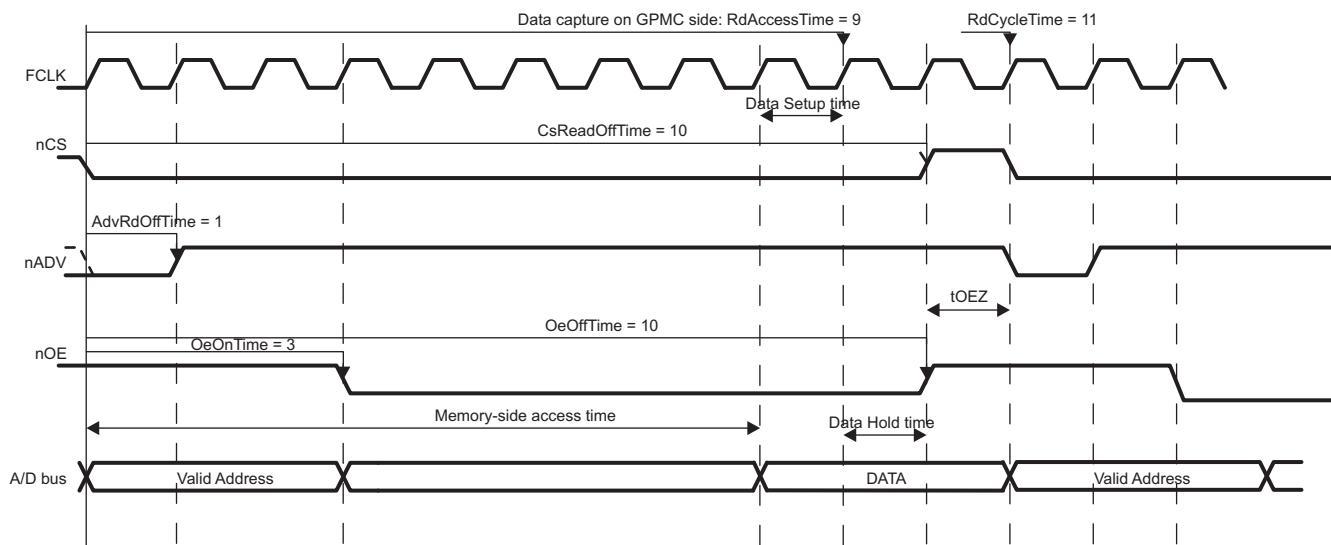
- First, on the memory side, the external memory makes the data available to the output bus. This is the memory-side read access time defined in [Table 7-49](#): the number of clock cycles between the address capture (ADVn rising edge) and the data valid on the output bus.

The GPMC requires some hold time to allow the data to be captured correctly and the access to be finished.

- To read the data correctly, the GPMC must be configured to meet the data setup time requirement of the memory; the GPMC module captures the data on the next rising edge. This is access time on the GPMC side.
- There must also be a data hold time for correctly reading the data (checking that there is no OEn/CSn deassertion while reading the data). This data hold time is 1 clock cycle (that is, AccessTime + 1).
- To complete the access, OEn/CSn signals are driven to high-impedance. AccessTime + 1 + tOEZ is the read cycle time.
- Addresses can now be relatched and a new read cycle begun.

**Table 7-49. GPMC Timing Parameters for Asynchronous Read Access**

| Parameter Name on GPMC side | Formula   | Duration (ns) | Number of Clock Cycles ( $F = 104$ MHz) | GPMC Register Configurations |
|-----------------------------|---|---------------|---|------------------------------|
| ClkActivationTime           | n/a (asynchronous mode)                         |               |   |                              |
| AccessTime                  | round max (tCE)                                 | 80            | 9                                       | ACCESSTIME = 9h              |
| PageBurstAccessTime         | n/a (single access)                             |               |   |                              |
| RdCycleTime                 | AccessTime + 1cycle + tOEZ                      | 96, 615       | 11                                      | RDCYCLETIME = Bh             |
| CsOnTime                    | tCAS  | 0             | 0                                       | CSONTIME = 0                 |
| CsReadOffTime               | AccessTime + 1 cycle                            | 89, 615       | 10                                      | CSRDOFFTIME = Ah             |
| AdvOnTime                   | tAAVDS  | 3             | 1                                       | ADVONTIME = 1                |
| AdvRdOffTime                | tAAVDS + tAVDP                                  | 9             | 1                                       | ADVRDOFFTIME = 1             |
| OeOnTime                    | OeOnTime $\geq$ AdvRdOffTime (multiplexed mode) | -             | 3, for instance                         | OEONTIME = 3h                |
| OeOffTime                   | AccessTime + 1cycle                             | 89, 615       | 10                                      | OEOFETIME = Ah               |

**Figure 7-49. Asynchronous Single Read Access (Timing Parameters in Clock Cycles)**


#### 7.1.4.1.5 GPMC Configuration for Asynchronous Single Write Access

The clock runs at 104 MHz: ( $f = 104$  MHz;  $T = 9,615$  ns).

[Table 7-50](#) shows how to calculate timings for the GPMC using the memory parameters.

[Table 7-51](#) shows the timing parameters (on the memory side) that determine the parameters on the GPMC side.

[Figure 7-50](#) shows the synchronous burst write access.

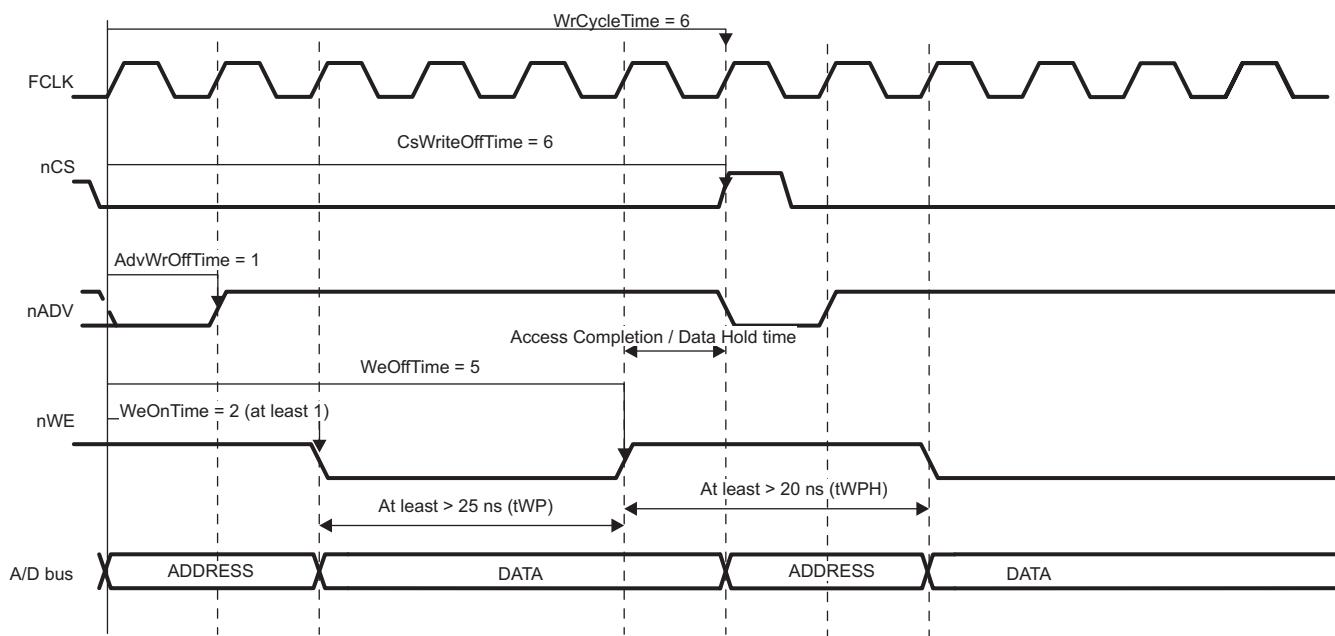
**Table 7-50. AC Characteristics for Asynchronous Single Write (Memory Side)**

| AC Characteristics on the Memory Side | Description            | Duration (ns) |
|---------------------------------------|------------------------|---------------|
| tWC                                   | Write cycle time       | 60            |
| tAVDP                                 | ADVn low time          | 6             |
| tWP                                   | Write pulse width      | 25            |
| tWPH                                  | Write pulse width high | 20            |
| tCS                                   | CSn setup time to WEn  | 3             |
| tCAS                                  | CSn setup time to ADVn | 0             |
| tAVSC                                 | ADVn setup time        | 3             |

For asynchronous single write access, write cycle time is WrCycleTime = WeOffTime + AccessCompletion = WeOffTime + 1. For the AccessCompletion, the GPMC requires 1 cycle of data hold time (CSn de-assertion).

**Table 7-51. GPMC Timing Parameters for Asynchronous Single Write**

| Parameter Name on GPMC side | Formula  | Duration (ns) | Number of Clock Cycles (F = 104 MHz) | GPMC Register Configurations |
|-----------------------------|--|---------------|--------------------------------------|------------------------------|
| ClkActivationTime           | n/a (asynchronous mode)  |               |                                      |                              |
| AccessTime                  | Applicable only to WAITMONITORING (the value is the same as for read access) |               |                                      |                              |
| PageBurstAccessTime         | n/a (single access)  |               |                                      |                              |
| WrCycleTime                 | WeOffTime + AccessCompletion   | 57, 615       | 6                                    | WRCYCLETIME = 6h             |
| CsOnTime                    | tCAS   | 0             | 0                                    | CSONTIME = 0                 |
| CsWrOffTime                 | WeOffTime + 1  | 57, 615       | 6                                    | CSWROFFTIME = 6h             |
| AdvOnTime                   | tAVSC  | 3             | 1                                    | ADVONTIME = 1                |
| AdvWrOffTime                | tAVSC + tAVDP  | 9             | 1                                    | ADVWRFFTIME = 1              |
| WeOnTime                    | tCS  | 3             | 1                                    | WEONTIME = 1                 |
| WeOffTime                   | tCS + tWP + tWPH   | 48            | 5                                    | WEOFFTIME = 5h               |

**Figure 7-50. Asynchronous Single Write Access (Timing Parameters in Clock Cycles)**


### 7.1.4.2 How to Choose a Suitable Memory to Use With the GPMC

This section is intended to help the user select a suitable memory device to interface with the GPMC controller.

#### 7.1.4.2.1 Supported Memories or Devices

NAND flash and NOR flash architectures are the two flash technologies. The GPMC supports various types of external memory or device, basically any one that supports NAND or NOR protocols:

- 8- and 16-bit width asynchronous or synchronous memory or device (8-bit: non burst device only)
- 16-bit address and data multiplexed NOR flash devices (pSRAM)
- 8- and 16-bit NAND flash device

#### 7.1.4.2.2 NAND Interface Protocol

NAND flash architecture, introduced in 1989, is a flash technology. NAND is a page-oriented memory device, that is, read and write accesses are done by pages. NAND achieves great density by sharing common areas of the storage transistor, which creates strings of serially connected transistors (in NOR devices, each transistor stands alone). Thanks to its high density NAND is best suited to devices requiring high capacity data storage, such as pictures, music, or data files. NAND non-volatility, makes of it a good storage solution for many applications where mobility, low power, and speed are key factors. Low pin count and simple interface are other advantages of NAND.

Table 7-52 summarizes the NAND interface signals level applied to external device or memories.

**Table 7-52. NAND Interface Bus Operations Summary**

| Bus Operation         | CLE | ALE | CEn | WE <sub>n</sub> | RE <sub>n</sub> | WP <sub>n</sub> |
|-----------------------|-----|-----|-----|-----------------|-----------------|-----------------|
| Read (cmd input)      | H   | L   | L   | RE              | H               | x               |
| Read (add input)      | L   | H   | L   | RE              | H               | x               |
| Write (cmd input)     | H   | L   | L   | RE              | H               | H               |
| Write (add input)     | L   | H   | L   | RE              | H               | H               |
| Data input            | L   | L   | L   | RE              | H               | H               |
| Data output           | L   | L   | L   | H               | FE              | x               |
| Busy (during read)    | x   | x   | H   | H               | H               | x               |
| Busy (during program) | x   | x   | x   | x               | x               | H               |
| Busy (during erase)   | x   | x   | x   | x               | x               | H               |
| Write protect         | x   | x   | x   | x               | x               | L               |
| Stand-by              | x   | x   | H   | x               | x               | H/L             |

#### 7.1.4.2.3 NOR Interface Protocol

NOR flash architecture, introduced in 1988, is a flash technology. Unlike NAND, which is a sequential access device, NOR is directly addressable; i.e., it is designed to be a random access device. NOR is best suited to devices used to store and run code or firmware, usually in small capacities. While NOR has fast read capabilities it has slow write and erase functions compared to NAND architecture.

Table 7-53 summarizes the NOR interface signals level applied to external device or memories.

**Table 7-53. NOR Interface Bus Operations Summary**

| Bus Operation        | CLK     | ADV <sub>n</sub> | CS <sub>n</sub> | OEn | WE <sub>n</sub> | WAIT     | DQ[15:0] |
|----------------------|---------|------------------|-----------------|-----|-----------------|----------|----------|
| Read (asynchronous)  | x       | L                | L               | L   | H               | Asserted | Output   |
| Read (synchronous)   | Running | L                | L               | L   | H               | Driven   | Output   |
| Read (burst suspend) | Halted  | x                | L               | H   | H               | Active   | Output   |
| Write                | x       | L                | L               | H   | L               | Asserted | Input    |

**Table 7-53. NOR Interface Bus Operations Summary (continued)**

| Bus Operation  | CLK | ADVn | CSn | OEn | WEn | WAIT     | DQ[15:0] |
|----------------|-----|------|-----|-----|-----|----------|----------|
| Output disable | x   | x    | L   | H   | H   | Asserted | High-Z   |
| Standby        | x   | x    | H   | x   | x   | High-Z   | High-Z   |

#### 7.1.4.2.4 Other Technologies

Other supported device type interact with the GPMC through the NOR interface protocol.

OneNAND Flash is a high-density and low-power memory device. It is based on single- or multi-level-cell NAND core with SRAM and logic, and interfaces as a synchronous NOR Flash, plus has synchronous write capability. It reads faster than conventional NAND and writes faster than conventional NOR flash. Hence, it is appropriate for both mass storage and code storage.

pSRAM stands for pseudo-static random access memory. pSRAM is a low-power memory device for mobile applications. pSRAM is based on the DRAM cell with internal refresh and address control features, and interfaces as a synchronous NOR Flash, plus has synchronous write capability.

#### 7.1.4.2.5 Supported Protocols

The GPMC supports the following interface protocols when communicating with external memory or external devices:

- Asynchronous read/write access
- Asynchronous read page access (4-8-16 Word16)
- Synchronous read/write access
- Synchronous read burst access without wrap capability (4-8-16 Word16)
- Synchronous read burst access with wrap capability (4-8-16 Word16)

#### 7.1.4.2.6 GPMC Features and Settings

This section lists GPMC features and settings:

- Supported device type: up to four NAND or NOR protocol external memories or devices
- Operating Voltage: 3.3V
- Maximum GPMC addressing capability: 512 MBytes divided into eight chip-selects
- Maximum supported memory size: 256 MBytes (must be a power-of-2)
- Minimum supported memory size: 16 MBytes (must be a power-of-2). Aliasing occurs when addressing smaller memories.
- Data path to external memory or device: 8- and 16-bit wide
- Burst and page access: burst of 4-8-16 Word16
- Supports bus keeping
- Supports bus turn around

## 7.1.5 Registers

**Table 7-54** provides a summary of the GPMC registers. All GPMC registers are aligned to 32-bit address boundaries. All register file accesses, except the GPMC\_NAND\_DATA\_i register, are little-endian. If the GPMC\_NAND\_DATA\_i register is accessed, the endianness is access-dependent.

In this section, i corresponds to the chip-select number, i = 0 to 6.

**Table 7-54. GPMC Registers**

| Address Offset   | Register Name                      | Section                          |
|------------------|------------------------------------|----------------------------------|
| 0h               | GPMC_REVISION                      | <a href="#">Section 7.1.5.1</a>  |
| 10h              | GPMC_SYSCONFIG                     | <a href="#">Section 7.1.5.2</a>  |
| 14h              | GPMC_SYSSTATUS                     | <a href="#">Section 7.1.5.3</a>  |
| 18h              | GPMC_IRQSTATUS                     | <a href="#">Section 7.1.5.4</a>  |
| 1Ch              | GPMC_IRQENABLE                     | <a href="#">Section 7.1.5.5</a>  |
| 40h              | GPMC_TIMEOUT_CONTROL               | <a href="#">Section 7.1.5.6</a>  |
| 44h              | GPMC_ERR_ADDRESS                   | <a href="#">Section 7.1.5.7</a>  |
| 48h              | GPMC_ERR_TYPE                      | <a href="#">Section 7.1.5.8</a>  |
| 50h              | GPMC_CONFIG                        | <a href="#">Section 7.1.5.9</a>  |
| 54h              | GPMC_STATUS                        | <a href="#">Section 7.1.5.10</a> |
| 60h + (30h × i)  | GPMC_CONFIG1_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.11</a> |
| 64h + (30h × i)  | GPMC_CONFIG2_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.12</a> |
| 68h + (30h × i)  | GPMC_CONFIG3_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.13</a> |
| 6Ch + (30h × i)  | GPMC_CONFIG4_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.14</a> |
| 70h + (30h × i)  | GPMC_CONFIG5_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.15</a> |
| 74h + (30h × i)  | GPMC_CONFIG6_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.16</a> |
| 78h + (30h × i)  | GPMC_CONFIG7_i <sup>(1)</sup>      | <a href="#">Section 7.1.5.17</a> |
| 7Ch + (30h × i)  | GPMC_NAND_COMMAND_i <sup>(1)</sup> | <a href="#">Section 7.1.5.18</a> |
| 80h + (30h × i)  | GPMC_NAND_ADDRESS_i <sup>(1)</sup> | <a href="#">Section 7.1.5.19</a> |
| 84h + (30h × i)  | GPMC_NAND_DATA_i <sup>(1)</sup>    | <a href="#">Section 7.1.5.20</a> |
| 1E0h             | GPMC_PREFETCH_CONFIG1              | <a href="#">Section 7.1.5.21</a> |
| 1E4h             | GPMC_PREFETCH_CONFIG2              | <a href="#">Section 7.1.5.22</a> |
| 1ECH             | GPMC_PREFETCH_CONTROL              | <a href="#">Section 7.1.5.23</a> |
| 1F0h             | GPMC_PREFETCH_STATUS               | <a href="#">Section 7.1.5.24</a> |
| 1F4h             | GPMC_ECC_CONFIG                    | <a href="#">Section 7.1.5.25</a> |
| 1F8h             | GPMC_ECC_CONTROL                   | <a href="#">Section 7.1.5.26</a> |
| 1FCh             | GPMC_ECC_SIZE_CONFIG               | <a href="#">Section 7.1.5.27</a> |
| 200h + (4h × k)  | GPMC_ECCj_RESULT <sup>(2)(3)</sup> | <a href="#">Section 7.1.5.28</a> |
| 240h + (10h × i) | GPMC_BCH_RESULT0_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.29</a> |
| 244h + (10h × i) | GPMC_BCH_RESULT1_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.30</a> |
| 248h + (10h × i) | GPMC_BCH_RESULT2_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.31</a> |
| 24Ch + (10h × i) | GPMC_BCH_RESULT3_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.32</a> |
| 300h + (10h × i) | GPMC_BCH_RESULT4_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.34</a> |
| 304h + (10h × i) | GPMC_BCH_RESULT5_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.35</a> |
| 308h + (10h × i) | GPMC_BCH_RESULT6_i <sup>(1)</sup>  | <a href="#">Section 7.1.5.36</a> |
| 2D0h             | GPMC_BCH_SWDATA                    | <a href="#">Section 7.1.5.33</a> |

<sup>(1)</sup> i = 0 to 6 for GPMC

<sup>(2)</sup> j = 1 to 9 for GPMC

<sup>(3)</sup> k = j - 1

### 7.1.5.1 GPMC\_REVISION

This register contains the IP revision code.

**Figure 7-51. GPMC\_REVISION**

|    |          |   |     |
|----|----------|---|-----|
| 31 | Reserved |   | 16  |
|    | R-0      |   |     |
| 15 | 8        | 7 | 0   |
|    | Reserved |   | REV |
|    | R-0      |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-55. GPMC\_REVISION Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-8 | Reserved | 0     | Reserved   |
| 7-0  | REV      | 0-FFh | IP revision. Major revision is [7-4]. Minor revision is [3-0]. Examples: 10h for revision 1.0, 21h for revision 2.1. |

### 7.1.5.2 GPMC\_SYSCONFIG

This register controls the various parameters of the OCP interface.

**Figure 7-52. GPMC\_SYSCONFIG**

|    |          |           |                         |
|----|----------|-----------|-------------------------|
| 31 | Reserved |           | 16                      |
|    | R-0      |           |                         |
| 15 | 5        | 4         | 3 2 1 0                 |
|    | Reserved | SIDLEMODE | Rsvd SOFTRESET AUTOIDLE |
|    | R-0      | R/W       | R-0 R/W R/W             |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

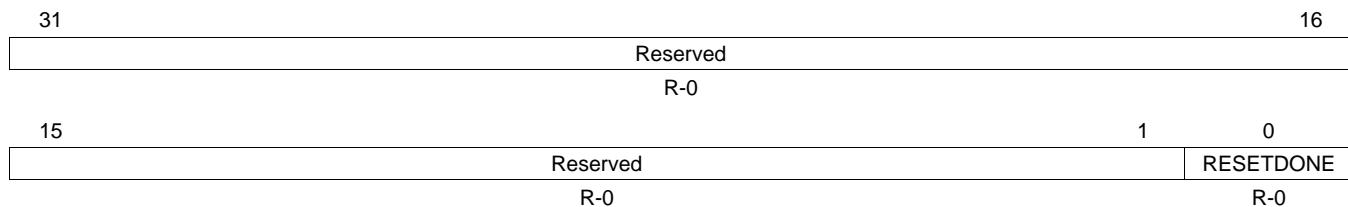
**Table 7-56. GPMC\_SYSCONFIG Field Descriptions**

| Bit  | Field     | Value               | Description  |
|------|-----------|---------------------|--|
| 31-5 | Reserved  | 0                   | Reserved   |
| 4-3  | SIDLEMODE | 0<br>1h<br>2h<br>3h | Idle mode<br>Force-idle. An idle request is acknowledged unconditionally<br>No-idle. An idle request is never acknowledged<br>Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module<br>Reserved |
| 2    | Reserved  | 0                   | Reserved   |
| 1    | SOFTRESET | 0<br>1              | Software reset (Set 1 to this bit triggers a module reset. This bit is automatically reset by hardware. During reads, it always returns 0)<br>Normal mode<br>The module is reset   |
| 0    | AUTOIDLE  | 0<br>1              | Internal OCP clock gating strategy<br>Interface clock is free-running<br>Reserved  |

### **7.1.5.3 GPMC\_SYSSTATUS**

This register provides status information about the module, excluding the interrupt status information.

**Figure 7-53. GPMC\_SYSSTATUS**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-57. GPMC\_SYSSTATUS Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>  |
|------------|--------------|--------------|---|
| 31-1       | Reserved     | 0            | Reserved  |
| 0          | RESETDONE    | R0<br>R1     | Internal reset monitoring<br>Internal module reset in on-going<br>Reset completed |

### 7.1.5.4 GPMC\_IRQSTATUS

This interrupt status register regroups all the status of the module internal events that can generate an interrupt.

**Figure 7-54. GPMC\_IRQSTATUS**

|          |          |                              |                              |
|----------|----------|------------------------------|------------------------------|
|          |          |                              |                              |
| Reserved |          |                              |                              |
| 31       | 15       | 10                           | 9                            |
|          |          |                              |                              |
| R-0      | Reserved | WAIT1EDGE<br>DETECTIONSTATUS | WAIT0EDGE<br>DETECTIONSTATUS |
| 16       | 8        | R/W-0                        | R/W-0                        |
| 7        | 2        | 1                            | 0                            |
|          |          |                              |                              |
| R-0      | Reserved | TERMINAL<br>COUNTSTATUS      | FIFOEVENT<br>STATUS          |
| R/W-0    | R/W-0    | R/W-0                        | R/W-0                        |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-58. GPMC\_IRQSTATUS Field Descriptions**

| Bit   | Field                    | Value                | Description  |
|-------|--------------------------|----------------------|--|
| 31-10 | Reserved                 | 0                    | Reserved   |
| 9     | WAIT1EDGEDETECTIONSTATUS | R0<br>W0<br>R1<br>W1 | Status of the Wait1 Edge Detection interrupt<br>A transition on WAIT1 input pin has not been detected<br>WAIT1EDGEDETECTIONSTATUS bit unchanged<br>A transition on WAIT1 input pin has been detected<br>WAIT1EDGEDETECTIONSTATUS bit is reset  |
| 8     | WAIT0EDGEDETECTIONSTATUS | R0<br>W0<br>R1<br>W1 | Status of the Wait0 Edge Detection interrupt<br>A transition on WAIT0 input pin has not been detected<br>WAIT0EDGEDETECTIONSTATUS bit unchanged<br>A transition on WAIT0 input pin has been detected<br>WAIT0EDGEDETECTIONSTATUS bit is reset  |
| 7-2   | Reserved                 | 0                    | Reserved   |
| 1     | TERMINALCOUNTSTATUS      | R0<br>W0<br>R1<br>W1 | Status of the TerminalCountEvent interrupt<br>Indicates that CountValue is greater than 0<br>TERMINALCOUNTSTATUS bit unchanged<br>Indicates that CountValue is equal to 0<br>TERMINALCOUNTSTATUS bit is reset  |
| 0     | FIFOEVENTSTATUS          | R0<br>W0<br>R1<br>W1 | Status of the FIFOEvent interrupt<br>Indicates than less than GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bytes are available in prefetch mode and less than FIFOTHRESHOLD bytes free places are available in write-posting mode.<br>FIFOEVENTSTATUS bit unchanged<br>Indicates than at least GPMC_PREFETCH_STATUS[16] FIFOTHRESHOLDSTATUS bytes are available in prefetch mode and at least FIFOTHRESHOLD bytes free places are available in write-posting mode.<br>FIFOEVENTSTATUS bit is reset |

### 7.1.5.5 GPMC\_IRQENABLE

The interrupt enable register allows to mask/unmask the module internal sources of interrupt, on a event-by-event basis.

**Figure 7-55. GPMC\_IRQENABLE**

|          |    |                              |                              |    |
|----------|----|------------------------------|------------------------------|----|
| 31       |    |                              |                              | 16 |
| Reserved |    |                              |                              |    |
|          |    |                              | R-0                          |    |
| 15       | 10 | 9                            | 8                            |    |
| Reserved |    | WAIT1EDGE<br>DETECTIONENABLE | WAIT0EDGE<br>DETECTIONENABLE |    |
| R-0      |    | R/W-0                        | R/W-0                        |    |
| 7        | 2  | 1                            | 0                            |    |
| Reserved |    | TERMINAL<br>COUNTENABLE      | FIFOEVENT<br>ENABLE          |    |
| R-0      |    | R/W-0                        | R/W-0                        |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-59. GPMC\_IRQENABLE Field Descriptions**

| Bit   | Field                    | Value | Description   |
|-------|--------------------------|-------|---|
| 31-10 | Reserved                 | 0     | Reserved  |
| 9     | WAIT1EDGEDETECTIONENABLE | 0     | Enables the Wait1 Edge Detection interrupt                                      |
|       |                          | 1     | Wait1EdgeDetection interrupt is masked  |
|       |                          | 1     | Wait1EdgeDetection event generates an interrupt if occurs                       |
| 8     | WAIT0EDGEDETECTIONENABLE | 0     | Enables the Wait0 Edge Detection interrupt                                      |
|       |                          | 1     | Wait0EdgeDetection interrupt is masked  |
|       |                          | 1     | Wait0EdgeDetection event generates an interrupt if occurs                       |
| 7-2   | Reserved                 | 0     | Reserved  |
| 1     | TERMINALCOUNTEVENTENABLE | 0     | Enables TerminalCountEvent interrupt issuing in pre-fetch or write posting mode |
|       |                          | 1     | TerminalCountEvent interrupt is masked  |
|       |                          | 1     | TerminalCountEvent interrupt is not masked                                      |
| 0     | FIFOEVENTENABLE          | 0     | Enables the FIFOEvent interrupt   |
|       |                          | 1     | FIFOEvent interrupt is masked   |
|       |                          | 1     | FIFOEvent interrupt is not masked   |

### 7.1.5.6 GPMC\_TIMEOUT\_CONTROL

The GPMC\_TIMEOUT\_CONTROL register allows the user to set the start value of the timeout counter

**Figure 7-56. GPMC\_TIMEOUT\_CONTROL**

|          |          |    |                   |  |   |          |   |               |  |   |    |
|----------|----------|----|-------------------|--|---|----------|---|---------------|--|---|----|
| 31       | Reserved |    |                   |  |   |          |   |               |  |   | 16 |
|          | R-0      |    |                   |  |   |          |   |               |  |   |    |
| 15       | 13       | 12 |                   |  | 4 | 3        | 1 |               |  | 0 |    |
| Reserved |          |    | TIMEOUTSTARTVALUE |  |   | Reserved |   | TIMEOUTENABLE |  |   |    |
| R-0      |          |    | R/W-0             |  |   | R-0      |   | R/W-0         |  |   |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-60. GPMC\_TIMEOUT\_CONTROL Field Descriptions**

| Bit   | Field             | Value  | Description   |
|-------|-------------------|--------|---|
| 31-13 | Reserved          | 0      | Reserved  |
| 12-4  | TIMEOUTSTARTVALUE | 0-1FFh | Start value of the time-out counter (000 corresponds to 0 GPMC.FCLK cycle, 1h corresponds to 1 GPMC.FCLK cycle, and 1FFh corresponds to 511 GPMC.FCLK cycles) |
| 3-1   | Reserved          | 0      | Reserved  |
| 0     | TIMEOUTENABLE     | 0<br>1 | Enable bit of the TimeOut feature<br>TimeOut feature is disabled<br>TimeOut feature is enabled  |

### 7.1.5.7 GPMC\_ERR\_ADDRESS

The GPMC\_ERR\_ADDRESS register stores the address of the illegal access when an error occurs.

**Figure 7-57. GPMC\_ERR\_ADDRESS**

|      |       |            |   |
|------|-------|------------|---|
| 31   | 30    |            | 0 |
| Rsvd |       | ILLEGALADD |   |
| R-0  | R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-61. GPMC\_ERR\_ADDRESS Field Descriptions**

| Bit  | Field      | Value        | Description  |
|------|------------|--------------|--|
| 31   | Reserved   | 0            | Reserved   |
| 30-0 | ILLEGALADD | 0-7FFF FFFFh | Address of illegal access: A30 (0 for memory region, 1 for GPMC register region) and A29-A0 (1GByte maximum) |

### 7.1.5.8 GPMC\_ERR\_TYPE

The GPMC\_ERR\_TYPE register stores the type of error when an error occurs.

**Figure 7-58. GPMC\_ERR\_TYPE**

|    |          |                  |                   |               |          |            |  |       |
|----|----------|------------------|-------------------|---------------|----------|------------|--|-------|
| 31 | Reserved |                  |                   |               |          |            |  | 16    |
|    | R-0      |                  |                   |               |          |            |  |       |
| 15 | Reserved |                  |                   |               |          |            |  | 8     |
|    | R-0      |                  |                   |               |          |            |  | R/W-0 |
| 7  | 5        | 4                | 3                 | 2             | 1        | 0          |  |       |
|    | Reserved | ERRORNOT SUPPADD | ERRORNOT SUPPMCMD | ERROR TIMEOUT | Reserved | ERRORVALID |  |       |
|    | R-0      | R/W-0            | R/W-0             | R/W-0         | R-0      | R/W-0      |  |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-62. GPMC\_ERR\_TYPE Field Descriptions**

| Bit   | Field            | Value  | Description  |
|-------|------------------|--------|--|
| 31-11 | Reserved         | 0      | Reserved   |
| 10-8  | ILLEGALMCMD      | 0-7h   | System Command of the transaction that caused the error  |
| 7-5   | Reserved         | 0      | Reserved   |
| 4     | ERRORNOTSUPPADD  | 0<br>1 | Not supported Address error<br>No error occurs<br>The error is due to a non supported Address  |
| 3     | ERRORNOTSUPPMCMD | 0<br>1 | Not supported Command error<br>No error occurs<br>The error is due to a non supported Command  |
| 2     | ERRORTIMEOUT     | 0<br>1 | Time-out error<br>No error occurs<br>The error is due to a time out  |
| 1     | Reserved         | 0      | Reserved   |
| 0     | ERRORVALID       | 0<br>1 | Error validity status - Must be explicitly cleared with a write 1 transaction<br>All error fields no longer valid<br>Error detected and logged in the other error fields |

### 7.1.5.9 GPMC\_CONFIG

The configuration register allows global configuration of the GPMC.

**Figure 7-59. GPMC\_CONFIG**

|          |          |                  |          |                    |                          |   |
|----------|----------|------------------|----------|--------------------|--------------------------|---|
| 31       | Reserved |                  |          |                    | 16                       |   |
| R-0      |          |                  |          |                    |                          |   |
| 15       | Reserved |                  | 10       | 9                  | 8                        |   |
| R-0      |          |                  | R/W-0    |                    | R/W-0                    |   |
| 7        | 5        | 4                | 3        | 2                  | 1                        | 0 |
| Reserved |          | WRITE<br>PROTECT | Reserved | LIMITED<br>ADDRESS | NANDFORCE<br>POSTEDWRITE |   |
| R-0      |          | R/W-0            | R-0      | R/W-0              | R/W-0                    |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-63. GPMC\_CONFIG Field Descriptions**

| Bit   | Field                | Value  | Description   |
|-------|----------------------|--------|---|
| 31-10 | Reserved             | 0      | Reserved  |
| 9     | WAIT1PINPOLARITY     | 0<br>1 | Selects the polarity of input pin WAIT1<br>WAIT1 active low<br>WAIT1 active high  |
| 8     | WAIT0PINPOLARITY     | 0<br>1 | Selects the polarity of input pin WAIT0<br>WAIT0 active low<br>WAIT0 active high  |
| 7-5   | Reserved             | 0      | Reserved  |
| 4     | WRITEPROTECT         | 0<br>1 | Controls the $\overline{WP}$ output pin level<br>$\overline{WP}$ output pin is low<br>$\overline{WP}$ output pin is high                |
| 3-2   | Reserved             | 0      | Reserved  |
| 1     | LIMITEDADDRESS       | 0<br>1 | Limited Address device support<br>No effect. GPMC controls all addresses.<br>A26-A11 are not modified during an external memory access. |
| 0     | NANDFORCEPOSTEDWRITE | 0<br>1 | Enables the Force Posted Write feature to NAND Cmd/Add/Data location<br>Disables Force Posted Write<br>Enables Force Posted Write       |

### **7.1.5.10 GPMC\_STATUS**

The status register provides global status bits of the GPMC.

**Figure 7-60. GPMC\_STATUS**

|     |          |    |                              |
|-----|----------|----|------------------------------|
| 31  | Reserved |    | 16                           |
| R-0 |          |    |                              |
| 15  | Reserved | 10 | 9 8                          |
|     |          |    | WAIT1STATUS   WAIT0STATUS    |
| 7   | R-0      |    | R/W-0 R/W-0                  |
|     | Reserved |    | 1 0                          |
|     | R-0      |    | EMPTYWRITEBUFFERSTATUS R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-64. GPMC\_STATUS Field Descriptions**

| Bit   | Field                  | Value  | Description  |
|-------|------------------------|--------|--|
| 31-10 | Reserved               | 0      | Reserved   |
| 9     | WAIT1STATUS            | 0<br>1 | Is a copy of input pin WAIT1. (Reset value is WAIT1 input pin sampled at IC reset)<br>WAIT1 asserted (inactive state)<br>WAIT1 de-asserted |
| 8     | WAIT0STATUS            | 0<br>1 | Is a copy of input pin WAIT0. (Reset value is WAIT0 input pin sampled at IC reset)<br>WAIT0 asserted (inactive state)<br>WAIT0 de-asserted |
| 7-1   | Reserved               | 0      | Reserved   |
| 0     | EMPTYWRITEBUFFERSTATUS | 0<br>1 | Stores the empty status of the write buffer<br>Write Buffer is not empty<br>Write Buffer is empty  |

### 7.1.5.11 GPMC\_CONFIG1\_i

The configuration 1 register sets signal control parameters per chip select.

**Figure 7-61. GPMC\_CONFIG1\_i**

| 31                       | 30                 | 29                  | 28            | 27                 | 26                | 25                       | 24    |
|--------------------------|--------------------|---------------------|---------------|--------------------|-------------------|--------------------------|-------|
| WRAPBURST                | READMULTIPLE       | READYTYPE           | WRITEMULTIPLE | WRITETYPE          | CLKACTIVATIONTIME | ATTACHEDDEVICEPAGELENGTH |       |
| R/W-0                    | R/W-0              | R/W-0               | R/W-0         | R/W-0              | R/W-0             | R/W-0                    | R/W-0 |
| 23                       | 22                 | 21                  | 20            | 19                 | 18                | 17                       | 16    |
| ATTACHEDDEVICEPAGELENGTH | WAITREADMONITORING | WAITWRITEMONITORING | Reserved      | WAITMONITORINGTIME | WAITPINSELECT     |                          |       |
| R/W-0                    | R/W-0              | R/W-0               | R-0           | R/W-0              |                   | R/W-0                    |       |
| 15                       | 14                 | 13                  | 12            | 11                 | 10                | 9                        | 8     |
| Reserved                 |                    | DEVICESIZE          |               | DEVICETYPE         |                   | MUXADDDATA               |       |
| R-0                      |                    | R/W-0               |               | R/W-0              |                   | R/W-0                    |       |
| 7                        |                    | 5                   | 4             | 3                  | 2                 | 1                        | 0     |
| Reserved                 |                    | TIMEPARAGRANULARITY |               | Reserved           |                   | GPMCFCLKDIVIDER          |       |
| R-0                      |                    | R/W-0               |               | R-0                |                   | R/W-0                    |       |

LEGEND: R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 7-65. GPMC\_CONFIG1\_i Field Descriptions**

| Bit   | Field                    | Value               | Description  |
|-------|--------------------------|---------------------|--|
| 31    | WRAPBURST                | 0<br>1              | Enables the wrapping burst capability. Must be set if the attached device is configured in wrapping burst<br>Synchronous wrapping burst not supported<br>Synchronous wrapping burst supported  |
| 30    | READMULTIPLE             | 0<br>1              | Selects the read single or multiple access<br>single access<br>multiple access (burst if synchronous, page if asynchronous)  |
| 29    | READYTYPE                | 0<br>1              | Selects the read mode operation<br>Read Asynchronous<br>Read Synchronous   |
| 28    | WRITEMULTIPLE            | 0<br>1              | Selects the write single or multiple access<br>Single access<br>Multiple access (burst if synchronous, considered as single if asynchronous)   |
| 27    | WRITETYPE                | 0<br>1              | Selects the write mode operation<br>Write Asynchronous<br>Write Synchronous  |
| 26-25 | CLKACTIVATIONTIME        | 0<br>1h<br>2h<br>3h | Output GPMC.CLK activation time<br>First rising edge of GPMC_CLK at start access time<br>First rising edge of GPMC_CLK one GPMC_FCLK cycle after start access time<br>First rising edge of GPMC_CLK two GPMC_FCLK cycles after start access time<br>Reserved |
| 24-23 | ATTACHEDDEVICEPAGELENGTH | 0<br>1h<br>2h<br>3h | Specifies the attached device page (burst) length (1 Word = Interface size)<br>4 Words<br>8 Words<br>16 Words<br>Reserved  |

**Table 7-65. GPMC\_CONFIG1\_i Field Descriptions (continued)**

| Bit   | Field               | Value               | Description  |
|-------|---------------------|---------------------|--|
| 22    | WAITREADMONITORING  | 0<br>1              | Selects the Wait monitoring configuration for Read accesses<br>WAIT pin is not monitored for read accesses<br>WAIT pin is monitored for read accesses  |
| 21    | WAITWRITEMONITORING | 0<br>1              | Selects the Wait monitoring configuration for Write accesses<br>WAIT pin is not monitored for write accesses<br>WAIT pin is monitored for write accesses   |
| 20    | Reserved            | 0                   | Reserved   |
| 19-18 | WAITMONITORINGTIME  | 0<br>1h<br>2h<br>3h | Selects input pin Wait monitoring time<br>WAIT pin is monitored with valid data<br>WAIT pin is monitored one GPMC_CLK cycle before valid data<br>WAIT pin is monitored two GPMC_CLK cycle before valid data<br>Reserved  |
| 17-16 | WAITPINSELECT       | 0<br>1h<br>2h<br>3h | Selects the input WAIT pin for this chip select<br>WAIT input pin is WAIT0<br>WAIT input pin is WAIT1<br>Reserved<br>Reserved  |
| 15-14 | Reserved            | 0                   | Reserved   |
| 13-12 | DEVICESIZE          | 0<br>1h<br>2h<br>3h | Selects the device size attached.<br>8 bit<br>16 bit<br>Reserved<br>Reserved   |
| 11-10 | DEVICETYPE          | 0<br>1h<br>2h<br>3h | Selects the attached device type<br>NOR Flash like, asynchronous and synchronous devices<br>Reserved<br>NAND Flash like devices, stream mode<br>Reserved   |
| 9-8   | MUXADDDATA          | 0<br>1h<br>2h<br>3h | Enables the Address and data multiplexed protocol (Reset value is SYSBOOT[11:10] input pin sampled at IC reset for CS[0] and 0 for CS[1-6])<br>Non-multiplexed attached device<br>AAD-multiplexed protocol device<br>Address and data multiplexed attached device<br>Reserved  |
| 7-5   | Reserved            | 0                   | Reserved   |
| 4     | TIMEPARAGRANULARITY | 0<br>1              | Signals timing latencies scalar factor (Rd/WRCycleTime, AccessTime, PageBurstAccessTime, CSOnTime, CSRd/WrOffTime, ADVOnTime, ADVRd/WrOffTime, OEOOnTime, OEOFFTime, WEOOnTime, WEOFFTime, Cycle2CycleDelay, BusTurnAround, TimeOutStartValue)<br>x1 latencies<br>x2 latencies |
| 3-2   | Reserved            | 0                   | Reserved   |
| 1-0   | GPMCFCLKDIVIDER     | 0<br>1h<br>2h<br>3h | Divides the GPMC.FCLK clock<br>GPMC_CLK frequency = GPMC_FCLK frequency<br>GPMC_CLK frequency = GPMC_FCLK frequency/2<br>GPMC_CLK frequency = GPMC_FCLK frequency/3<br>GPMC_CLK frequency = GPMC_FCLK frequency/4  |

### 7.1.5.12 GPMC\_CONFIG2\_i

Chip-select signal timing parameter configuration.

**Figure 7-62. GPMC\_CONFIG2\_i**

|          |             |               |          |          |                      |
|----------|-------------|---------------|----------|----------|----------------------|
| 31       | Reserved    |               | 21       | 20       | 16                   |
| R-0      |             |               |          |          | CSWROFFTIME<br>R/W-0 |
| 15       | 13          | 12            | 8        | 7        | 6 4 3 0              |
| Reserved | CSRDOFFTIME | CSEXTTRADELAY | Reserved | Reserved | CSONTIME             |
| R-0      | R/W-0       | R/W-0         | R-0      | R-0      | R/W-0                |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-66. GPMC\_CONFIG2\_i Field Descriptions**

| Bit   | Field         | Value               | Description  |
|-------|---------------|---------------------|--|
| 31-21 | Reserved      | 0                   | Reserved   |
| 20-16 | CSWROFFTIME   | 0<br>1h<br>⋮<br>1Fh | CS# de-assertion time from start cycle time for write accesses<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>31 GPMC_FCLK cycles                   |
| 15-13 | Reserved      | 0                   | Reserved   |
| 12-8  | CSRDOFFTIME   | 0<br>1h<br>⋮<br>1Fh | CS# de-assertion time from start cycle time for read accesses<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>31 GPMC_FCLK cycles                    |
| 7     | CSEXTTRADELAY | 0<br>1              | CS# Add Extra Half GPMC.FCLK cycle<br>CS i Timing control signal is not delayed<br>CS i Timing control signal is delayed of half GPMC_FCLK clock cycle |
| 6-4   | Reserved      | 0                   | Reserved   |
| 3-0   | CSONTIME      | 0<br>1h<br>⋮<br>1Fh | CS# assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>15 GPMC_FCLK cycles   |

### 7.1.5.13 GPMC\_CONFIG3\_i

ADV# signal timing parameter configuration.

**Figure 7-63. GPMC\_CONFIG3\_i**

|                   |                    |    |              |                    |    |
|-------------------|--------------------|----|--------------|--------------------|----|
| 31                | 30                 | 28 | 27           | 26                 | 24 |
| Reserved          | ADVAADMUXWROFFTIME |    | Reserved     | ADVAADMUXRDOFFTIME |    |
| R-0               | R/W-0              |    | R-0          | R/W-0              |    |
| 23                | 21                 | 20 |              |                    | 16 |
| Reserved          |                    |    | ADVWROFFTIME |                    |    |
|                   | R/W-0              |    | R/W-0        |                    |    |
| 15                | 13                 | 12 |              |                    | 8  |
| Reserved          |                    |    | ADVRDOFFTIME |                    |    |
|                   | R-0                |    | R/W-0        |                    |    |
| 7                 | 6                  | 4  | 3            |                    | 0  |
| ADVEXTRA<br>DELAY | ADVAADMUXONTIME    |    |              | ADVONTIME          |    |
| R/W-0             | R/W-0              |    |              | R/W-0              |    |

LEGEND: R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 7-67. GPMC\_CONFIG3\_i Field Descriptions**

| Bit   | Field              | Value               | Description   |
|-------|--------------------|---------------------|---|
| 31    | Reserved           | 0                   | Reserved  |
| 30-28 | ADVAADMUXWROFFTIME | 0<br>1h<br>:<br>7h  | ADV# de-assertion for first address phase when using the AAD-Mux protocol<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>7 GPMC_FCLK cycles        |
| 27    | Reserved           | 0                   | Reserved  |
| 26-24 | ADVAADMUXRDOFFTIME | 0<br>1h<br>:<br>7h  | ADV# assertion for first address phase when using the AAD-Mux protocol<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>7 GPMC_FCLK cycles           |
| 23-21 | Reserved           | 0                   | Reserved  |
| 20-16 | ADVWROFFTIME       | 0<br>1h<br>:<br>1Fh | ADV# de-assertion time from start cycle time for write accesses<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                 |
| 15-13 | Reserved           | 0                   | Reserved  |
| 12-8  | ADVRDOFFTIME       | 0<br>1h<br>:<br>1Fh | ADV# de-assertion time from start cycle time for read accesses<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                  |
| 7     | ADVEXTRA<br>DELAY  | 0<br>1              | ADV# Add Extra Half GPMC.FCLK cycle<br>ADV Timing control signal is not delayed<br>ADV Timing control signal is delayed of half GPMC_FCLK clock cycle |

**Table 7-67. GPMC\_CONFIG3\_i Field Descriptions (continued)**

| Bit | Field           | Value              | Description   |
|-----|-----------------|--------------------|---|
| 6-4 | ADVAADMUXONTIME | 0<br>1h<br>⋮<br>7h | ADV# assertion for first address phase when using the AAD-Multiplexed protocol<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>7 GPMC_FCLK cycles |
| 3-0 | ADVONTIME       | 0<br>1h<br>⋮<br>Fh | ADV# assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>15 GPMC_FCLK cycles                                     |

### 7.1.5.14 GPMC\_CONFIG4\_i

WE# and OE# signals timing parameter configuration.

**Figure 7-64. GPMC\_CONFIG4\_i**

|                  |               |           |          |       |  |
|------------------|---------------|-----------|----------|-------|--|
| 31               | 29            | 28        |          | 24    |  |
| Reserved         |               | WEOFETIME |          |       |  |
| R-0              |               |           |          | R/W-0 |  |
| 23               | 22            | 20        | 19       | 16    |  |
| WEEXTRADELAY     | Reserved      |           | WEONTIME |       |  |
| R/W-0            | R-0           |           | R/W-0    |       |  |
| 15               | 13            | 12        |          | 8     |  |
| OEADMUXOFFTIME   |               | OEOFETIME |          |       |  |
| R/W-0            |               | R/W-0     |          |       |  |
| 7                | 6             | 4         | 3        | 0     |  |
| OEEXTRA<br>DELAY | OEADMUXONTIME |           | OEONTIME |       |  |
| R/W-0            | R/W-0         |           | R/W-0    |       |  |

LEGEND: R = Read only; W1C = Write 1 to clear bit; -n = value after reset

**Table 7-68. GPMC\_CONFIG4\_i Field Descriptions**

| Bit   | Field          | Value               | Description   |
|-------|----------------|---------------------|---|
| 31-29 | Reserved       | 0                   | Reserved  |
| 28-24 | WEOFETIME      | 0<br>1h<br>:<br>1Fh | WE# de-assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                                   |
| 23    | WEEXTRADELAY   | 0<br>1              | WE# Add Extra Half GPMC.FCLK cycle<br>WE Timing control signal is not delayed<br>WE Timing control signal is delayed of half GPMC_FCLK clock cycle  |
| 22-20 | Reserved       | 0                   | Reserved  |
| 19-16 | WEONTIME       | 0<br>1h<br>:<br>Fh  | WE# assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>15 GPMC_FCLK cycles                                      |
| 15-13 | OEADMUXOFFTIME | 0<br>1h<br>:<br>7h  | OE# de-assertion time for the first address phase in an AAD-Multiplexed access<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>7 GPMC_FCLK cycles |
| 12-8  | OEOFETIME      | 0<br>1h<br>:<br>1Fh | OE# de-assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                                   |
| 7     | OEEXTRADELAY   | 0<br>1              | OE# Add Extra Half GPMC.FCLK cycle<br>OE Timing control signal is not delayed<br>OE Timing control signal is delayed of half GPMC_FCLK clock cycle  |

**Table 7-68. GPMC\_CONFIG4\_i Field Descriptions (continued)**

| Bit | Field          | Value              | Description  |
|-----|----------------|--------------------|--|
| 6-4 | OEAADMUXONTIME | 0<br>1h<br>⋮<br>7h | OE# assertion time for the first address phase in an AAD-Multiplexed access<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>7 GPMC_FCLK cycles |
| 3-0 | OEONTIME       | 0<br>1h<br>⋮<br>Fh | OE# assertion time from start cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>⋮<br>15 GPMC_FCLK cycles                                   |

### 7.1.5.15 GPMC\_CONFIG5\_i

RdAccessTime and CycleTime timing parameters configuration.

**Figure 7-65. GPMC\_CONFIG5\_i**

| 31       | 28                  | 27 | 24 | 23       | 21           | 20 | 16 |
|----------|---------------------|----|----|----------|--------------|----|----|
| Reserved | PAGEBURSTACCESSTIME |    |    | Reserved | RDACCESSTIME |    |    |
| R-0      | R/W-0               |    |    | R-0      | R/W-0        |    |    |
| 15       | 13                  | 12 | 8  | 7        | 5            | 4  | 0  |
| Reserved | WRCYCLETIME         |    |    | Reserved | RDCYCLETIME  |    |    |
| R-0      | R/W-0               |    |    | R-0      | R/W-0        |    |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-69. GPMC\_CONFIG5\_i Field Descriptions**

| Bit   | Field               | Value               | Description   |
|-------|---------------------|---------------------|---|
| 31-28 | Reserved            | 0                   | Reserved  |
| 27-24 | PAGEBURSTACCESSTIME | 0<br>1h<br>:<br>Fh  | Delay between successive words in a multiple access<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>15 GPMC_FCLK cycles |
| 23-21 | Reserved            | 0                   | Reserved  |
| 20-16 | RDACCESSTIME        | 0<br>1h<br>:<br>1Fh | Delay between start cycle time and first data valid<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles |
| 15-13 | Reserved            | 0                   | Reserved  |
| 12-8  | WRCYCLETIME         | 0<br>1h<br>:<br>1Fh | Total write cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                              |
| 7-5   | Reserved            | 0                   | Reserved  |
| 4-0   | RDCYCLETIME         | 0<br>1h<br>:<br>1Fh | Total read cycle time<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>31 GPMC_FCLK cycles                               |

### 7.1.5.16 GPMC\_CONFIG6\_i

WrAccessTime, WrDataOnAdmuxBus, Cycle2Cycle, and BusTurnAround parameters configuration

**Figure 7-66. GPMC\_CONFIG6\_i**

| 31                      | 29                      | 28           | 24 | 23               | 20 | 19               | 16 |
|-------------------------|-------------------------|--------------|----|------------------|----|------------------|----|
| Reserved                |                         | WRACCESSTIME |    | Reserved         |    | WRDATAONADMUXBUS |    |
| R-0                     |                         | R/W-0        |    | R-0              |    | R/W-0            |    |
| 15                      |                         | 12           |    | 11               |    | 8                |    |
| Reserved                |                         |              |    | CYCLE2CYCLEDELAY |    |                  |    |
| R-0                     |                         |              |    | R/W-0            |    |                  |    |
| 7                       | 6                       | 5            | 4  | 3                |    | 0                |    |
| CYCLE2CYCLE<br>SAMECSEN | CYCLE2CYCLE<br>DIFFCSEN | Reserved     |    | BUSTURNAROUND    |    |                  |    |
| R/W-0                   | R/W-0                   | R-0          |    | R/W-0            |    |                  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-70. GPMC\_CONFIG6\_i Field Descriptions**

| Bit   | Field               | Value | Description  |
|-------|---------------------|-------|--|
| 31-29 | Reserved            | 0     | Reserved   |
| 28-24 | WRACCESSTIME        | 0     | Delay from StartAccessTime to the GPMC.FCLK rising edge corresponding to the GPMC.CLK rising edge used by the attached memory for the first data capture       |
|       |                     | 0     | 0 GPMC_FCLK cycle  |
|       |                     | 1h    | 1 GPMC_FCLK cycle  |
|       |                     | :     | :  |
|       |                     | 1Fh   | 31 GPMC_FCLK cycles  |
| 23-20 | Reserved            | 0     | Reserved   |
| 19-16 | WRDATAONADMUXBUS    | 0-Fh  | Specifies on which GPMC.FCLK rising edge the first data of the synchronous burst write is driven in the add/data multiplexed bus                               |
| 15-12 | Reserved            | 0     | Reserved   |
| 11-8  | CYCLE2CYCLEDELAY    | 0     | Chip select high pulse delay between two successive accesses   |
|       |                     | 0     | 0 GPMC_FCLK cycle  |
|       |                     | 1h    | 1 GPMC_FCLK cycle  |
|       |                     | :     | :  |
|       |                     | Fh    | 15 GPMC_FCLK cycles  |
| 7     | CYCLE2CYCLESAMECSEN | 0     | Add Cycle2CycleDelay between two successive accesses to the same chip-select (any access type)   |
|       |                     | 0     | No delay between the two accesses  |
|       |                     | 1     | Add CYCLE2CYCLEDELAY   |
| 6     | CYCLE2CYCLEDIFFCSEN | 0     | Add Cycle2CycleDelay between two successive accesses to a different chip-select (any access type)  |
|       |                     | 0     | No delay between the two accesses  |
|       |                     | 1     | Add CYCLE2CYCLEDELAY   |
| 5-4   | Reserved            | 0     | Reserved   |
| 3-0   | BUSTURNAROUND       | 0     | Bus turn around latency between two successive accesses to the same chip-select (read to write) or to a different chip-select (read to read and read to write) |
|       |                     | 0     | 0 GPMC_FCLK cycle  |
|       |                     | 1h    | 1 GPMC_FCLK cycle  |
|       |                     | :     | :  |
|       |                     | Fh    | 15 GPMC_FCLK cycles  |

### 7.1.5.17 GPMC\_CONFIG7\_i

Chip-select address mapping configuration.

**Figure 7-67. GPMC\_CONFIG7\_i**

|          |          |             |      |         |   |             |   |  |    |
|----------|----------|-------------|------|---------|---|-------------|---|--|----|
| 31       | Reserved |             |      |         |   |             |   |  | 16 |
|          | R-0      |             |      |         |   |             |   |  |    |
| 15       | 12       | 11          | 8    | 7       | 6 | 5           | 0 |  |    |
| Reserved |          | MASKADDRESS | Rsvd | CSVALID |   | BASEADDRESS |   |  |    |
| R-0      |          | R/W-0       | R-0  | R/W-0   |   | R/W-0       |   |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-71. GPMC\_CONFIG7\_i Field Descriptions**

| Bit   | Field       | Value                     | Description   |
|-------|-------------|---------------------------|---|
| 31-12 | Reserved    | 0                         | Reserved  |
| 11-8  | MASKADDRESS | 0<br>8h<br>Ch<br>Eh<br>Fh | Chip-select mask address. Values not listed must be avoided as they create holes in the chip-select address space.<br>0 Chip-select size of 256 Mbytes<br>8h Chip-select size of 128 Mbytes<br>Ch Chip-select size of 64 Mbytes<br>Eh Chip-select size of 32 Mbytes<br>Fh Chip-select size of 16 Mbytes |
| 7     | Reserved    | 0                         | Reserved  |
| 6     | CSVALID     | 0<br>1                    | Chip-select enable (reset value is 1 for CS[0] and 0 for CS[1-5]).<br>0 CS disabled<br>1 CS enabled   |
| 5-0   | BASEADDRESS | 0-3Fh                     | Chip-select base address.<br>CSi base address where i = 0 to 3 (16 Mbytes minimum granularity). Bits [5-0] corresponds to A29, A28, A27, A26, A25, and A24.   |

### 7.1.5.18 GPMC\_NAND\_COMMAND\_i

This register is not a true register, just an address location.

**Figure 7-68. GPMC\_NAND\_COMMAND\_i**

|                     |   |
|---------------------|---|
| 31                  | 0 |
| GPMC_NAND_COMMAND_i |   |
| W-0                 |   |

LEGEND: W = Write only; -n = value after reset

**Table 7-72. GPMC\_NAND\_COMMAND\_i Field Descriptions**

| Bit  | Field               | Value        | Description   |
|------|---------------------|--------------|---|
| 31-0 | GPMC_NAND_COMMAND_i | 0xFFFF FFFFh | Writing data at the GPMC_NAND_COMMAND_i location places the data as the NAND command value on the bus, using a regular asynchronous write access. |

### 7.1.5.19 GPMC\_NAND\_ADDRESS\_i

This register is not a true register, just an address location.

**Figure 7-69. GPMC\_NAND\_ADDRESS\_i**

|                     |   |
|---------------------|---|
| 31                  | 0 |
| GPMC_NAND_ADDRESS_i |   |
| W-0                 |   |

LEGEND: W = Write only; -n = value after reset

**Table 7-73. GPMC\_NAND\_ADDRESS\_i Field Descriptions**

| Bit  | Field               | Value        | Description   |
|------|---------------------|--------------|---|
| 31-0 | GPMC_NAND_ADDRESS_i | 0xFFFF FFFFh | Writing data at the GPMC_NAND_ADDRESS_i location places the data as the NAND partial address value on the bus, using a regular asynchronous write access. |

### 7.1.5.20 GPMC\_NAND\_DATA\_i

This register is not a true register, just an address location.

**Figure 7-70. GPMC\_NAND\_DATA\_i**

|                  |   |
|------------------|---|
| 31               | 0 |
| GPMC_NAND_DATA_i |   |
| R/W              |   |

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

**Table 7-74. GPMC\_NAND\_DATA\_i Field Descriptions**

| Bit  | Field            | Value        | Description   |
|------|------------------|--------------|---|
| 31-0 | GPMC_NAND_DATA_i | 0xFFFF FFFFh | Reading data from the GPMC_NAND_DATA_i location or from any location in the associated chip-select memory region activates an asynchronous read access. |

### 7.1.5.21 GPMC\_PREFETCH\_CONFIG1

**Figure 7-71. GPMC\_PREFETCH\_CONFIG1**

| 31               | 30                | 28              | 27                        | 26               | 24         |
|------------------|-------------------|-----------------|---------------------------|------------------|------------|
| Reserved         | CYCLOOPTIMIZATION |                 | ENABLEOPTIMIZED<br>ACCESS | ENGINECSSELECTOR |            |
| R-0              | R/W-0             |                 | R/W-0                     | R/W-0            |            |
| 23               | 22                | 20              | 19                        |                  | 16         |
| PFPWENROUNDROBIN | Reserved          |                 |                           | PFPWWIGHTEDPRIO  |            |
| R/W-0            | R-0               |                 |                           | R/W-0            |            |
| 15               | 14                |                 | FIFOTHRESHOLD             |                  | 8          |
| Reserved         |                   |                 | R/W-0                     |                  |            |
| R-0              |                   |                 |                           |                  |            |
| 7                | 6                 | 5               | 4                         | 3                | 2          |
| ENABLEENGINE     | Reserved          | WAITPINSELECTOR |                           | SYNCHROMODE      | DMAMODE    |
| R/W-0            | R-0               | R/W-0           |                           | R/W-0            | R/W-0      |
|                  |                   |                 |                           | Reserved         | ACCESSMODE |
|                  |                   |                 |                           | R-0              | R/W-0      |

LEGEND: R = Read only; -n = value after reset

**Table 7-75. GPMC\_PREFETCH\_CONFIG1 Field Descriptions**

| Bit   | Field                 | Value                                       | Description   |
|-------|-----------------------|---|---|
| 31    | Reserved              | 0   | Reserved  |
| 30-28 | CYCLOOPTIMIZATION     | 0<br>1h<br>:<br>7h                          | Define the number of GPMC.FCLK cycles to be subtracted from RdCycleTime, WrCycleTime, AccessTime, CSRdOffTime, CSWrOffTime, ADVRdOffTime, ADVWrOffTime, OEOffTime, WEOffTime<br>0 GPMC_FCLK cycle<br>1 GPMC_FCLK cycle<br>:<br>7 GPMC_FCLK cycles   |
| 27    | ENABLEOPTIMIZEDACCESS | 0<br>1                                      | Enables access cycle optimization<br>Access cycle optimization is disabled<br>Access cycle optimization is enabled  |
| 26-24 | ENGINECSSELECTOR      | 0<br>1h<br>2h<br>3h<br>4h<br>5h<br>6h<br>7h | Selects the CS where Prefetch Postwrite engine is active<br>CS0<br>CS1<br>CS2<br>CS3<br>CS4<br>CS5<br>CS6<br>CS7  |
| 23    | PFPWENROUNDROBIN      | 0<br>1                                      | Enables the PFPW RoundRobin arbitration<br>Prefetch Postwrite engine round robin arbitration is disabled<br>Prefetch Postwrite engine round robin arbitration is enabled  |
| 22-20 | Reserved              | 0   | Reserved  |
| 19-16 | PFPWWIGHTEDPRIO       | 0<br>1h<br>:<br>Fh                          | When an arbitration occurs between a direct memory access and a PFPW engine access, the direct memory access is always serviced. If the PFPWEnRoundRobin is enabled,<br>The next access is granted to the PFPW engine<br>The two next accesses are granted to the PFPW engine<br>:<br>The 16 next accesses are granted to the PFPW engine |
| 15    | Reserved              | 0   | Reserved  |

**Table 7-75. GPMC\_PREFETCH\_CONFIG1 Field Descriptions (continued)**

| Bit  | Field           | Value               | Description  |
|------|-----------------|---------------------|--|
| 14-8 | FIFOTHRESHOLD   | 0<br>1h<br>:<br>40h | Selects the maximum number of bytes read from the FIFO or written to the FIFO by the host on a DMA or interrupt request<br>0 byte<br>1 byte<br>:<br>64 bytes   |
| 7    | ENABLEENGINE    | 0<br>1              | Enables the Prefetch Postwrite engine<br>Prefetch Postwrite engine is disabled<br>Prefetch Postwrite engine is enabled   |
| 6    | Reserved        | 0                   | Reserved   |
| 5-4  | WAITPINSELECTOR | 0<br>1h<br>2h<br>3h | Select which WAIT pin edge detector should start the engine in synchronized mode<br>Selects Wait0EdgeDetection<br>Selects Wait1EdgeDetection<br>Reserved<br>Reserved   |
| 3    | SYNCHROMODE     | 0<br>1              | Selects when the engine starts the access to CS<br>Engine starts the access to CS as soon as STARTENGINE is set<br>Engine starts the access to CS as soon as STARTENGINE is set AND wait to non wait edge detection on the selected wait pin           |
| 2    | DMAMODE         | 0<br>1              | Selects interrupt synchronization or DMA request synchronization<br>Interrupt synchronization is enabled. Only interrupt line will be activated on FIFO threshold crossing.<br>DMA request synchronization is enabled. A DMA request protocol is used. |
| 1    | Reserved        | 0                   | Reserved   |
| 0    | ACCESSMODE      | 0<br>1              | Selects pre-fetch read or write posting accesses<br>Prefetch read mode<br>Write-posting mode   |

### 7.1.5.22 GPMC\_PREFETCH\_CONFIG2

**Figure 7-72. GPMC\_PREFETCH\_CONFIG2**

|                    |          |               |       |    |
|--------------------|----------|---------------|-------|----|
| 31                 | Reserved |               |       | 16 |
|                    | R-0      |               |       |    |
| 15      14      13 |          | TRANSFERCOUNT |       | 0  |
| Reserved           |          |               | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-76. GPMC\_PREFETCH\_CONFIG2 Field Descriptions**

| Bit   | Field         | Value                 | Description   |
|-------|---------------|-----------------------|---|
| 31-14 | Reserved      | 0                     | Reserved  |
| 13-0  | TRANSFERCOUNT | 0<br>1h<br>:<br>2000h | Selects the number of bytes to be read or written by the engine to the selected CS<br>0 byte<br>1 byte<br>:<br>8 Kbytes |

### 7.1.5.23 GPMC\_PREFETCH\_CONTROL

**Figure 7-73. GPMC\_PREFETCH\_CONTROL**

|    |          |   |       |    |
|----|----------|---|-------|----|
| 31 | Reserved |   |       | 16 |
|    | R-0      |   |       |    |
| 15 | Reserved | 1 | 0     |    |
|    | R-0      |   | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-77. GPMC\_PREFETCH\_CONTROL Field Descriptions**

| Bit  | Field       | Value                | Description  |
|------|-------------|----------------------|--|
| 31-1 | Reserved    | 0                    | Reserved   |
| 0    | STARTENGINE | R0<br>W0<br>R1<br>W1 | Resets the FIFO pointer and starts the engine<br>Engine is stopped<br>Stops the engine<br>Engine is running<br>Resets the FIFO pointer to 0 in prefetch mode and 40h in postwrite mode and starts the engine |

### 7.1.5.24 GPMC\_PREFETCH\_STATUS

**Figure 7-74. GPMC\_PREFETCH\_STATUS**

| 31       | 30          | 24 | 23         | 17 | 16                  |
|----------|-------------|----|------------|----|---------------------|
| Rsvd     | FIFOPOINTER |    | Reserved   |    | FIFOTHRESHOLDSTATUS |
| R-0      | R-0         |    | R-0        |    | R-0                 |
| 15       | 14          | 13 |            |    | 0                   |
| Reserved |             |    | COUNTVALUE |    |                     |
| R-0      |             |    | R-0        |    |                     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-78. GPMC\_PREFETCH\_STATUS Field Descriptions**

| Bit   | Field               | Value                 | Description   |
|-------|---------------------|-----------------------|---|
| 31    | Reserved            | 0                     | Reserved  |
| 30-24 | FIFOPOINTER         | 0<br>⋮<br>40h         | 0 byte available to be read or 0 free empty place to be written<br>⋮<br>64 bytes available to be read or 64 empty places to be written  |
| 23-17 | Reserved            | 0                     | Reserved  |
| 16    | FIFOTHRESHOLDSTATUS | 0<br>1                | Set when FIFOPointer exceeds FIFOThreshold value<br>FIFOPointer smaller or equal to FIFOThreshold. Writing to this bit has no effect<br>FIFOPointer greater than FIFOThreshold. Writing to this bit has no effect   |
| 15-14 | Reserved            | 0                     | Reserved  |
| 13-0  | COUNTVALUE          | 0<br>1h<br>⋮<br>2000h | Number of remaining bytes to be read or to be written by the engine according to the TransferCount value<br>0 byte remaining to be read or to be written<br>1 byte remaining to be read or to be written<br>⋮<br>8 Kbytes remaining to be read or to be written |

### 7.1.5.25 GPMC\_ECC\_CONFIG

**Figure 7-75. GPMC\_ECC\_CONFIG**

|          |          |              |       |             |       |           |       |
|----------|----------|--------------|-------|-------------|-------|-----------|-------|
| 31       | Reserved |              |       |             |       | 17        | 16    |
|          | R-0      |              |       |             |       |           | R/W-0 |
| 15       | 14       | 13           | 12    | 11          |       |           | 8     |
| Reserved |          | ECCBCHTSEL   |       | ECCWRAPMODE |       |           |       |
|          | R-0      |              | R/W-0 |             | R/W-0 |           |       |
| 7        | 6        | 4            | 3     | 1           |       | 0         |       |
| ECC16B   |          | ECCTOPSECTOR |       | ECCCS       |       | ECCENABLE |       |
| R/W-0    |          | R/W-0        |       | R/W-0       |       | R/W-0     |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-79. GPMC\_ECC\_CONFIG Field Descriptions**

| Bit   | Field        | Value                                    | Description   |
|-------|--------------|--|---|
| 31-17 | Reserved     | 0  | Reserved  |
| 16    | ECCALGORITHM | 0<br>1                                   | ECC algorithm used<br>Hamming code<br>BCH code  |
| 15-14 | Reserved     | 0  | Reserved  |
| 13-12 | ECCBCHTSEL   | 0<br>1h<br>2h<br>3h                      | Error correction capability used for BCH<br>Up to 4 bits error correction ( $t = 4$ )<br>Up to 8 bits error correction ( $t = 8$ )<br>Up to 16 bits error correction ( $t = 16$ )<br>Reserved |
| 11-8  | ECCWRAPMODE  | 0-Fh                                     | Spare area organization definition for the BCH algorithm. See the BCH syndrome/parity calculator module functional specification for more details   |
| 7     | ECC16B       | 0<br>1                                   | Selects an ECC calculated on 16 columns<br>ECC calculated on 8 columns<br>ECC calculated on 16 columns  |
| 6-4   | ECCTOPSECTOR | 0<br>1h<br>3h<br>:<br>7h                 | Number of sectors to process with the BCH algorithm<br>1 sector (512kB page)<br>2 sectors<br>4 sectors (2kB page)<br>8 sectors (4kB page)   |
| 3-1   | ECCCS        | 0<br>1h<br>2h<br>3h<br>4h<br>5h<br>6h-7h | Selects the Chip-select where ECC is computed<br>Chip-select 0<br>Chip-select 1<br>Chip-select 2<br>Chip-select 3<br>Chip-select 4<br>Chip-select 5<br>Reserved                               |
| 0     | ECCENABLE    | 0<br>1                                   | Enables the ECC feature<br>ECC disabled<br>ECC enabled  |

### 7.1.5.26 GPMC\_ECC\_CONTROL

**Figure 7-76. GPMC\_ECC\_CONTROL**

|     |          |          |   |          |            |   |  |  |    |
|-----|----------|----------|---|----------|------------|---|--|--|----|
| 31  | Reserved |          |   |          |            |   |  |  | 16 |
| R-0 |          |          |   |          |            |   |  |  |    |
| 15  | 9        | 8        | 7 | 4        | 3          | 0 |  |  |    |
|     | Reserved | ECCCLEAR |   | Reserved | ECCPOINTER |   |  |  |    |
|     | R-0      | R/W-0    |   | R-0      | R/W-0      |   |  |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-80. GPMC\_ECC\_CONTROL Field Descriptions**

| Bit  | Field      | Value   | Description  |
|------|------------|---|--|
| 31-9 | Reserved   | 0   | Reserved   |
| 8    | ECCCLEAR   | R0<br>W0<br>W1  | Clear all ECC result registers<br>All reads return 0<br>Ignored<br>Clears all ECC result registers   |
| 7-4  | Reserved   | 0   | Reserved   |
| 3-0  | ECCPOINTER | 0-Fh<br>0<br>1h<br>2h<br>3h<br>4h<br>5h<br>6h<br>7h<br>8h<br>9h | Selects ECC result register (Reads to this field give the dynamic position of the ECC pointer - Writes to this field select the ECC result register where the first ECC computation will be stored). Writing values not listed disables the ECC engine (ECCENABLE bit of GPMC_ECC_CONFIG cleared to 0).<br>Writing 0 disables the ECC engine (ECCENABLE bit of GPMC_ECC_CONFIG cleared to 0)<br>ECC result register 1 selected<br>ECC result register 2 selected<br>ECC result register 3 selected<br>ECC result register 4 selected<br>ECC result register 5 selected<br>ECC result register 6 selected<br>ECC result register 7 selected<br>ECC result register 8 selected<br>ECC result register 9 selected |

### 7.1.5.27 GPMC\_ECC\_SIZE\_CONFIG

**Figure 7-77. GPMC\_ECC\_SIZE\_CONFIG**

|                 |                 |                 |                 |                 |                 |                 |                 |       |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-------|
| 31              | 30              | 29              |                 | 22              | 21              | 20              | 19              | 16    |
| Reserved        |                 | ECCSIZE1        |                 | Reserved        |                 | ECCSIZE0        |                 |       |
| R-0             |                 | R/W-0           |                 | R-0             |                 | R/W-0           |                 |       |
| 15              |                 | 12              | 11              |                 | 9               | 8               |                 |       |
|                 | ECCSIZE0        |                 |                 | Reserved        |                 | ECC9RESULT SIZE |                 |       |
|                 |                 | R/W-0           |                 | R-0             |                 | R/W-0           |                 |       |
| 7               | 6               | 5               | 4               | 3               | 2               | 1               | 0               |       |
| ECC8RESULT SIZE | ECC7RESULT SIZE | ECC6RESULT SIZE | ECC5RESULT SIZE | ECC4RESULT SIZE | ECC3RESULT SIZE | ECC2RESULT SIZE | ECC1RESULT SIZE |       |
| R/W-0           | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-81. GPMC\_ECC\_SIZE\_CONFIG Field Descriptions**

| Bit   | Field          | Value                           | Description   |
|-------|----------------|---------------------------------|---|
| 31-30 | Reserved       | 0                               | Reserved  |
| 29-22 | ECCSIZE1       | 0<br>1h<br>2h<br>3h<br>:<br>FFh | Defines ECC size 1<br>0 2 Bytes<br>1h 4 Bytes<br>2h 6 Bytes<br>3h 8 Bytes<br>:<br>FFh 512 Bytes |
| 21-20 | Reserved       | 0                               | Reserved  |
| 19-12 | ECCSIZE0       | 0<br>1h<br>2h<br>3h<br>:<br>FFh | Defines ECC size 0<br>0 2 Bytes<br>1h 4 Bytes<br>2h 6 Bytes<br>3h 8 Bytes<br>:<br>FFh 512 Bytes |
| 11-9  | Reserved       | 0                               | Reserved  |
| 8     | ECC9RESULTSIZE | 0<br>1                          | Selects ECC size for ECC 9 result register<br>ECCSIZE0 selected<br>ECCSIZE1 selected            |
| 7     | ECC8RESULTSIZE | 0<br>1                          | Selects ECC size for ECC 8 result register<br>ECCSIZE0 selected<br>ECCSIZE1 selected            |
| 6     | ECC7RESULTSIZE | 0<br>1                          | Selects ECC size for ECC 7 result register<br>ECCSIZE0 selected<br>ECCSIZE1 selected            |
| 5     | ECC6RESULTSIZE | 0<br>1                          | Selects ECC size for ECC 6 result register<br>ECCSIZE0 selected<br>ECCSIZE1 selected            |
| 4     | ECC5RESULTSIZE | 0<br>1                          | Selects ECC size for ECC 5 result register<br>ECCSIZE0 selected<br>ECCSIZE1 selected            |

**Table 7-81. GPMC\_ECC\_SIZE\_CONFIG Field Descriptions (continued)**

| Bit | Field          | Value | Description   |
|-----|----------------|-------|---|
| 3   | ECC4RESULTSIZE | 0     | Selects ECC size for ECC 4 result register<br>ECCSIZE0 selected |
|     |                | 1     | ECCSIZE1 selected   |
| 2   | ECC3RESULTSIZE | 0     | Selects ECC size for ECC 3 result register<br>ECCSIZE0 selected |
|     |                | 1     | ECCSIZE1 selected   |
| 1   | ECC2RESULTSIZE | 0     | Selects ECC size for ECC 2 result register<br>ECCSIZE0 selected |
|     |                | 1     | ECCSIZE1 selected   |
| 0   | ECC1RESULTSIZE | 0     | Selects ECC size for ECC 1 result register<br>ECCSIZE0 selected |
|     |                | 1     | ECCSIZE1 selected   |

### 7.1.5.28 GPMC\_ECCj\_RESULT

**Figure 7-78. GPMC\_ECCj\_RESULT**

| 31       | 28     | 27     | 26    | 25    | 24    | 23   | 22   | 21   | 20  | 19  | 18  | 17  | 16  |
|----------|--------|--------|-------|-------|-------|------|------|------|-----|-----|-----|-----|-----|
| Reserved | P2048O | P1024O | P512O | P256O | P128O | P64O | P32O | P16O | P8O | P4O | P2O | P1O |     |
| R-0      | R-0    | R-0    | R-0   | R-0   | R-0   | R-0  | R-0  | R-0  | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15       | 12     | 11     | 10    | 9     | 8     | 7    | 6    | 5    | 4   | 3   | 2   | 1   | 0   |
| Reserved | P2048E | P1024E | P512E | P256E | P128E | P64E | P32E | P16E | P8E | P4E | P2E | P1E |     |
| R-0      | R-0    | R-0    | R-0   | R-0   | R-0   | R-0  | R-0  | R-0  | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-82. GPMC\_ECCj\_RESULT Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-28 | Reserved | 0     | Reserved  |
| 27    | P2048O   | 0-1   | Odd Row Parity bit 2048, only used for ECC computed on 512 Bytes  |
| 26    | P1024O   | 0-1   | Odd Row Parity bit 1024   |
| 25    | P512O    | 0-1   | Odd Row Parity bit 512  |
| 24    | P256O    | 0-1   | Odd Row Parity bit 256  |
| 23    | P128O    | 0-1   | Odd Row Parity bit 128  |
| 22    | P64O     | 0-1   | Odd Row Parity bit 64   |
| 21    | P32O     | 0-1   | Odd Row Parity bit 32   |
| 20    | P16O     | 0-1   | Odd Row Parity bit 16   |
| 19    | P8O      | 0-1   | Odd Row Parity bit 8  |
| 18    | P4O      | 0-1   | Odd Column Parity bit 4   |
| 17    | P2O      | 0-1   | Odd Column Parity bit 2   |
| 16    | P1O      | 0-1   | Odd Column Parity bit 1   |
| 15-12 | Reserved | 0     | Reserved  |
| 11    | P2048E   | 0-1   | Even Row Parity bit 2048, only used for ECC computed on 512 Bytes |
| 10    | P1024E   | 0-1   | Even Row Parity bit 1024  |
| 9     | P512E    | 0-1   | Even Row Parity bit 512   |
| 8     | P256E    | 0-1   | Even Row Parity bit 256   |
| 7     | P128E    | 0-1   | Even Row Parity bit 128   |
| 6     | P64E     | 0-1   | Even Row Parity bit 64  |
| 5     | P32E     | 0-1   | Even Row Parity bit 32  |
| 4     | P16E     | 0-1   | Even Row Parity bit 16  |
| 3     | P8E      | 0-1   | Even Row Parity bit 8   |
| 2     | P4E      | 0-1   | Even Column Parity bit 4  |
| 1     | P2E      | 0-1   | Even Column Parity bit 2  |
| 0     | P1E      | 0-1   | Even Column Parity bit 1  |

### 7.1.5.29 GPMC\_BCH\_RESULT0\_i

**Figure 7-79. GPMC\_BCH\_RESULT0\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT0_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-83. GPMC\_BCH\_RESULT0\_i Field Descriptions**

| Bit  | Field         | Value        | Description                  |
|------|---------------|--------------|------------------------------|
| 31-0 | BCH_RESULT0_i | 0-FFFF FFFFh | BCH ECC result, bits 0 to 31 |

### 7.1.5.30 GPMC\_BCH\_RESULT1\_i

**Figure 7-80. GPMC\_BCH\_RESULT1\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT1_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-84. GPMC\_BCH\_RESULT1\_i Field Descriptions**

| Bit  | Field         | Value        | Description                   |
|------|---------------|--------------|-------------------------------|
| 31-0 | BCH_RESULT1_i | 0-FFFF FFFFh | BCH ECC result, bits 32 to 63 |

### 7.1.5.31 GPMC\_BCH\_RESULT2\_i

**Figure 7-81. GPMC\_BCH\_RESULT2\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT2_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-85. GPMC\_BCH\_RESULT2\_i Field Descriptions**

| Bit  | Field         | Value        | Description                   |
|------|---------------|--------------|-------------------------------|
| 31-0 | BCH_RESULT2_i | 0-FFFF FFFFh | BCH ECC result, bits 64 to 95 |

### **7.1.5.32 GPMC\_BCH\_RESULT3\_i**

**Figure 7-82. GPMC\_BCH\_RESULT3\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT3_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-86. GPMC\_BCH\_RESULT3\_i Field Descriptions**

| Bit  | Field         | Value        | Description                    |
|------|---------------|--------------|--------------------------------|
| 31-0 | BCH_RESULT3_i | 0xFFFF FFFFh | BCH ECC result, bits 96 to 127 |

### **7.1.5.33 GPMC\_BCH\_SWDATA**

This register is used to directly pass data to the BCH ECC calculator without accessing the actual NAND flash interface.

**Figure 7-83. GPMC\_BCH\_SWDATA**

|       |          |    |
|-------|----------|----|
| 31    | Reserved | 16 |
| R-0   |          |    |
| 15    | BCH_DATA | 0  |
| R/W-0 |          |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-87. GPMC\_BCH\_SWDATA Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31-16 | Reserved | 0       | Reserved   |
| 15-0  | BCH_DATA | 0xFFFFh | Data to be included in the BCH calculation. Only bits 0 to 7 are taken into account, if the calculator is configured to use 8 bits data (GPMC_ECC_CONFIG[7] ECC16B = 0). |

### **7.1.5.34 GPMC\_BCH\_RESULT4\_i**

**Figure 7-84. GPMC\_BCH\_RESULT4\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT4_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-88. GPMC\_BCH\_RESULT4\_i Field Descriptions**

| Bit  | Field         | Value        | Description                     |
|------|---------------|--------------|---------------------------------|
| 31-0 | BCH_RESULT4_i | 0xFFFF FFFFh | BCH ECC result, bits 128 to 159 |

### 7.1.5.35 GPMC\_BCH\_RESULT5\_i

**Figure 7-85. GPMC\_BCH\_RESULT5\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT5_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-89. GPMC\_BCH\_RESULT5\_i Field Descriptions**

| Bit  | Field         | Value        | Description                     |
|------|---------------|--------------|---------------------------------|
| 31-0 | BCH_RESULT5_i | 0-FFFF FFFFh | BCH ECC result, bits 160 to 191 |

### 7.1.5.36 GPMC\_BCH\_RESULT6\_i

**Figure 7-86. GPMC\_BCH\_RESULT6\_i**

|       |               |   |
|-------|---------------|---|
| 31    | BCH_RESULT6_i | 0 |
| R/W-0 |               |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-90. GPMC\_BCH\_RESULT6\_i Field Descriptions**

| Bit  | Field         | Value        | Description                     |
|------|---------------|--------------|---------------------------------|
| 31-0 | BCH_RESULT6_i | 0-FFFF FFFFh | BCH ECC result, bits 192 to 207 |

## 7.2 OCMC-RAM

### 7.2.1 *Introduction*

#### 7.2.1.1 OCMC-RAM Features

The on-chip memory controller consists of two separate modules that are OCP to memory wrappers. The first wrapper is for a ROM; the second is for a RAM. Each wrapper has its own dedicated interface to the L3 interconnect.

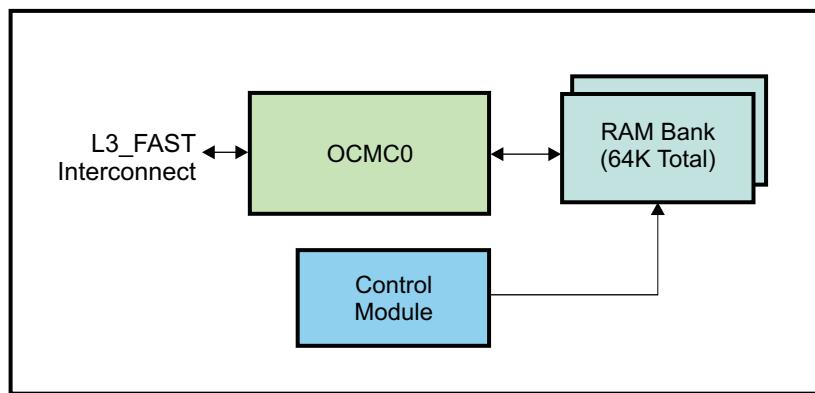
- 32- or 64-bit width
- Initial latency max 2 cycles (due to OCP to memory core wrapper).
- Multiple memory bank control based on address MSBs
- Full OCP IP 2.0 Burst support. No wait state.

#### 7.2.1.2 Unsupported OCMC-RAM Features

For this device, the OCMC-RAM implementation does not support parity.

### 7.2.2 Integration

This device includes a single instantiation of the on-chip memory controller interfacing to a single 64-K bank of RAM.



**Figure 7-87. OCMC RAM Integration**

#### 7.2.2.1 OCMC RAM Connectivity Attributes

The general connectivity attributes for the OCMC RAM modules are summarized in [Table 7-91](#).

**Table 7-91. OCMC RAM Connectivity Attributes**

| Attributes          | Type               |
|---------------------|--------------------|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L3_GCLK     |
| Reset Signals       | PER_DOM_RST_N      |
| Idle/Wakeup Signals | Smart Idle         |
| Interrupt Requests  | None               |
| DMA Requests        | None               |
| Physical Address    | L3 Fast slave port |

#### 7.2.2.2 OCMC RAM Clock and Reset Management

The OCMC module uses a single clock for the module and its OCP interface.

**Table 7-92. OCMC RAM Clock Signals**

| Clock Signal                                    | Max Freq | Reference / Source | Comments                    |
|---|----------|--------------------|-----------------------------|
| prcm_ocmc_clock<br>Interface / Functional clock | 200 MHz  | CORE_CLKOUTM4      | pd_per_l3_gclk<br>From PRCM |

#### 7.2.2.3 OCMC RAM Pin List

The OCMC RAM module does not include any external interface pins.

## 7.3 EMIF

### 7.3.1 *Introduction*

#### 7.3.1.1 Features

The general features of the EMIF module are as follows:

- 16-bit data path to external SDRAM memory
- One 128-bit OCPIP 2.2 interface
- Support for the following memory types:
  - mDDR (LPDDR1)
  - DDR2
  - DDR3

#### **External memory configurations supported:**

- Memory device capacity
  - Up to 1G Byte addressability
- Flexible bank/row/column/chip-select address multiplexing schemes
- Device driver strength feature for mobile DDR supported
- Supports following CAS latencies:
  - DDR2 => 2, 3, 4, 5, 6, and 7
  - DDR3 => 5, 6, 7, 8, 9, 10, and 11
  - mDDR => 2, 3, and 4
- Supports following number of internal banks:
  - DDR2 => 1, 2, 4, and 8
  - DDR3 => 1, 2, 4, and 8
  - mDDR => 1, 2, and 4
- Supports 256, 512, 1024, and 2048-word page sizes
- Supports burst length of 8 (sequential burst)
- Write/read leveling/calibration and data eye training in conjunction with DID
- Self Refresh and Power-Down modes for low power:
  - Flexible OCP to DDR address mapping to support Partial Array Self Refresh in LPDDR1, DDR2 and DDR3.
  - Temperature Controlled Self Refresh for LPDDR1 and DDR3 having on-chip temperature sensor.
- Periodic ZQ calibration for DDR3
- ODT on DDR2 and DDR3
- Prioritized refresh scheduling
- Programmable SDRAM refresh rate and backlog counter
- Programmable SDRAM timing parameters
- Big and little endian modes

### 7.3.1.2 Unsupported EMIF Features

The following EMIF4DC module features are not supported in this device.

**Table 7-93. Unsupported EMIF Features**

| Feature            | Reason                   |
|--------------------|--------------------------|
| 32-bit data        | Only 16 bits pinned out  |
| Multiple DDR banks | Only 1 CS/ODT pinned out |
| DDR2 CAS Latency 2 | Not supported by DID     |

## 7.3.2 Integration

### 7.3.2.1 EMIF Connectivity Attributes

The general connectivity attributes for the EMIF are shown in [Table 7-94](#).

**Table 7-94. EMIF Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain                               |
| Clock Domain        | PD_PER_L3_GCLK (OCP)<br>PD_PER_EMIF_GCLK (Func) |
| Reset Signals       | POR_N   |
| Idle/Wakeup Signals | Smart Idle                                      |
| Interrupt Requests  | 1 interrupt to MPU Subsystem (DDRERR0)          |
| DMA Requests        | None  |
| Physical Address    | L3 Fast Slave Port                              |

### 7.3.2.2 EMIF Clock Management

The EMIF4 OCP interface (ocp\_clk) is clocked by the L3 Fast clock sourced from the Core PLL. The DDR Command and Data macros are clocked by the DDR PLL. The PRCM divides this clock by two to create the EMIF functional clock (m\_clk).

The OCP and functional clocks may be asynchronous because synchronization is managed in the EMIF internal FIFO (EMIF4 is set in asynchronous mode).

**Table 7-95. EMIF Clock Signals**

| Clock Signal   | Maximum Frequency | Reference Source   | Comments                      |
|--|-------------------|--------------------|-------------------------------|
| ocp_clk<br>(Interface clock)   | 200 MHz           | CORE_CLKOUTM4      | pd_per_l3_gclk<br>From PRCM   |
| m_clk<br>(EMIF functional clock)   | 152 MHz           | DDR PLL CLKOUT / 2 | pd_per_emif_gclk<br>From PRCM |
| cmd0_dfi_clk<br>cmd1_dfi_clk<br>cmd2_dfi_clk<br>data0_dfi_clk<br>data1_dfi_clk<br>(Macro clocks) | 303 MHz           | DDR PLL CLKOUT     | clkout_po<br>From DDR PLL     |

### 7.3.2.3 EMIF Pin List

The EMIF/DDR external interface signals are shown in [Table 7-96](#).

**Table 7-96. EMIF Pin List**

| Pin               | Type | Description             |
|-------------------|------|-------------------------|
| DDR_CK<br>DDR_NCK | O    | Differential clock pair |
| DDR_CKE           | O    | Clock enable            |
| DDR_CS0           | O    | Chip select             |
| DDR_RASn          | O    | Row address strobe      |
| DDR_CASn          | O    | Column address strobe   |
| DDR_WEn           | O    | Write enable            |
| DDR_BA[2:0]       | O    | Bank address            |

**Table 7-96. EMIF Pin List (continued)**

| <b>Pin</b>    | <b>Type</b> | <b>Description</b>         |
|---------------|-------------|----------------------------|
| DDR_A[15:0]   | O           | Row/column address         |
| DDR_DQS[1:0]  | I/O         | Data strobes               |
| DDR_DQSn[1:0] | I/O         | Complimentary data strobes |
| DDR_DQM[1:0]  | O           | Data masks                 |
| DDR_D[15:0]   | I/O         | Data                       |
| DDR_ODT       | O           | On-die termination         |
| DDR_RESETn    | O           | DDR device reset           |
| DDR_VREF      | I           | I/O Voltage reference      |
| DDR_VTP       | I           | VTP compensation pin       |

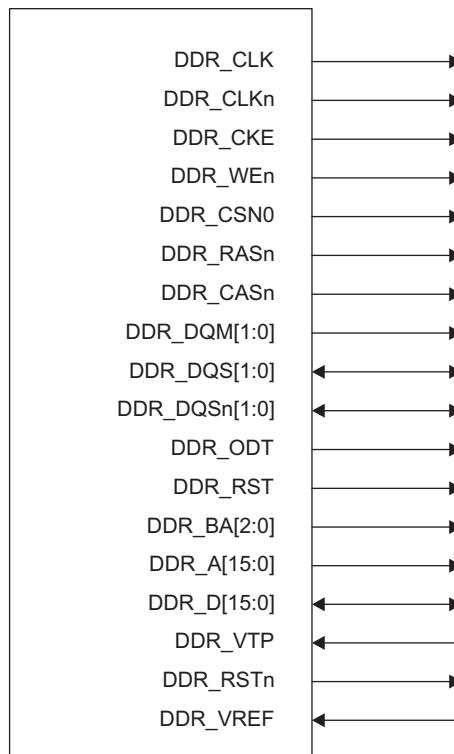
### 7.3.3 Functional Description

#### 7.3.3.1 Signal Descriptions

The DDR2/3/mDDR memory controller signals are shown in [Figure 7-88](#) and described in [Table 7-97](#). The following features are included:

- The maximum width for the data bus (DDR\_D[15:0]) is 16-bits
- The address bus (DDR\_A[15:0]) is 16-bits wide with an additional 3 bank address pins (DDR\_BA[2:0])
- Two differential output clocks (DDR\_CK and DDR\_nCK) driven by internal clock sources
- Command signals: Row and column address strobe (DDR\_RASn and DDR\_CASn), write enable strobe (DDR\_WEn), data strobe (DDR\_DQS[1:0] and DDR\_DQSn[1:0]), and data mask (DDR\_DQM[1:0]).
- One chip select signal (DDR\_CSN0) and one clock enable signal (DDR\_CKE)
- One on-die termination output signals (DDR\_ODT).

**Figure 7-88. DDR2/3/mDDR Memory Controller Signals**



**Table 7-97. DDR2/3/mDDR Memory Controller Signal Descriptions**

| Pin              | Description  |
|------------------|--|
| DDR_D[15:0]      | Bidirectional data bus. Input for data reads and output for data writes.                       |
| DDR_A[15:0]      | External address output.   |
| DDR_CSN0         | Chip select output.  |
| DDR_DQM[1:0]     | Active-low output data mask.   |
| DDR_CLK/DDR_CLKn | Differential clock outputs. All DDR2/3/mDDR interface signals are synchronous to these clocks. |
| DDR_CKE          | Clock enable. Used to select Power-Down and Self-Refresh operations.                           |
| DDR_CASn         | Active-low column address strobe.  |
| DDR_RASn         | Active-low row address strobe.   |
| DDR_WEn          | Active-low write enable.   |

**Table 7-97. DDR2/3/mDDR Memory Controller Signal Descriptions (continued)**

| Pin                        | Description  |
|----------------------------|--|
| DDR_DQS[1:0]/DDR_DQSn[1:0] | Differential data strobe bidirectional signals. Edge-aligned inputs on reads and center-aligned outputs on writes.                   |
| DDR_ODT                    | On-die termination signal to external DDR2/3 SDRAM. ODT is not supported for mDDR.   |
| DDR_BA[2:0]                | Bank-address control outputs.  |
| DDR_VREF                   | Memory Controller reference voltage. This voltage must be supplied externally. See the device-specific data manual for more details. |
| DDR_VTP                    | DDR2/3/mDDR VTP Compensation Resistor Connection.  |
| DDR_RESETn                 | Reset output. Asynchronous reset for DDR3 devices.   |

### 7.3.3.2 Clock Control

DDR2/3/mDDR clock is derived directly from the DDR PLL's VCO output. The frequency of DDR\_CLK can be determined by using the following formula:

$$\text{DDR\_CLK frequency} = (\text{DDRPLL input clock frequency} \times \text{multiplier}) / ((\text{pre-divider}+1) * \text{post-divider})$$

The second output clock of the DDR2/3/mDDR memory controller DDR\_CLKn, is the inverse of DDR\_CLK. You can change the multiplier, pre-divider and post-divider to get the desired DDR\_CLK frequency.

For detailed information on DDR PLL, see [Section 8.1, Power Management and Clock Module \(PRCM\)](#).

### 7.3.3.3 DDR2/3/mDDR Memory Controller Subsystem Overview

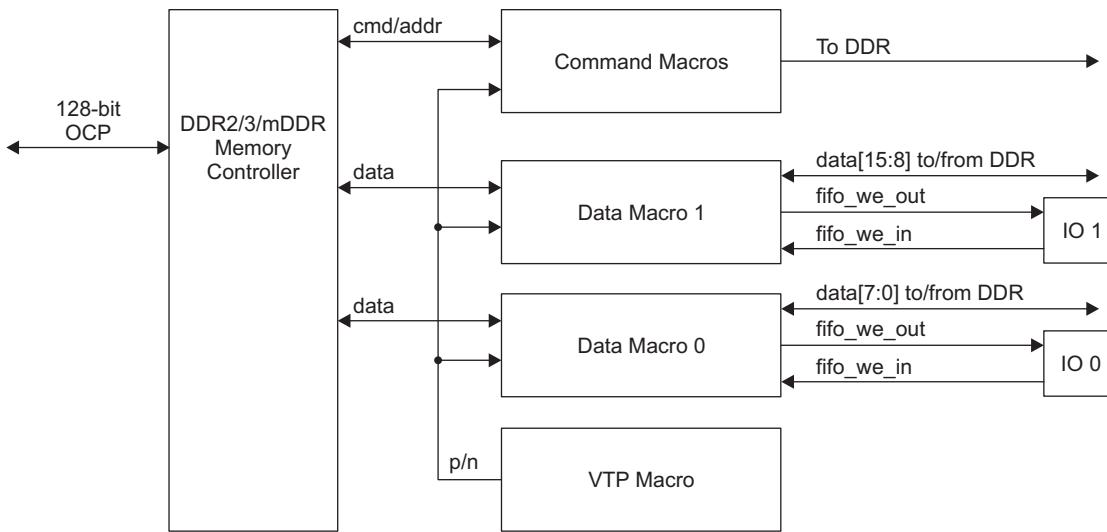
The DDR2/3/mDDR memory controller can gluelessly interface to most standard DDR2/3/mDDR SDRAM devices and supports such features as self-refresh mode and prioritized refresh. In addition, it provides flexibility through programmable parameters such as the refresh rate, CAS latency, and many SDRAM timing parameters. The DDR2/3/mDDR subsystem consists of the following:

- DDR2/3/mDDR memory controller
- Command macro
- Data macro
- VTP controller macro
- IOs for DQS gate

The subsystem supports JEDEC standard compliant DDR2/DDR3 and mDDR(LPDRR1) devices. It does not support CAS latency of 2 for DDR2 due to data and command macro limitations. It supports a 128-bit wide OCP interface on the core side for programmability. The subsystem can be used to connect to 16-bit memory devices.

[Figure 7-89](#) shows the DDR2/3/mDDR subsystem block diagram.

**Figure 7-89. DDR2/3/mDDR Subsystem Block Diagram**



Where

**fifo\_we\_out** = DQS enable output for timing match between DQS and system (Memory) clock.

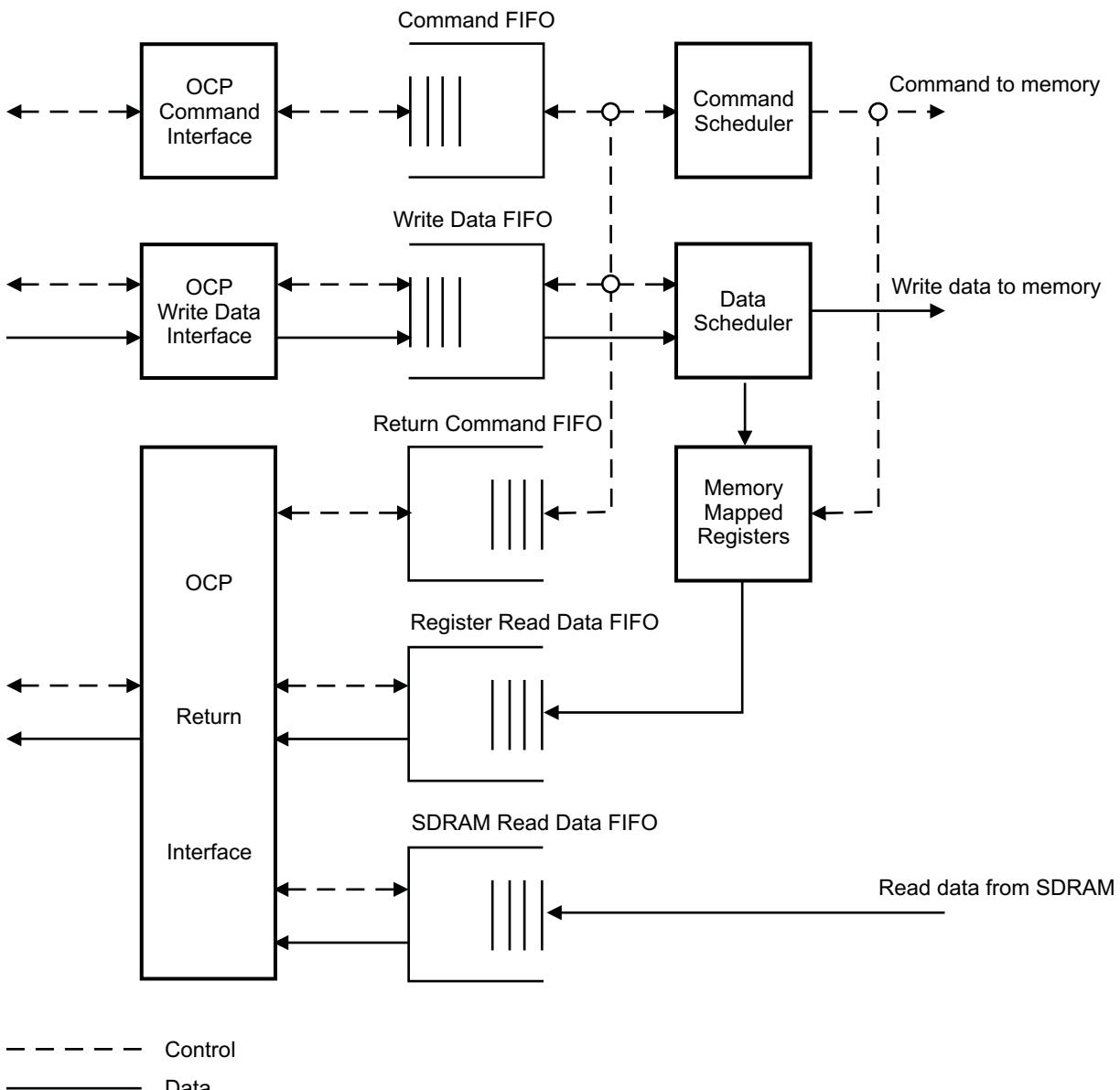
**fifo\_we\_in** = DQS enable input for timing match between DQS and system (Memory) clock.

#### 7.3.3.3.1 DDR2/3/mDDR Memory Controller Interface

To move data efficiently from on-chip resources to external DDR2/3/mDDR SDRAM device, the DDR2/3/mDDR memory controller makes use of a command FIFO, a write data FIFO, a return command FIFO, and two Read Data FIFOs. Purpose of each FIFO is described below.

Figure 7-90 shows the block diagram of the DDR2/3/mDDR memory controller FIFOs. Commands, write data, and read data arrive at the DDR2 memory controller parallel to each other. The same peripheral bus is used to write and read data from external memory as well as internal memory-mapped registers.

**Figure 7-90. DDR2/3/mDDR Memory Controller FIFO Block Diagram**



The command FIFO stores all the commands coming in on the OCP command interface.

The Write Data FIFO stores the write data for all the write transactions coming in on the OCP write data interface.

The Return Command FIFO stores all the return transactions that are to be issued to the OCP return interface. These include the write status return and the read data return commands.

There are two Read Data FIFOs that store the read data to be sent to the OCP return interface. One Read Data FIFO stores read data from the memory mapped registers and other Read Data FIFO stores read data from external memory.

### 7.3.3.3.2 Data Macro

The data macro consists of 8 data channels, one pair of complementary strobes (one pair for 8 bits of data), and one data mask channel (one for 8 bits of data).

The data macros consists of PHY Data Macro, DLLs and IOs integrated into a macro.

The data macro is a bidirectional interface. It is used to transmit data from the memory controller to the external memory chip during a write operation and receive data from memory and transmit it to the memory controller during a read operation.

During a write operation, the data macro translates 32/16-bit words from memory controller to 8-bit words and transmits them at double the bit rate to the memory along with the strobe. The strobe is center-aligned to the data. Data can be prevented from writing to the memory using data mask signal.

During a read operation, the data macro receives 8-bit DDR data along with the strobe and converts it to 32/16-bit words and transmits them to the memory controller along with the read-valid signals.

#### **7.3.3.3.3 Command Macro**

It consists of PHY Command Macro, DLLs and the IOs integrated together. The command macro acts as a unidirectional macro that transmits address and control bits from memory controller to the memory chip. The clocks DDR\_CLK and DDR\_CLKn are used by the memory to register the command and address transmitted on the transmit channels. All address and control signals are transmitted clock-centered with respect to DDR\_CLK and DDR\_CLKn. The memory, on the positive edge of DDR\_CLK and the negative edge of DDR\_CLKn, samples all address and control signals.

#### **7.3.3.3.4 VTP Controller Macro**

The VTP controller macro evaluates silicon performance at current voltage, temperature, and process (VTP) to enable IO drivers to set constant predetermined output driver impedance. The controller operates by comparing driver impedances to the external reference resistor and adjusting driver impedance to obtain an impedance match. VTP Controller supports the following features:

- The VTP controller generates information regarding the Voltage, temperature, and process(VTP) on a chip to be shared with the device's IO drivers
- Requires a Clock input from the core running at 20MHz or less
- 56 Clock cycles are needed to guarantee the VTP outputs are initially set after reset is removed
- Can be used in static or dynamic update mode of operation
- The VTP controller has internal noise filtering which allows it to control spurious update requests due to noise

Impedance of the drivers and terminations must be updated often even while in operation. In such scenarios where Voltage and Temperature are variables VTP macro can be configured in dynamic update mode. In contrast, static mode of operation does not allow dynamic calibration of IO impedance, and hence consumes lesser power compared to dynamic update mode.

It is possible that under noisy conditions, dynamic update controller can generate too frequent update requests. Noise can cause the controller to request a change in the impedance that can again be quickly reversed on subsequent clock cycles. To prevent the controller from making excessive numbers of impedance changes, a digital filter is included which can be configured to regulate the update rate. For example if the user configures the filter value as F2=0,F1=1 and F0=1, then an impedance update will be performed only if four successive update requests are generated from the VTP controller. It is recommended to use a filter value of 011'b.

Table 7-98 shows the configuration details of the digital filter.

**Table 7-98. Digital Filter Configuration**

| F2 | F1 | F0 | Description                             |
|----|----|----|---|
| 0  | 0  | 0  | Off                                     |
| 0  | 0  | 1  | Update on 2 consecutive update requests |
| 0  | 1  | 0  | Update on 3 consecutive update requests |
| 0  | 1  | 1  | Update on 4 consecutive update requests |
| 1  | 0  | 0  | Update on 5 consecutive update requests |
| 1  | 0  | 1  | Update on 6 consecutive update requests |
| 1  | 1  | 0  | Update on 7 consecutive update requests |

**Table 7-98. Digital Filter Configuration (continued)**

| F2 | F1 | F0 | Description                             |
|----|----|----|---|
| 1  | 1  | 1  | Update on 8 consecutive update requests |

### 7.3.3.3.5 DQS-Gate IOs

To effectively model the I/O delay on the DQS gating signal during a read request (the DQS receiver and the CLK driver I/Os), the signal is expected to be looped on a single I/O connecting fifo\_we\_in to fifo\_we\_out. The board and memory delay, being fairly constant across PTV variations, are calibrated within the IDID using a compensated delay line. The loop-back is done at the die level without bringing the signals out to the package level. Each Data Macro supports delay compensation independent of each other.

The data and command macros are responsible for System level flight time compensation. The following controls are supported by the DDR2/3/mDDR controller Sub System.

- Aligning DDR\_DQS w.r.t DDR\_CLK during Write Cycle: For DDR3 operation, initiate the write leveling state machine on each rank in turn to capture the proper delay settings to align DDR\_DQS with DDR\_CLK clock for each memory chip. If one wants to do this in a manual way one can write to the control register that controls the delay of DDR\_DQS vs DDR\_CLK Clock position. To produce a given amount of skew to center the DDR\_DQS vs. Clock at the SDRAM the following register can be programmed.

Data Macro 0/1 Write DQS Slave Ratio Register=256 x ([command delay] – [DDR\_DQS delay]) / DDR\_CLK Clock period.

- Aligning ADDR/CMD w.r.t DDR\_CLK
- Aligning DDR\_DQ[15:0] w.r.t DDR\_DQS during Write Operation
- Offset DDR\_D[15:0] w.r.t DDR\_DQS during Read Operation
- Align FIFO WE Window

### 7.3.3.4 Address Mapping

The DDR2/3/mDDR memory controller views external DDR2/3/mDDR SDRAM as one continuous block of memory. This statement is true regardless of the number of memory devices located on the chip select space. The DDR2/3/mDDR memory controller receives DDR2/3/mDDR memory access requests along with a 32-bit logical address from the rest of the system. In turn, DDR2/3/mDDR memory controller uses the logical address to generate a row/page, column, and bank address for the DDR2/3/mDDR SDRAM. The number of column, row and bank address bits used is determined by the IBANK, RSIZE and PAGESIZE fields (see [Table 7-99](#)). The DDR2/3/mDDR memory controller uses up to 16 bits for the row/page address.

**Table 7-99. IBANK, RSIZE and PAGESIZE Fields Information**

| Bit Field | Bit Value | Bit Description  |
|-----------|-----------|--|
| RSIZE     |           | Defines the number of address lines to be connected to DDR2/3/mDDR memory device |
|           | 0         | 9 row bits   |
|           | 1h        | 10 row bits  |
|           | 2h        | 11 row bits  |
|           | 3h        | 12 row bits  |
|           | 4h        | 13 row bits  |
|           | 5h        | 14 row bits  |
|           | 6h        | 15 row bits  |
|           | 7h        | 16 row bits  |

**Table 7-99. IBANK, RSIZE and PAGESIZE Fields Information (continued)**

| Bit Field | Bit Value | Bit Description  |
|-----------|-----------|--|
| PAGESIZE  |           | Defines the page size of each page of the external DDR2/3/mDDR memory device   |
|           | 0         | 256 words (requires 8 column address bits)                                     |
|           | 1h        | 512 words (requires 9 column address bits)                                     |
|           | 2h        | 1024 words (requires 10 column address bits)                                   |
|           | 3h        | 2048 words (requires 11 column address bits)                                   |
| IBANK     |           | Defines the number of internal banks on the external DDR2/3/mDDR memory device |
|           | 0         | 1 bank   |
|           | 1h        | 2 banks  |
|           | 2h        | 4 banks  |
|           | 3h        | 8 banks  |
| EBANK     |           | Defines the number of DDR2/3/mDDR memory controller chip selects               |
|           | 0         | CS0 only   |
|           | 1h        | Reserved   |

When addressing SDRAM, if the REG\_IBANK\_POS field in the SDRAM Config register is set to 0, and the REG\_EBANK\_POS field in the SDRAM Config 2 register is also set to 0, the DDR2/3/mDDR memory controller uses the three fields, IBANK, EBANK and PAGESIZE in the SDRAM Config register to determine the mapping from source address to SDRAM row, column, bank, and chip select. If the REG\_IBANK\_POS field in the SDRAM Config register is set to 1, 2, or 3, or the REG\_EBANK\_POS field in the SDRAM Config 2 register is set to 1, the DDR2/3/mDDR memory controller uses the 4 fields - IBANK, EBANK, PAGESIZE, and ROWSIZE in the SDRAM Config register to determine the mapping from source address to SDRAM row, column, bank, and chip select. In all cases the DDR2/3/mDDR memory controller considers its SDRAM address space to be a single logical block regardless of the number of physical devices or whether the devices are mapped across 1 or 2 DDR2/3/mDDR memory controller chip selects.

#### 7.3.3.4.1 Address Mapping when REG\_IBANK\_POS=0 and REG\_EBANK\_POS=0

For REG\_IBANK\_POS=0 and REG\_EBANK\_POS=0, the effect of address mapping scheme is that as the source address increments across DDR2/3/mDDR memory device page boundaries, the DDR2/3/mDDR controller moves onto the same page in the next bank in the current device DDR\_CS<sub>n</sub>[0]. This movement along the banks of the current proceeds to the same page in the next device(if EBANK=1, DDR\_CS<sub>n</sub>[1]) and proceeds through the same page in all its banks before moving over to the next page in the first device(DDR\_CS<sub>n</sub>[0]). The DDR2/3/mDDR controller exploits this traversal across internal banks and chip selects while remaining on the same page to maximize the number of open DDR2/3/mDDR memory device banks within the overall DDR2/3/mDDR memory device space.

Thus, the DDR2/3/mDDR controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, and can interleave among all of them.

**Table 7-100. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=0 and REG\_EBANK\_POS=0**

| Logical Address |                                     |                                     |  |
|-----------------|-------------------------------------|-------------------------------------|--|
| Row Address     | Chip Select                         | Bank Address                        | Column Address                         |
|                 | # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| 15 bits         | EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
|                 | EBANK=1 => 1 bit                    | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
|                 |                                     | IBANK=2 => 2 bits                   | PAGESIZE=2 => 10 bits                  |
|                 |                                     | IBANK=3 => 3 bits                   | PAGESIZE=3 => 11 bits                  |

#### 7.3.3.4.2 Address Mapping when REG\_IBANK\_POS = 1 and REG\_EBANK\_POS = 0

For REG\_IBANK\_POS = 1 and REG\_EBANK\_POS = 0, the interleaving of banks within a device (per chip select) is limited to 4 banks. However, it can still interleave banks between the two chip selects. Thus, the DDR2/3/mDDR controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among eight of them.

**Table 7-101. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=1 and REG\_EBANK\_POS=0**

| Logical Address                     |                                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Bank Address[2]                     | Row Address                         | Chip Select                         | Bank Address[1:0]                   | Column Address                         |
| # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| IBANK=1 => 0 bits                   | RSIZE=1 => 10 bits                  | EBANK=1 => 1 bit                    | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
| IBANK=2 => 0 bits                   | RSIZE=2 => 11 bits                  |                                     | IBANK=2 => 2 bits                   | PAGESIZE=2 => 10 bits                  |
| IBANK=3 => 1 bit                    | RSIZE=3 => 12 bits                  |                                     | IBANK=3 => 3 bits                   | PAGESIZE=3 => 11 bits                  |
|                                     | RSIZE=4 => 13 bits                  |                                     |                                     |  |
|                                     | RSIZE=5 => 14 bits                  |                                     |                                     |  |
|                                     | RSIZE=6 => 15 bits                  |                                     |                                     |  |
|                                     | RSIZE=7 => 16 bits                  |                                     |                                     |  |

#### 7.3.3.4.3 Address Mapping when REG\_IBANK\_POS=2 and REG\_EBANK\_POS = 0

For REG\_IBANK\_POS=2 and REG\_EBANK\_POS = 0, the interleaving of banks within a device (per chip select) is limited to 2 banks. However, it can still interleave banks between the two chip selects. Thus, the DDR2/3/mDDR controller can keep a maximum of 16 banks (eight internal banks across 2 chip selects) open at a time, but can only interleave among four of them.

**Table 7-102. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=2 and REG\_EBANK\_POS=0**

| Logical Address                     |                                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Bank Address[2:1]                   | Row Address                         | Chip Select                         | Bank Address[0]                     | Column Address                         |
| # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| IBANK=1 => 0 bits                   | RSIZE=1 => 10 bits                  | EBANK=1 => 1 bit                    | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
| IBANK=2 => 1 bit                    | RSIZE=2 => 11 bits                  |                                     | IBANK=2 => 1 bit                    | PAGESIZE=2 => 10 bits                  |
| IBANK=3 => 2 bits                   | RSIZE=3 => 12 bits                  |                                     | IBANK=3 => 1 bit                    | PAGESIZE=3 => 11 bits                  |
|                                     | RSIZE=4 => 13 bits                  |                                     |                                     |  |
|                                     | RSIZE=5 => 14 bits                  |                                     |                                     |  |
|                                     | RSIZE=6 => 15 bits                  |                                     |                                     |  |
|                                     | RSIZE=7 => 16 bits                  |                                     |                                     |  |

#### 7.3.3.4.4 Address Mapping when REG\_IBANK\_POS= 3 and REG\_EBANK\_POS = 0

For REG\_IBANK\_POS= 3 and REG\_EBANK\_POS = 0, the DDR2/3/mDDR controller cannot interleave banks within a device (per chip select). However, it can still interleave banks between the two chip selects. Thus, the DDR2/3/mDDR controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among two of them.

**Table 7-103. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=3 and REG\_EBANK\_POS=0**

| Logical Address                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Bank Address                        | Row Address                         | Chip Select                         | Column Address                         |
| # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by EBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | EBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| IBANK=1 => 1 bit                    | RSIZE=1 => 10 bits                  | EBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
| IBANK=2 => 2 bits                   | RSIZE=2 => 11 bits                  |                                     | PAGESIZE=2 => 10 bits                  |
| IBANK=3 => 3 bits                   | RSIZE=3 => 12 bits                  |                                     | PAGESIZE=3 => 11 bits                  |
|                                     | RSIZE=4 => 13 bits                  |                                     |  |
|                                     | RSIZE=5 => 14 bits                  |                                     |  |
|                                     | RSIZE=6 => 15 bits                  |                                     |  |
|                                     | RSIZE=7 => 16 bits                  |                                     |  |

#### 7.3.3.4.5 Address Mapping when REG\_IBANK\_POS = 0 and REG\_EBANK\_POS = 1

For REG\_IBANK\_POS = 0 and REG\_EBANK\_POS = 1, the DDR2/3/mDDR memory controller interleaves among all the banks within a device (per chip select). However, the DDR2/3/mDDR memory controller cannot interleave banks between the two chip selects. Thus, the DDR2/3/mDDR memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among 8 of them.

**Table 7-104. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=0 and REG\_EBANK\_POS=1**

| Logical Address                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Chip Select                         | Row Address                         | Bank Address                        | Column Address                         |
| # of bits defined by EBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| EBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| EBANK=1 => 1 bit                    | RSIZE=1 => 10 bits                  | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
|                                     | RSIZE=2 => 11 bits                  | IBANK=2 => 2 bits                   | PAGESIZE=2 => 10 bits                  |
|                                     | RSIZE=3 => 12 bits                  | IBANK=3 => 3 bits                   | PAGESIZE=3 => 11 bits                  |
|                                     | RSIZE=4 => 13 bits                  |                                     |  |
|                                     | RSIZE=5 => 14 bits                  |                                     |  |
|                                     | RSIZE=6 => 15 bits                  |                                     |  |
|                                     | RSIZE=7 => 16 bits                  |                                     |  |

#### 7.3.3.4.6 Address Mapping when REG\_IBANK\_POS = 1 and REG\_EBANK\_POS = 1

For REG\_IBANK\_POS = 1 and REG\_EBANK\_POS = 1, the interleaving of banks within a device (per chip select) is limited to 4 banks. Also, the DDR2/3/mDDR memory controller cannot interleave banks between the two chip selects. Thus, the DDR2/3/mDDR memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among four of them.

**Table 7-105. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=1 and REG\_EBANK\_POS = 1**

| Logical Address                     |                                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Chip Select                         | Bank Address[2]                     | Row Address                         | Bank Address[1:0]                   | Column Address                         |
| # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| EBANK=1 => 1 bit                    | IBANK=1 => 0 bits                   | RSIZE=1 => 10 bits                  | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |

**Table 7-105. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=1 and REG\_EBANK\_POS = 1 (continued)**

| Logical Address |                   |                    |                   |                       |
|-----------------|-------------------|--------------------|-------------------|-----------------------|
| Chip Select     | Bank Address[2]   | Row Address        | Bank Address[1:0] | Column Address        |
|                 | IBANK=2 => 0 bits | RSIZE=2 => 11 bits | IBANK=2 => 2 bits | PAGESIZE=2 => 10 bits |
|                 | IBANK=3 => 1 bit  | RSIZE=3 => 12 bits | IBANK=3 => 2 bits | PAGESIZE=3 => 11 bits |
|                 |                   | RSIZE=4 => 13 bits |                   |                       |
|                 |                   | RSIZE=5 => 14 bits |                   |                       |
|                 |                   | RSIZE=6 => 15 bits |                   |                       |
|                 |                   | RSIZE=7 => 16 bits |                   |                       |

#### 7.3.3.4.7 Address Mapping when REG\_IBANK\_POS = 2 and REG\_EBANK\_POS = 1

For REG\_IBANK\_POS = 2 and REG\_EBANK\_POS = 1, the interleaving of banks within a device (per chip select) is limited to 2 banks. Also, the DDR2/3/mDDR memory controller cannot interleave banks between the two chip selects. Thus, the DDR2/3/mDDR memory controller can keep a maximum of 16 banks (8 internal banks across 2 chip selects) open at a time, but can only interleave among two of them.

**Table 7-106. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=2 and REG\_EBANK\_POS = 1**

| Logical Address                     |                                     |                                     |                                     |  |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--|
| Chip Select                         | Bank Address[2:1]                   | Row Address                         | Bank Address[0]                     | Column Address                         |
| # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by PAGESIZE of SDRCR |
| EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | IBANK=0 => 0 bits                   | PAGESIZE=0 => 8 bits                   |
| EBANK=1 => 1 bit                    | IBANK=1 => 0 bits                   | RSIZE=1 => 10 bits                  | IBANK=1 => 1 bit                    | PAGESIZE=1 => 9 bits                   |
|                                     | IBANK=2 => 1 bit                    | RSIZE=2 => 11 bits                  | IBANK=2 => 1 bit                    | PAGESIZE=2 => 10 bits                  |
|                                     | IBANK=3 => 2 bits                   | RSIZE=3 => 12 bits                  | IBANK=3 => 1 bit                    | PAGESIZE=3 => 11 bits                  |
|                                     |                                     | RSIZE=4 => 13 bits                  |                                     |  |
|                                     |                                     | RSIZE=5 => 14 bits                  |                                     |  |
|                                     |                                     | RSIZE=6 => 15 bits                  |                                     |  |
|                                     |                                     | RSIZE=7 => 16 bits                  |                                     |  |

#### 7.3.3.4.8 Address Mapping when REG\_IBANK\_POS = 3 and REG\_EBANK\_POS = 1

For REG\_IBANK\_POS = 3 and REG\_EBANK\_POS = 1, the DDR2/3/mDDR memory controller cannot interleave banks within a device (per chip select) or between the two chip selects. Thus, the DDR2/3/mDDR memory controller can keep a maximum of 16 banks (8 internal banks across two chip selects) open at a time, but cannot interleave among of them.

**Table 7-107. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=3 and REG\_EBANK\_POS=1**

| Logical Address                     |                                     |                                     |  |  |
|-------------------------------------|-------------------------------------|-------------------------------------|--|--|
| Chip Select                         | Bank Address                        | Row Address                         | Column Address                         |  |
| # of bits defined by EBANK of SDRCR | # of bits defined by IBANK of SDRCR | # of bits defined by RSIZE of SDRCR | # of bits defined by PAGESIZE of SDRCR |  |
| EBANK=0 => 0 bits                   | IBANK=0 => 0 bits                   | RSIZE=0 => 9 bits                   | PAGESIZE=0 => 8 bits                   |  |
| EBANK=1 => 1 bit                    | IBANK=1 => 1 bit                    | RSIZE=1 => 10 bits                  | PAGESIZE=1 => 9 bits                   |  |
|                                     | IBANK=2 => 2 bits                   | RSIZE=2 => 11 bits                  | PAGESIZE=2 => 10 bits                  |  |
|                                     | IBANK=3 => 3 bits                   | RSIZE=3 => 12 bits                  | PAGESIZE=3 => 11 bits                  |  |
|                                     |                                     | RSIZE=4 => 13 bits                  |  |  |
|                                     |                                     | RSIZE=5 => 14 bits                  |  |  |

**Table 7-107. OCP Address to DDR2/3/mDDR Address Mapping for REG\_IBANK\_POS=3 and REG\_EBANK\_POS=1 (continued)**

| Logical Address |              |                    |                |
|-----------------|--------------|--------------------|----------------|
| Chip Select     | Bank Address | Row Address        | Column Address |
|                 |              | RSIZE=6 => 15 bits |                |
|                 |              | RSIZE=7 => 16 bits |                |

Since the DDR2/3/mDDR memory controller interleaves among less number of banks when IBANK\_POS!=0 or EBANK\_POS=1, these cases are lower in performance than the IBANK\_POS=0 case. Thus these cases are only recommended to be used along with partial array self-refresh where performance can be traded off for power savings.

### 7.3.3.5 Performance Management

#### 7.3.3.5.1 Command Ordering and Scheduling

The DDR2/3/mDDR memory controller performs command re-ordering and scheduling in an attempt to achieve efficient transfers with maximum throughput. The goal is to maximize the utilization of the data, address, and command buses while hiding the overhead of opening and closing DDR2/3/mDDR SDRAM rows. Command re-ordering takes place within the command FIFO.

The DDR2/3/mDDR memory controller examines all the commands stored in the command FIFO to schedule commands to the external memory. All commands from same master will complete in order, regardless of the master priority. The memory controller does not guarantee ordering between commands from different masters. However, the memory controller will maintain data coherency. Therefore, the memory controller will block a command, regardless of master priority, if that command is to the same block address (2048 bytes) as an older command. Thus, the memory controller might have one or more pending read or write for each master. Among all pending reads, the memory controller selects all reads that have their corresponding SDRAM banks already open. Similarly, among all pending writes, the memory controller selects all writes that have their corresponding SDRAM banks already open.

As a result of the above reordering, at any point of time the memory controller might have several pending reads and writes that have their corresponding banks open. The memory controller then selects the highest priority read from pending reads, and the highest priority write from pending writes. If two or more commands have the highest priority, the memory controller selects the oldest command. As a result, the memory controller might now have a final read and a final write command. The memory controller will pick either the read or the write command depending on the value programmed in the Read Write Execution Threshold register. The memory controller will keep executing reads until the read threshold is met and then switch to executing writes. The memory controller will then keep executing writes until the write threshold is met and then switch back to executing reads. The memory controller will satisfy meeting the threshold values only if that type of command is available for execution, otherwise it will switch to the other type. Similarly, the memory controller will satisfy meeting the threshold value only if the FIFOs for that type have space (Read Data FIFO for reads and Write Status FIFO for writes), otherwise it will switch to the other type.

The memory controller completes executing an OCP command before it switches to another command.

All the accesses to an SDRAM are pipe-lined to maximize the external bus utilization. In other words accesses to an SDRAM are issued back to back such that there are minimum idle cycles between any two accesses. This includes the scheduling listed above to minimize the overhead of opening and closing of SDRAM banks. All of these is done while fulfilling the access timing requirements of an SDRAM.

Besides commands received from on-chip resources, the DDR2/3/mDDR memory controller also issues refresh commands. The DDR2/3/mDDR memory controller attempts to delay refresh commands as long as possible to maximize performance while meeting the SDRAM refresh requirements. As the DDR2/3/mDDR memory controller issues read, write, and refresh commands to DDR2/3/mDDR SDRAM device, it follows the following priority scheme:

1. (Highest priority) SDRAM refresh request due to Refresh Must level of refresh urgency reached (Refer Section Refresh Scheduling)

2. Request for a read or a write.
3. SDRAM Activate commands.
4. SDRAM Deactivate commands.
5. SDRAM Deep Power-Down request.
6. SDRAM clock stop or Power-Down request.
7. SDRAM refresh request due to Refresh May or Release level of refresh urgency reached (Refer Section Refresh Scheduling)
8. (Lowest priority) SDRAM self-refresh request.

#### **7.3.3.5.2 Command Starvation**

The reordering and scheduling rules listed above may lead to command starvation, which is the prevention of certain commands from being processed by the DDR2/3/mDDR memory controller. Command starvation results from the following conditions:

- A continuous stream of high-priority read commands can block a low-priority write command
- A continuous stream of DDR2/3/mDDR SDRAM commands to a row in an open bank can block commands to the closed row in the same bank.

To avoid these conditions, the DDR2/3/mDDR memory controller can momentarily raise the priority of the oldest command in the command FIFO after a set number of transfers have been made. The REG\_COS\_COUNT\_1,REG\_COS\_COUNT\_2 field in the Interface Configuration Register (INT\_CONFIG) sets the number of the transfers that must be made before the DDR2/3/mDDR memory controller will raise the priority of the oldest command. See Class of Service (COS) section for more details.

---

**NOTE:** Leaving the REG\_COS bits at their default value (FFh) in the Interface Configuration register (INT\_CONFIG) disables this feature of the DDR2/3/mDDR memory controller. This means commands can stay in the command FIFO indefinitely. Therefore, these bits should be set to FEh immediately following reset to enable this feature with the highest level of allowable memory transfers. It is suggested that system-level prioritization be set to avoid placing high-bandwidth masters on the highest priority levels. These bits can be left as FEh unless advanced bandwidth/prioritization control is required.

---

#### **7.3.3.5.3 Possible Race Condition**

A race condition may exist when certain masters write data to the DDR2/3/mDDR memory controller. For example, if master A passes a software message via a buffer in DDR2/3/mDDR memory and does not wait for indication that the write completes, when master B attempts to read the software message it may read stale data and therefore receive an incorrect message. In order to confirm that a write from master A has landed before a read from master B is performed, master A must wait for the write completion status from the DDR2/3/mDDR memory controller before indicating to master B that the data is ready to be read. If master A does not wait for indication that a write is complete, it must perform the following workaround:

1. Perform the required write.
2. Perform a dummy write to the DDR2 memory controller module ID and revision register.
3. Perform a dummy read to the DDR2 memory controller module ID and revision register.
4. Indicate to master B that the data is ready to be read after completion of the read in step 3.

The completion of the read in step 3 ensures that the previous write was done.

For a list of the master peripherals that need this workaround, see the device-specific data sheet.

#### **7.3.3.5.4 Class of Service (COS)**

The commands in the Command FIFO can be mapped to 2 classes of service namely 1 and 2. The mapping of commands to a particular class of service can be done based on the priority or the connection ID.

The mapping based on priority can be done by setting the appropriate values in the Priority to Class of Service Mapping register (PRI\_COS\_MAP).

The mapping based on connection ID can be done by setting the appropriate values of connection ID and the masks in the Connection ID to Class of Service Mapping registers(CONNID\_COS\_1\_MAP and CONNID\_COS\_2\_MAP).

There are 3 connection ID and mask values that can be set for each class of service. In conjunction with the masks, each class of service can have a maximum of 144 connection IDs mapped to it. For example, a connection ID value of 0xFF along with a mask value of 0x3 will map all connection IDs from 0xF8 to 0xFF to that particular class of service.

Each class of service has an associated latency counter (REG\_COS\_COUNT). The value of this counter can be set in the Interface Configuration Register. When the latency counter for a command expires, i.e., reaches the value programmed for the class of service that the command belongs to, that command will be the one that is executed next. If there are more than one commands that have expired latency counters, the command with the highest priority will be executed first. One exception to this rule is, if the oldest command in the queue has an expired reg\_pr\_old\_count, that command will be executed first irrespective of priority or class of service. This is done to prevent a continuous block effect.

The connection ID mapping allows the same connection ID to be put in both class of service 1 and 2. Also, a transaction might belong to one class of service if viewed by connection ID and might belong to another class of service if viewed by priority. In these cases, the command will belong to both class of service. The DDR2/3/mDDR memory controller will try executing the command as soon as possible, when the smaller of the two counters ( REG\_COS\_COUNT\_1 OR REG\_COS\_COUNT\_2) expire.

#### **7.3.3.5.5 Refresh Scheduling**

The DDR2/3/mDDR memory controller issues autorefresh (REFR) commands to DDR2/3/mDDR SDRAM devices at a rate defined in the refresh rate (REFRESH\_RATE) bit field in the SDRAM refresh control register (SDRFC). A refresh interval counter is loaded with the value of the REFRESH\_RATE bit field and decrements by 1 each cycle until it reaches zero. Once the interval counter reaches zero, it reloads with the value of the REFRESH\_RATE bit. Each time the interval counter expires, a refresh backlog counter increments by 1. Conversely, each time the DDR2/3/mDDR memory controller performs a REFR command, the backlog counter decrements by 1. This means the refresh backlog counter records the number of REFR commands the DDR2/3/mDDR memory controller currently has outstanding.

The DDR2/3/mDDR memory controller issues REFR commands based on the level of urgency. The level of urgency is defined below. Whenever the refresh level of urgency is reached, the DDR2/3/mDDR memory controller issues a REFR command before servicing any new memory access requests. Following a REFR command, the DDR2/3/mDDR memory controller waits T\_RFC cycles, defined in the SDRAM timing 1 register (SDRTIM1), before rechecking the refresh urgency level.

The refresh counters do not operate when the SDRAM memory is in self-refresh mode.

**Table 7-108. Refresh Modes**

| Urgency Level   | Description   |
|-----------------|---|
| Refresh May     | Backlog count is greater than 0. Indicates there is a backlog of REFR commands, when the DDR2/3/mDDR memory controller is not busy it will issue the REFR command.  |
| Refresh Release | Backlog count is greater than 4. Indicates that the refresh backlog of REFR commands is getting high and when DDR2/3/mDDR memory controller is not busy it should issue the REFR command.   |
| Refresh Must    | Backlog count is greater than 7. Indicates that the refresh backlog of REFR commands is getting excessive and DDR2/3/mDDR memory controller should perform an auto refresh cycle before servicing any new memory access requests. |

The DDR2/3/mDDR memory controller starts servicing new memory accesses after Refresh Release level is cleared. If any of the commands in the Command FIFO have class of service latency counters that are expired, the DDR2/3/mDDR memory controller will not wait for Refresh Release level to be cleared but will only perform one refresh command and exit the refresh state.

### 7.3.3.6 DDR3 Read-Write Leveling

The DDR2/3/mDDR memory controller supports read-write leveling in conjunction with the DDR PHY. The DDR2/3/mDDR memory controller supports two types of write/read leveling:

1. Full leveling
2. Incremental leveling

---

**NOTE:** Please refer the device specific data sheet to know the type of leveling supported.

---

Each leveling type has three parts:

1. Write leveling
2. Read DQS gate training
3. Read data eye training

Read and write leveling is only supported to DDR3 memory.

The DDR2/3/mDDR memory controller does not perform full leveling after initialization upon reset deassertion. Full leveling must be triggered by software after the DDR2/3/mDDR memory controller registers are properly configured. The DDR2/3/mDDR memory controller supports triggering of full leveling through software through the use of the REG\_RDWRLVLFULL\_START field in the Read-Write Leveling Control register(RWLCR). Since full leveling takes considerable amount of time and refreshes cannot be issued to DDR3 when DDR3 is put in leveling mode, refresh interval will be violated and data inside DDR3 can be lost. Although, this is not an issue at power-up, this might be an issue if full leveling is triggered when DDR3 is functional.

The memory controller supports incremental leveling to better track voltage and temperature changes during normal operation. The incremental leveling can be enabled by writing a non-zero value to the REG\_WRLVLINC\_INT, REG\_RDLVLGATEINC\_INT, and REG\_RDLVLINC\_INT fields in the Read-Write Leveling Control register(RWLCR). The memory controller periodically triggers incremental write leveling every time REG\_WRLVLINC\_INT expires. In other words, the REG\_WRLVLINC\_INT defines the interval between successive incremental write leveling.

Similarly, the memory controller periodically triggers incremental read DQS gate training every time REG\_RDLVLGATEINC\_INT expires, and triggers incremental read data eye training every time REG\_RDLVLINC\_INT expires.

To minimize impact on bandwidth, the software can program these intervals such that these three intervals do not expire at same time. The value of interval programmed is dependent on the slope of voltage and temperature changes.

The memory controller supports increasing the rate of incremental leveling automatically for a defined period of time. This can be achieved by programming the Read-Write Leveling Ramp Window register(RDWR\_LVL\_RMP\_WIN) and the Read-Write Leveling Ramp Control register(RDWR\_LVL\_RMP\_CTRL). Whenever a pulse is received, the memory controller would use the intervals programmed in the Read-Write Leveling Ramp Control register until the REG\_RDWRLVLINC\_RMP\_WIN in the Read-Write Leveling Ramp Window register expires. After the expiration of REG\_RDWRLVLINC\_RMP\_WIN the memory controller switches back to use the intervals programmed in the Read-Write Leveling Control register.

To guarantee none of the incremental leveling events are missed, the REG\_RDWRLVLINC\_RMP\_WIN must be programmed greater than the intervals in the Read-Write Leveling Ramp Control register.

If the memory controller is in Self-Refresh or Power-Down modes when any of the incremental leveling intervals expire, the memory controller will exit Self-Refresh or Power-Down mode, perform the required leveling, and then re-enter the Self-Refresh or Power-Down mode. The memory controller also triggers incremental leveling on Self-Refresh exit.

### 7.3.3.7 PRCM Sequence for DDR2/3/mDDR Memory controller

The memory controller clock, reset and power are handled by the device PRCM module. Refer to the Power Reset Clock Management (PRCM) chapter for the PRCM register details.

### 7.3.3.8 Interrupt Support

The DDR2/3/mDDR controller is compliant with Open Core Protocol Specification (OCP-IP 2.2). The controller supports only Idle, Write, Read, and WriteNonPost command types. Also, the controller supports only incrementing, wrapping, and 2-dimensional block addressing modes. The controller supports generation of an error interrupt if an unsupported command or a command with unsupported addressing mode is received.

### 7.3.3.9 EDMA Event Support

The DDR2/3/mDDR memory controller is a DMA slave peripheral and therefore does not generate EDMA events. Data read and write requests may be made directly by masters including the EDMA controller.

### 7.3.3.10 Emulation Considerations

The DDR2/3/mDDR memory controller will remain fully functional during emulation halts to allow emulation access to external memory.

### 7.3.3.11 Power Management

This section defines the power management capabilities and requirements.

#### 7.3.3.11.1 Clock Stop Mode

The memory controller supports Clock Stop mode for LPDDR1/mDDR. The memory controller automatically stops the clocks to the memory, after the memory controller is idle for REG\_CS\_TIM number of DDR clock cycles and the REG\_LP\_MODE field is set to 1. The REG\_LP\_MODE and REG\_CS\_TIM fields can be programmed in the power management control register (PMCR).

When the clock to the memory is stopped, the memory controller services register accesses as normal. If an SDRAM access is requested, or the Refresh Must level is reached while in the Clock Stop mode, the memory controller will start the clocks. The memory controller now can issue any commands. If the power saving mode is changed by changing REG\_LP\_MODE from 1 to some other value, the memory controller will exit Clock Stop mode and enter the new power saving mode.

#### 7.3.3.11.2 Self-Refresh Mode

The DDR2/3/mDDR memory controller supports self-refresh mode for low power. The memory controller automatically puts the SDRAM into self-refresh after the memory controller is idle for REG\_SR\_TIM number of DDR clock cycles and the REG\_LP\_MODE field is set to 2. The REG\_LP\_MODE and REG\_SR\_TIM fields can be programmed in the Power Management Control register(PMCR). The memory controller will complete all pending refreshes before it puts the SDRAM into self-refresh. Therefore, after the expiration of REG\_SR\_TIM, the memory controller will start issuing refreshes to complete the refresh backlog, and then issue a SELF-REFRESH command to the SDRAM.

In self-refresh mode, the memory controller automatically stops the clocks DDR\_CLK to the SDRAM. The memory controller maintains DDR\_CKE low to maintain the self-refresh state. When the SDRAM is in self-refresh, the memory controller services register accesses as normal. If the REG\_LP\_MODE field is set not equal to 2, or an SDRAM access is requested while it is in self-refresh, and T\_CKE + 1 cycles have elapsed since the SELF-REFRESH command was issued, the memory controller will bring the SDRAM out of self-refresh. The value of T\_CKE is taken from SDRAM Timing 2 register. For DDR3, memory controller will also exit self-refresh to perform incremental leveling.

**Exit sequence of self-refresh mode for LPDDR1 device:** The memory controller:

- Enables clocks.
- Drives DDR\_CKE high.

- Waits for T\_XSNR + 1 cycles. The value of T\_XSNR is taken from SDRAM Timing 2 register.
- Starts an auto-refresh cycle in the next cycle.
- Enters its idle state and can issue any commands.

**Exit sequence of self-refresh mode for DDR2 device:** The memory controller:

- Enables clocks.
- Drives DDR\_CKE high.
- Waits for T\_XSNR + 1 cycles. The value of T\_XSNR is taken from SDRAM Timing 2 register.
- If the REG\_DDR\_DISABLE\_DLL bit in the SDRAM Config register is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 with the pad\_a\_o bits set as follows:

| Bits         | Value           | Description  |
|--------------|-----------------|--|
| DDR_A[15:13] | 0x0             | !reg_ddr2_ddqs   |
| DDR_A[12]    | 0x0             | Output buffer enabled                                      |
| DDR_A[11]    | 0x0             | RDQS disable   |
| DDR_A[10]    | !reg_ddr2_ddqs  | Differential DQS enable value from SDRAM Config register   |
| DDR_A[9:7]   | 0x0             | Exit OCD calibration mode                                  |
| DDR_A[6]     | reg_ddr_term[1] | DDR2 termination resistor value from SDRAM Config register |
| DDR_A[5:3]   | 0x0             | Additive latency = 0                                       |
| DDR_A[2]     | reg_ddr_term[0] | DDR2 termination resistor value from SDRAM Config register |
| DDR_A[1]     | reg_sdram_drive | SDRAM drive strength from SDRAM Config register            |
| DDR_A[0]     | 0x1             | Disable DLL  |

- Starts an auto-refresh cycle in the next cycle.
- Enters its idle state and can issue any other commands except a write or a read. A write or a read will only be issued after T\_XSRD + 1 clock cycles have elapsed since DDR\_CKE is driven high. The value of T\_XSRD is taken from SDRAM Timing 2 register.

**Exit sequence of self-refresh mode for DDR3 device:** The memory controller:

- Enables clocks.
- Drives DDR\_CKE high.
- Waits for T\_XSNR + 1 cycles. The value of T\_XSNR is taken from SDRAM Timing 2 register.
- If the REG\_DDR\_DISABLE\_DLL bit in the SDRAM Config register is 1, issues a LOAD MODE REGISTER command to the extended mode register 1 with the pad\_a\_o bits set as follows:

| Bits         | Value              | Description  |
|--------------|--------------------|--|
| DDR_A[15:13] | 0x0                | Reserved   |
| DDR_A[12]    | 0x0                | Output buffer enabled                                      |
| DDR_A[11]    | 0x0                | TDQS disable   |
| DDR_A[10]    | 0x0                | Reserved   |
| DDR_A[9]     | reg_ddr_term[2]    | DDR3 termination resistor value from SDRAM Config register |
| DDR_A[8]     | 0x0                | Reserved   |
| DDR_A[7]     | 0x0                | Write leveling disabled                                    |
| DDR_A[6]     | reg_ddr_term[1]    | DDR3 termination resistor value from SDRAM Config register |
| DDR_A[5]     | reg_sdram_drive[1] | SDRAM drive strength from SDRAM Config register            |
| DDR_A[4:3]   | 0x0                | Additive latency = 0                                       |
| DDR_A[2]     | reg_ddr_term[0]    | DDR3 termination resistor value from SDRAM Config register |
| DDR_A[1]     | reg_sdram_drive[0] | SDRAM drive strength from SDRAM Config register            |
| DDR_A[0]     | 0x1                | Disable DLL  |

- Starts an auto-refresh cycle in the next cycle.
- Performs one write incremental leveling.
- Performs read DQS incremental training.
- Performs read data-eye incremental training.
- Enters its idle state and can issue any other commands except a write or a read. A write or a read will only be issued after  $T_{XSRD} + 1$  clock cycles have elapsed since DDR\_CKE is driven high. The value of  $T_{XSRD}$  is taken from SDRAM Timing 2 register.

#### **7.3.3.11.3 Power Down Mode**

The memory controller also supports power-down mode for low power. The memory controller automatically puts the SDRAM into power-down after the memory controller is idle for REG\_PD\_TIM number of DDR clock cycles and the REG\_LP\_MODE field is set to 4. The REG\_LP\_MODE and REG\_PD\_TIM fields can be programmed in the Power Management Control register (PMCR). If the Refresh Must level is not reached before the entry into power-down, the memory controller will not precharge all banks before issuing the POWER-DOWN command. This will result in SDRAM entering active power-down mode.

If the Refresh Must level is reached before the entry into power-down, the memory controller will precharge all banks and issue refreshes until the Refresh Release Level is reached before issuing the POWER-DOWN command. This will result in SDRAM entering precharge power-down mode.

In power-down mode, the memory controller does not stop the clocks DDR\_CLK to the SDRAM. The memory controller maintains DDR\_CKE low to maintain the power-down state.

When the SDRAM is in power-down, the memory controller services register accesses as normal. If the REG\_LP\_MODE field is set not equal to 4, or an SDRAM access is requested, or the Refresh Must level is reached while the SDRAM is in power-down, the memory controller will bring the SDRAM out of power-down. For DDR3, memory controller will also exit power-down to perform incremental leveling.

Exit sequence of power-down mode for DDR2, DDR3 and LPDDR1: The memory controller

- Drives DDR\_CKE high after  $T_{CKE} + 1$  cycles have elapsed since the POWER-DOWN command was issued. The value of  $T_{CKE}$  is taken from SDRAM Timing 2 register.
- Waits for  $T_{XP} + 1$  cycles. The value of  $T_{XP}$  is taken from SDRAM Timing 2 register.
- Enters its idle state and can issue any commands.

#### **7.3.3.11.4 Deep Power-Down Mode**

For ultimate power savings, the memory controller supports deep power-down mode for LPDDR1.

The SDRAM can be forced into deep power-down through software by setting the reg\_dpd\_en field in the Power Management Control register to 1. In this case, the memory controller will continue normal operation until all SDRAM memory access requests have been serviced. At this point the memory controller will issue a DEEP POWER-DOWN command. The memory controller then maintains pad\_cke\_o low to maintain the Deep Power-Down state. In deep power-down mode, the memory controller automatically stops the clocks to the SDRAM.

Setting the REG\_DPD\_EN field to 1 overrides the setting of REG\_LP\_MODE field. Therefore, if the SDRAM is in Clock Stop, Self Refresh, or Power-Down mode, and REG\_DPD\_EN field is set to 1, the memory controller will exit those modes and go into deep power-down mode.

When the SDRAM is in deep power-down, the memory controller services register accesses as normal.

If the REG\_DPD\_EN field is set to 0, or an SDRAM access is requested, the memory controller will bring the SDRAM out of deep power-down.

Exit sequence for LPDDR1: The memory controller:

- Performs SDRAM initialization as specified in the LPDDR1(mDDR) SDRAM Memory Initialization section.
- Enters its idle state and can issue any commands.

Since the memory controller performs initialization upon deep power-down exit, the REG\_REFRESH\_RATE field in the SDRAM Refresh Control register must be set appropriately to meet the 200 $\mu$ s wait requirement for LPDDR1.

#### 7.3.3.11.5 Save and Restore Mode

The DDR2/3/mDDR memory controller supports save and restore mechanism to completely switch off power to the DDR2/3/mDDR memory controller. The following sequence of operations is followed to put DDR2/3/mDDR memory controller in off mode:

An external master reads the following memory mapped registers and saves their value external to the DDR2/3/mDDR memory controller.

1. SDRAM Config register (SDRCR)
  2. SDRAM Config 2 register
  3. SDRAM Refresh Control register (SDRRCR)
  4. SDRAM Refresh Control Shadow register (SDRRCR)
  5. SDRAM Timing 1 register (SDRTIM1)
  6. SDRAM Timing 1 Shadow register (SDRTIM1SR)
  7. SDRAM Timing 2 register (SDRTIM2)
  8. SDRAM Timing 2 Shadow register (SDRTIM2SR)
  9. SDRAM Timing 3 register (SDRTIM3)
  10. SDRAM Timing 3 Shadow register (SDRTIM3SR)
  11. Power Management Control register (PMCR)
  12. Power Management Control Shadow register (PMCSR)
  13. Interface Configuration register (INT\_CONFIG)
  14. System OCP Interrupt Enable Set Register (SOIESR)
  15. DDR PHY Control 1 register (DDRPHYCR)
  16. DDR PHY Control 1 Shadow register (DDRPHYCSR)
- Memory controller completes all pending transactions and drains all its FIFOs.
  - Memory controller puts the SDRAM in Self Refresh.
  - Memory controller copies all shadow memory mapped registers to its main registers. It is assumed that the shadow register always has the same value as its corresponding main register.
  - Memory controller waits for all interrupts to be serviced.
  - Memory controller acknowledges assertion of internal power down request.
  - The internal module reset signal is asserted.
  - The clocks and power to the memory controller can now be switched off.

To restore power to the memory controller, the following sequence of operations is followed:

- The power and clocks to the Memory controller are switched on.
- The internal module reset signal is deasserted, indicating to the Memory controller that it is waking up from off mode.
- The memory controller does not perform SDRAM initialization and forces its state machine to be in self-refresh.
- The external master restores all of the above memory mapped registers.
- The external master restores all of the above memory mapped registers.
- The system can now perform access to the external memory.

#### 7.3.3.11.6 EMIF PHY Clock Gating

The clock to the DDR2/3/mDDR PHY can be gated off to achieve power saving. For more information, see the EMIF0/1 Clock Gate Control register (EMIF\_CLK\_GATE).

### 7.3.4 Use Cases

For details on connecting this device to mDDR/DDR2/DDR3 devices, see the device-specific data sheet, which will include specific instructions and routing guidelines for interfacing to mDDR (LPDDR), DDR2, and DDR3 devices.

### 7.3.5 EMIF4D Registers

[Table 7-109](#) lists the memory-mapped registers for the EMIF4D. All register offset addresses not listed in [Table 7-109](#) should be considered as reserved locations and the register contents should not be modified.

**Table 7-109. EMIF4D REGISTERS**

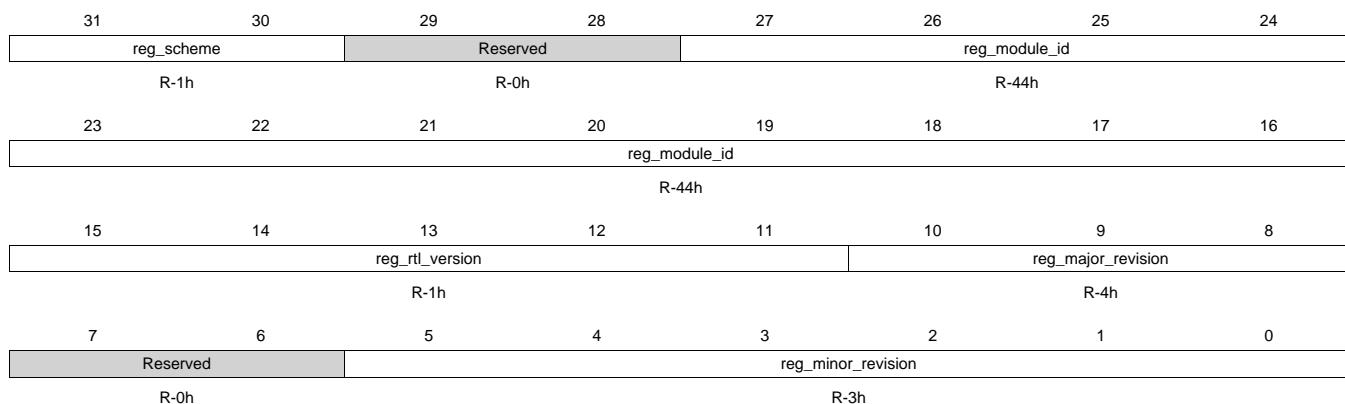
| Offset | Acronym             | Register Name  | Section                          |
|--------|---------------------|--|----------------------------------|
| 0h     | EMIF_MOD_ID_REV     |  | <a href="#">Section 7.3.5.1</a>  |
| 4h     | STATUS              |  | <a href="#">Section 7.3.5.2</a>  |
| 8h     | SDRAM_CONFIG        |  | <a href="#">Section 7.3.5.3</a>  |
| Ch     | SDRAM_CONFIG_2      |  | <a href="#">Section 7.3.5.4</a>  |
| 10h    | SDRAM_REF_CTRL      |  | <a href="#">Section 7.3.5.5</a>  |
| 14h    | SDRAM_REF_CTRL_SHDW |  | <a href="#">Section 7.3.5.6</a>  |
| 18h    | SDRAM_TIM_1         |  | <a href="#">Section 7.3.5.7</a>  |
| 1Ch    | SDRAM_TIM_1_SHDW    |  | <a href="#">Section 7.3.5.8</a>  |
| 20h    | SDRAM_TIM_2         |  | <a href="#">Section 7.3.5.9</a>  |
| 24h    | SDRAM_TIM_2_SHDW    |  | <a href="#">Section 7.3.5.10</a> |
| 28h    | SDRAM_TIM_3         |  | <a href="#">Section 7.3.5.11</a> |
| 2Ch    | SDRAM_TIM_3_SHDW    |  | <a href="#">Section 7.3.5.12</a> |
| 38h    | PWR_MGMT_CTRL       |  | <a href="#">Section 7.3.5.13</a> |
| 3Ch    | PWR_MGMT_CTRL_SHDW  |  | <a href="#">Section 7.3.5.14</a> |
| 54h    | INT_CONFIG          | Interface Configuration Register                     | <a href="#">Section 7.3.5.15</a> |
| 58h    | INT_CFG_VAL_1       | Interface Configuration Value 1 Register             | <a href="#">Section 7.3.5.16</a> |
| 5Ch    | INT_CFG_VAL_2       | Interface Configuration Value 2 Register             | <a href="#">Section 7.3.5.17</a> |
| 80h    | PERF_CNT_1          |  | <a href="#">Section 7.3.5.18</a> |
| 84h    | PERF_CNT_2          |  | <a href="#">Section 7.3.5.19</a> |
| 88h    | PERF_CNT_CFG        |  | <a href="#">Section 7.3.5.20</a> |
| 8Ch    | PERF_CNT_SEL        |  | <a href="#">Section 7.3.5.21</a> |
| 90h    | PERF_CNT_TIM        |  | <a href="#">Section 7.3.5.22</a> |
| 98h    | READ_IDLE_CTRL      |  | <a href="#">Section 7.3.5.23</a> |
| 9Ch    | READ_IDLE_CTRL_SHDW |  | <a href="#">Section 7.3.5.24</a> |
| A4h    | IRQSTATUS_RAW_SYS   |  | <a href="#">Section 7.3.5.25</a> |
| ACh    | IRQSTATUS_SYS       |  | <a href="#">Section 7.3.5.26</a> |
| B4h    | IRQENABLE_SET_SYS   |  | <a href="#">Section 7.3.5.27</a> |
| BCh    | IRQENABLE_CLR_SYS   |  | <a href="#">Section 7.3.5.28</a> |
| C8h    | ZQ_CONFIG           |  | <a href="#">Section 7.3.5.29</a> |
| D4h    | RDWR_LVL_RMP_WIN    | Read-Write Leveling Ramp Window Register             | <a href="#">Section 7.3.5.30</a> |
| D8h    | RDWR_LVL_RMP_CTRL   | Read-Write Leveling Ramp Control Register            | <a href="#">Section 7.3.5.31</a> |
| DCh    | RDWR_LVL_CTRL       | Read-Write Leveling Control Register                 | <a href="#">Section 7.3.5.32</a> |
| E4h    | DDR_PHY_CTRL_1      |  | <a href="#">Section 7.3.5.33</a> |
| E8h    | DDR_PHY_CTRL_1_SHDW |  | <a href="#">Section 7.3.5.34</a> |
| 100h   | PRI_COS_MAP         | Priority to Class of Service Mapping Register        | <a href="#">Section 7.3.5.35</a> |
| 104h   | CONNID_COS_1_MAP    | Connection ID to Class of Service 1 Mapping Register | <a href="#">Section 7.3.5.36</a> |
| 108h   | CONNID_COS_2_MAP    | Connection ID to Class of Service 2 Mapping Register | <a href="#">Section 7.3.5.37</a> |
| 120h   | RD_WR_EXEC_THRSH    | Read Write Execution Threshold Register              | <a href="#">Section 7.3.5.38</a> |



### 7.3.5.1 EMIF\_MOD\_ID\_REV Register (offset = 0h) [reset = 40440C03h]

EMIF\_MOD\_ID\_REV is shown in [Figure 7-91](#) and described in [Table 7-110](#).

**Figure 7-91. EMIF\_MOD\_ID\_REV Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-110. EMIF\_MOD\_ID\_REV Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-30 | reg_scheme         | R    | 1h    | Used to distinguish between old and current schemes. |
| 29-28 | Reserved           | R    | 0h    |  |
| 27-16 | reg_module_id      | R    | 44h   | EMIF module ID.                                      |
| 15-11 | reg_rtl_version    | R    | 1h    | RTL Version.   |
| 10-8  | reg_major_revision | R    | 4h    | Major Revision.                                      |
| 7-6   | Reserved           | R    | 0h    |  |
| 5-0   | reg_minor_revision | R    | 3h    | Minor Revision.                                      |

### 7.3.5.2 STATUS Register (offset = 4h) [reset = 0h]

STATUS is shown in [Figure 7-92](#) and described in [Table 7-111](#).

**Figure 7-92. STATUS Register**

| 31     | 30                | 29            | 28 | 27       | 26                | 25 | 24       |
|--------|-------------------|---------------|----|----------|-------------------|----|----------|
| reg_be | reg_dual_clk_mode | reg_fast_init |    |          | Reserved          |    |          |
| R-0h   | R-0h              | R-0h          |    |          | R-0h              |    |          |
| 23     | 22                | 21            | 20 | 19       | 18                | 17 | 16       |
|        |                   |               |    | Reserved |                   |    |          |
|        |                   |               |    | R-0h     |                   |    |          |
| 15     | 14                | 13            | 12 | 11       | 10                | 9  | 8        |
|        |                   |               |    | Reserved |                   |    |          |
|        |                   |               |    | R-0h     |                   |    |          |
| 7      | 6                 | 5             | 4  | 3        | 2                 | 1  | 0        |
|        |                   | Reserved      |    |          | reg_phy_dll_ready |    | Reserved |
|        |                   | R-0h          |    |          | R-0h              |    | R-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-111. STATUS Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31   | reg_be            | R    | 0h    | Big Endian.<br>Reflects the value on the config_big_endian port that defines whether the EMIF is in big or little endian mode.<br>0 = Little endian.<br>1 = Big endian.                             |
| 30   | reg_dual_clk_mode | R    | 0h    | Dual Clock mode.<br>Reflects the value on the config_dual_clk_mode port that defines whether the ocp_clk and m_clk are asynchronous.<br>0 = ocp_clk = m_clk.<br>1 = Asynchronous ocp_clk and m_clk. |
| 29   | reg_fast_init     | R    | 0h    | Fast Init.<br>Reflects the value on the config_fast_init port that defines whether the EMIF fast initialization mode has been enabled.<br>0 = Fast init disabled.<br>1 = Fast init enabled.         |
| 28-3 | Reserved          | R    | 0h    |   |
| 2    | reg_phy_dll_ready | R    | 0h    | DDR PHY Ready.<br>Reflects the value on the phy_ready port (active high) that defines whether the DDR PHY is ready for normal operation.  |
| 1-0  | Reserved          | R    | 0h    |   |

### 7.3.5.3 SDRAM\_CONFIG Register (offset = 8h) [reset = 0h]

SDRAM\_CONFIG is shown in [Figure 7-93](#) and described in [Table 7-112](#).

**Figure 7-93. SDRAM\_CONFIG Register**

|                 |             |        |                     |                 |              |         |    |
|-----------------|-------------|--------|---------------------|-----------------|--------------|---------|----|
| 31              | 30          | 29     | 28                  | 27              | 26           | 25      | 24 |
| reg_sdram_type  |             |        | reg_ibank_pos       |                 | reg_ddr_term |         |    |
| R/W-0h          |             |        | R/W-0h              |                 | R/W-0h       |         |    |
| 23              | 22          | 21     | 20                  | 19              | 18           | 17      | 16 |
| reg_ddr2_ddqs   | reg_dyn_odt |        | reg_ddr_disable_dll | reg_sdram_drive |              | reg_cwl |    |
| R/W-0h          |             | R/W-0h | R/W-0h              | R/W-0h          | R/W-0h       |         |    |
| 15              | 14          | 13     | 12                  | 11              | 10           | 9       | 8  |
| reg_narrow_mode |             | reg_cl |                     |                 | reg_rowsize  |         |    |
| R/W-0h          |             |        | R/W-0h              |                 |              | R/W-0h  |    |
| 7               | 6           | 5      | 4                   | 3               | 2            | 1       | 0  |
| reg_rowsize     | reg_ibank   |        | reg_ebank           |                 | reg_pagesize |         |    |
| R/W-0h          |             | R/W-0h | R/W-0h              | R/W-0h          |              | R/W-0h  |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-112. SDRAM\_CONFIG Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description  |
|-------|---------------------|------|-------|--|
| 31-29 | reg_sdram_type      | R/W  | 0h    | SDRAM Type selection.<br>Set to 0 for DDR1, set to 1 for LPDDR1, set to 2 for DDR2, set to 3 for DDR3.<br>All other values are reserved.   |
| 28-27 | reg_ibank_pos       | R/W  | 0h    | Internal bank position.<br>Set to 0 to assign internal bank address bits from lower OCP address bits, as shown in the tables for OCP Address to DDR2/3/mDDR Address Mapping.<br>Set to 1, 2, or 3 to assign internal bank address bits from higher OCP address, as shown in the tables for OCP Address to DDR2/3/mDDR Address Mapping. |
| 26-24 | reg_ddr_term        | R/W  | 0h    | DDR2 and DDR3 termination resistor value.<br>Set to 0 to disable termination.<br>For DDR2, set to 1 for 75 ohm, set to 2 for 150 ohm, and set to 3 for 50 ohm.<br>For DDR3, set to 1 for RZQ/4, set to 2 for RZQ/2, set to 3 for RZQ/6, set to 4 for RZQ/12, and set to 5 for RZQ/8.<br>All other values are reserved.                 |
| 23    | reg_ddr2_ddqs       | R/W  | 0h    | DDR2 differential DQS enable.<br>Set to 0 for single ended DQS.<br>Set to 1 for differential DQS.  |
| 22-21 | reg_dyn_odt         | R/W  | 0h    | DDR3 Dynamic ODT.<br>Set to 0 to turn off dynamic ODT.<br>Set to 1 for RZQ/4 and set to 2 for RZQ/2.<br>All other values are reserved.   |
| 20    | reg_ddr_disable_dll | R/W  | 0h    | Disable DLL select.<br>Set to 1 to disable DLL inside SDRAM.   |
| 19-18 | reg_sdram_drive     | R/W  | 0h    | SDRAM drive strength.<br>For DDR1/DDR2, set to 0 for normal, and set to 1 for weak drive strength.<br>For DDR3, set to 0 for RZQ/6 and set to 1 for RZQ/7.<br>For LPDDR1, set to 0 for full, set to 1 for 1/2, set to 2 for 1/4, and set to 3 for 1/8 drive strength.<br>All other values are reserved.                                |
| 17-16 | reg_cwl             | R/W  | 0h    | DDR3 CAS Write latency.<br>Value of 0, 1, 2, and 3 (CAS write latency of 5, 6, 7, and 8) are supported.<br>Use the lowest value supported for best performance.<br>All other values are reserved.  |

**Table 7-112. SDRAM\_CONFIG Register Field Descriptions (continued)**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 15-14 | reg_narrow_mode | R/W  | 0h    | SDRAM data bus width.<br>Set to 0 for 32-bit and set to 1 for 16-bit.<br>All other values are reserved.   |
| 13-10 | reg_cl          | R/W  | 0h    | CAS Latency.<br>The value of this field defines the CAS latency to be used when accessing connected SDRAM devices.<br>Value of 2, 3, 5, and 6 (CAS latency of 2, 3, 1.5, and 2.5) are supported for DDR1.<br>Value of 2, 3, 4, and 5 (CAS latency of 2, 3, 4, and 5) are supported for DDR2.<br>Value of 2, 4, 6, 8, 10, 12, and 14 (CAS latency of 5, 6, 7, 8, 9, 10, and 11) are supported for DDR3.<br>Value of 2 and 3 (CAS latency of 2 and 3) are supported for LPDDR1.<br>All other values are reserved. |
| 9-7   | reg_rowsize     | R/W  | 0h    | Row Size.<br>Defines the number of row address bits of connected SDRAM devices.<br>Set to 0 for 9 row bits, set to 1 for 10 row bits, set to 2 for 11 row bits, set to 3 for 12 row bits, set to 4 for 13 row bits, set to 5 for 14 row bits, set to 6 for 15 row bits, and set to 7 for 16 row bits.<br>This field is only used when reg_ibank_pos field in SDRAM Config register is set to 1, 2, or 3, or reg_ebank_pos field in SDRAM Config_2 register is set to 1.   |
| 6-4   | reg_ibank       | R/W  | 0h    | Internal Bank setup.<br>Defines number of banks inside connected SDRAM devices.<br>Set to 0 for 1 bank, set to 1 for 2 banks, set to 2 for 4 banks, and set to 3 for 8 banks.<br>All other values are reserved.   |
| 3     | reg_ebank       | R/W  | 0h    | External chip select setup.<br>Defines whether SDRAM accesses will use 1 or 2 chip select lines.<br>Set to 0 to use pad_cs_o_n[0] only.<br>All other values reserved.   |
| 2-0   | reg_pagesize    | R/W  | 0h    | Page Size.<br>Defines the internal page size of connected SDRAM devices.<br>Set to 0 for 256-word page (8 column bits), set to 1 for 512-word page (9 column bits), set to 2 for 1024-word page (10 column bits), and set to 3 for 2048-word page (11 column bits).<br>All other values are reserved.   |

### 7.3.5.4 SDRAM\_CONFIG\_2 Register (offset = Ch) [reset = 0h]

SDRAM\_CONFIG\_2 is shown in [Figure 7-94](#) and described in [Table 7-113](#).

**Figure 7-94. SDRAM\_CONFIG\_2 Register**

| 31       | 30       | 29       | 28 | 27            | 26 | 25       | 24 |
|----------|----------|----------|----|---------------|----|----------|----|
| Reserved | Reserved | Reserved |    | reg_ebank_pos |    | Reserved |    |
| R-0h     | R/W-0h   | R-0h     |    | R/W-0h        |    | R-0h     |    |
| 23       | 22       | 21       | 20 | 19            | 18 | 17       | 16 |
| Reserved |          |          |    |               |    |          |    |
| R-0h     |          |          |    |               |    |          |    |
| 15       | 14       | 13       | 12 | 11            | 10 | 9        | 8  |
| Reserved |          |          |    |               |    |          |    |
| R-0h     |          |          |    |               |    |          |    |
| 7        | 6        | 5        | 4  | 3             | 2  | 1        | 0  |
| Reserved |          | Reserved |    | Reserved      |    | Reserved |    |
| R-0h     |          | R/W-0h   |    | R-0h          |    | R/W-0h   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-113. SDRAM\_CONFIG\_2 Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31    | Reserved      | R    | 0h    |   |
| 30    | Reserved      | R/W  | 0h    | Reserved.   |
| 29-28 | Reserved      | R    | 0h    |   |
| 27    | reg_ebank_pos | R/W  | 0h    | External bank position.<br>Set to 0 to assign external bank address bits from lower OCP address, as shown in the tables for OCP Address to DDR2/3/mDDR Address Mapping.<br>Set to 1 to assign external bank address bits from higher OCP address bits, as shown in the tables for OCP Address to DDR2/3/mDDR Address Mapping. |
| 26-6  | Reserved      | R    | 0h    |   |
| 5-4   | Reserved      | R/W  | 0h    | Reserved.   |
| 3     | Reserved      | R    | 0h    |   |
| 2-0   | Reserved      | R/W  | 0h    | Reserved.   |

### 7.3.5.5 SDRAM\_REF\_CTRL Register (offset = 10h) [reset = 0h]

SDRAM\_REF\_CTRL is shown in [Figure 7-95](#) and described in [Table 7-114](#).

**Figure 7-95. SDRAM\_REF\_CTRL Register**

| 31               | 30       | 29      | 28      | 27       | 26 | 25       | 24 |
|------------------|----------|---------|---------|----------|----|----------|----|
| reg_initref_dis  | Reserved | reg_srt | reg_asr | Reserved |    | reg_pasr |    |
| R/W-0h           | R-0h     | R/W-0h  | R/W-0h  | R-0h     |    | R/W-0h   |    |
| Reserved         |          |         |         |          |    |          |    |
| 23               | 22       | 21      | 20      | 19       | 18 | 17       | 16 |
| R-0h             |          |         |         |          |    |          |    |
| 15               | 14       | 13      | 12      | 11       | 10 | 9        | 8  |
| reg_refresh_rate |          |         |         |          |    |          |    |
| R/W-0h           |          |         |         |          |    |          |    |
| 7                | 6        | 5       | 4       | 3        | 2  | 1        | 0  |
| reg_refresh_rate |          |         |         |          |    |          |    |
| R/W-0h           |          |         |         |          |    |          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

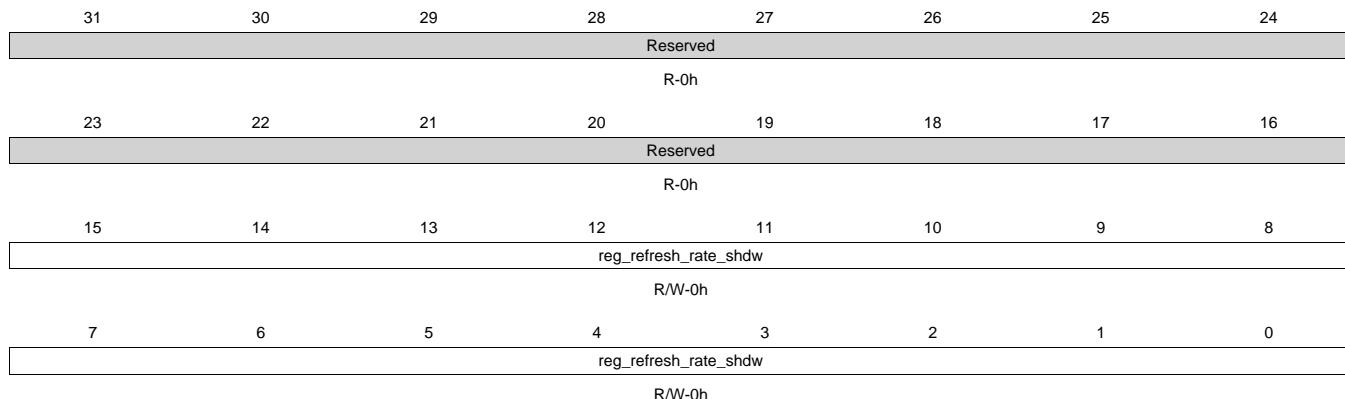
**Table 7-114. SDRAM\_REF\_CTRL Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31    | reg_initref_dis  | R/W  | 0h    | Initialization and Refresh disable.<br>When set to 1, EMIF will disable SDRAM initialization and refreshes, but will carry out SDRAM write/read transactions.   |
| 30    | Reserved         | R    | 0h    |   |
| 29    | reg_srt          | R/W  | 0h    | DDR3 Self Refresh temperature range.<br>Set to 0 for normal operating temperature range and set to 1 for extended operating temperature range when the reg_asr field is set to 0.<br>This bit must be set to 0 if the reg_asr field is set to 1.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.   |
| 28    | reg_asr          | R/W  | 0h    | DDR3 Auto Self Refresh enable.<br>Set to 1 for auto Self Refresh enable.<br>Set to 0 for manual Self Refresh reference indicated by the reg_srt field.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence.   |
| 27    | Reserved         | R    | 0h    |   |
| 26-24 | reg_pasr         | R/W  | 0h    | Partial Array Self Refresh.<br>These bits get loaded into the Extended Mode Register of an LPDDR1 or DDR3 during initialization.<br>For LPDDR1, set to 0 for full array, set to 1 for 1/2 array, set to 2 for 1/4 array, set to 5 for 1/8 array, and set to 6 for 1/16 array to be refreshed.<br>For DDR3, set to 0 for full array, set to 1 or 5 for 1/2 array, set to 2 or 6 for 1/4 array, set to 3 or 7 for 1/8 array, and set to 4 for 3/4 array to be refreshed.<br>All other values are reserved.<br>A write to this field will cause the EMIF to start the SDRAM initialization sequence. |
| 23-16 | Reserved         | R    | 0h    |   |
| 15-0  | reg_refresh_rate | R/W  | 0h    | Refresh Rate.<br>Value in this field is used to define the rate at which connected SDRAM devices will be refreshed.<br>SDRAM refresh rate = EMIF rate / reg_refresh_rate where EMIF rate is equal to m_clk rate.<br>If reg_refresh_rate andlt (8*reg_t_rfc)+reg_t_rp+reg_t_rcd+20 then it will be loaded with (8*reg_t_rfc)+reg_t_rp+reg_t_rcd+20.<br>This is done to avoid lock-up situations when illegal values are programmed.  |

### 7.3.5.6 SDRAM\_REF\_CTRL\_SHDW Register (offset = 14h) [reset = 0h]

SDRAM\_REF\_CTRL\_SHDW is shown in [Figure 7-96](#) and described in [Table 7-115](#).

**Figure 7-96. SDRAM\_REF\_CTRL\_SHDW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-115. SDRAM\_REF\_CTRL\_SHDW Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description   |
|-------|-----------------------|------|-------|---|
| 31-16 | Reserved              | R    | 0h    |   |
| 15-0  | reg_refresh_rate_shdw | R/W  | 0h    | Shadow field for reg_refresh_rate.<br>This field is loaded into reg_refresh_rate field in SDRAM Refresh Control register when SIdleAck is asserted.<br>This register is not auto corrected when the value is invalid. |

### 7.3.5.7 SDRAM\_TIM\_1 Register (offset = 18h) [reset = 0h]

SDRAM\_TIM\_1 is shown in Figure 7-97 and described in Table 7-116.

**Figure 7-97. SDRAM\_TIM\_1 Register**

|           |    |           |          |        |           |           |    |
|-----------|----|-----------|----------|--------|-----------|-----------|----|
| 31        | 30 | 29        | 28       | 27     | 26        | 25        | 24 |
| Reserved  |    | reg_t_rp  |          |        |           | reg_t_rcd |    |
| R-0h      |    |           |          | R/W-0h |           | R/W-0h    |    |
| 23        | 22 | 21        | 20       | 19     | 18        | 17        | 16 |
| reg_t_rcd |    | reg_t_wr  |          |        |           | reg_t_ras |    |
| R/W-0h    |    |           |          | R/W-0h |           | R/W-0h    |    |
| 15        | 14 | 13        | 12       | 11     | 10        | 9         | 8  |
| reg_t_ras |    |           | reg_t_rc |        |           |           |    |
| R/W-0h    |    |           |          | R/W-0h |           |           |    |
| 7         | 6  | 5         | 4        | 3      | 2         | 1         | 0  |
| reg_t_rc  |    | reg_t_rrd |          |        | reg_t_wtr |           |    |
| R/W-0h    |    |           | R/W-0h   |        |           | R/W-0h    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-116. SDRAM\_TIM\_1 Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-29 | Reserved  | R    | 0h    |  |
| 28-25 | reg_t_rp  | R/W  | 0h    | Minimum number of m_clk cycles from Precharge to Activate or Refresh, minus one.   |
| 24-21 | reg_t_rcd | R/W  | 0h    | Minimum number of m_clk cycles from Activate to Read or Write, minus one.  |
| 20-17 | reg_t_wr  | R/W  | 0h    | Minimum number of m_clk cycles from last Write transfer to Pre-charge, minus one.<br>The SDRAM initialization sequence will be started when the value of this field is changed from the previous value and the EMIF is in DDR2 mode. |
| 16-12 | reg_t_ras | R/W  | 0h    | Minimum number of m_clk cycles from Activate to Pre-charge, minus one.<br>reg_t_ras >= reg_t_rcd.  |
| 11-6  | reg_t_rc  | R/W  | 0h    | Minimum number of m_clk cycles from Activate to Activate, minus one.   |
| 5-3   | reg_t_rrd | R/W  | 0h    | Minimum number of m_clk cycles from Activate to Activate for a different bank, minus one.<br>For an 8-bank DDR2, this field must be equal to (((4*tRRD)+(2*tCK))/(4*tCK))-1).  |
| 2-0   | reg_t_wtr | R/W  | 0h    | Minimum number of m_clk cycles from last Write to Read, minus one.   |

### 7.3.5.8 SDRAM\_TIM\_1\_SHDW Register (offset = 1Ch) [reset = 0h]

SDRAM\_TIM\_1\_SHDW is shown in Figure 7-98 and described in Table 7-117.

**Figure 7-98. SDRAM\_TIM\_1\_SHDW Register**

|                |    |                |    |        |    |                |    |
|----------------|----|----------------|----|--------|----|----------------|----|
| 31             | 30 | 29             | 28 | 27     | 26 | 25             | 24 |
| Reserved       |    | reg_t_rp_shdw  |    |        |    | reg_t_rcd_shdw |    |
| R-0h           |    |                |    | R/W-0h |    | R/W-0h         |    |
| 23             | 22 | 21             | 20 | 19     | 18 | 17             | 16 |
| reg_t_rcd_shdw |    | reg_t_wr_shdw  |    |        |    | reg_t_ras_shdw |    |
| R/W-0h         |    |                |    | R/W-0h |    | R/W-0h         |    |
| 15             | 14 | 13             | 12 | 11     | 10 | 9              | 8  |
| reg_t_ras_shdw |    | reg_t_rc_shdw  |    |        |    | reg_t_rc_shdw  |    |
| R/W-0h         |    |                |    | R/W-0h |    |                |    |
| 7              | 6  | 5              | 4  | 3      | 2  | 1              | 0  |
| reg_t_rc_shdw  |    | reg_t_rrd_shdw |    |        |    | reg_t_wtr_shdw |    |
| R/W-0h         |    |                |    | R/W-0h |    |                |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-117. SDRAM\_TIM\_1\_SHDW Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description  |
|-------|----------------|------|-------|--|
| 31-29 | Reserved       | R    | 0h    |  |
| 28-25 | reg_t_rp_shdw  | R/W  | 0h    | Shadow field for reg_t_rp.<br>This field is loaded into reg_t_rp field in SDRAM Timing 1 register when SlidleAck is asserted.  |
| 24-21 | reg_t_rcd_shdw | R/W  | 0h    | Shadow field for reg_t_rcd.<br>This field is loaded into reg_t_rcd field in SDRAM Timing 1 register when SlidleAck is asserted.  |
| 20-17 | reg_t_wr_shdw  | R/W  | 0h    | Shadow field for reg_t_wr.<br>This field is loaded into reg_t_wr field in SDRAM Timing 1 register when SlidleAck is asserted.<br>initialization sequence will be started when the value of this field is changed from the previous value and the EMIF is in DDR2 mode. |
| 16-12 | reg_t_ras_shdw | R/W  | 0h    | Shadow field for reg_t_ras.<br>This field is loaded into reg_t_ras field in SDRAM Timing 1 register when SlidleAck is asserted.  |
| 11-6  | reg_t_rc_shdw  | R/W  | 0h    | Shadow field for reg_t_rc.<br>This field is loaded into reg_t_rc field in SDRAM Timing 1 register when SlidleAck is asserted.  |
| 5-3   | reg_t_rrd_shdw | R/W  | 0h    | Shadow field for reg_t_rrd.<br>This field is loaded into reg_t_rrd field in SDRAM Timing 1 register when SlidleAck is asserted.  |
| 2-0   | reg_t_wtr_shdw | R/W  | 0h    | Shadow field for reg_t_wtr.<br>This field is loaded into reg_t_wtr field in SDRAM Timing 1 register when SlidleAck is asserted.  |

### 7.3.5.9 SDRAM\_TIM\_2 Register (offset = 20h) [reset = 0h]

SDRAM\_TIM\_2 is shown in Figure 7-99 and described in Table 7-118.

**Figure 7-99. SDRAM\_TIM\_2 Register**

| 31         | 30 | 29        | 28 | 27         | 26        | 25 | 24         |
|------------|----|-----------|----|------------|-----------|----|------------|
| Reserved   |    | reg_t_xp  |    |            | reg_t_odt |    | reg_t_xsnr |
| R-0h       |    | R/W-0h    |    |            | R/W-0h    |    | R/W-0h     |
| 23         | 22 | 21        | 20 | 19         | 18        | 17 | 16         |
|            |    |           |    | reg_t_xsnr |           |    | R/W-0h     |
| 15         | 14 | 13        | 12 | 11         | 10        | 9  | 8          |
|            |    |           |    | reg_t_xsrd |           |    | R/W-0h     |
| 7          | 6  | 5         | 4  | 3          | 2         | 1  | 0          |
| reg_t_xsrd |    | reg_t_rtp |    |            | reg_t_cke |    | R/W-0h     |
|            |    | R/W-0h    |    |            | R/W-0h    |    | R/W-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-118. SDRAM\_TIM\_2 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31    | Reserved   | R    | 0h    |  |
| 30-28 | reg_t_xp   | R/W  | 0h    | Minimum number of m_clk cycles from Powerdown exit to any command other than a Read command, minus one. For DDR2 and LPDDR1, this field must satisfy greater of tXP or tCKE. |
| 27-25 | reg_t_odt  | R/W  | 0h    | Minimum number of m_clk cycles from ODT enable to write data driven for DDR2 and DDR3. reg_t_odt must be equal to tAOND.   |
| 24-16 | reg_t_xsnr | R/W  | 0h    | Minimum number of m_clk cycles from Self-Refresh exit to any command other than a Read command, minus one.   |
| 15-6  | reg_t_xsrd | R/W  | 0h    | Minimum number of m_clk cycles from Self-Refresh exit to a Read command, minus one.  |
| 5-3   | reg_t_rtp  | R/W  | 0h    | Minimum number of m_clk cycles from the last Read command to a Pre-charge command for DDR2, minus one.   |
| 2-0   | reg_t_cke  | R/W  | 0h    | Minimum number of m_clk cycles between pad_cke_o changes, minus one.   |

### 7.3.5.10 SDRAM\_TIM\_2\_SHDW Register (offset = 24h) [reset = 0h]

SDRAM\_TIM\_2\_SHDW is shown in [Figure 7-100](#) and described in [Table 7-119](#).

**Figure 7-100. SDRAM\_TIM\_2\_SHDW Register**

| 31              | 30 | 29             | 28              | 27             | 26 | 25              | 24 |
|-----------------|----|----------------|-----------------|----------------|----|-----------------|----|
| Reserved        |    | reg_t_xp_shdw  |                 | reg_t_odt_shdw |    | reg_t_xsnr_shdw |    |
| R-0h            |    | R/W-0h         |                 | R/W-0h         |    | R/W-0h          |    |
| 23              | 22 | 21             | 20              | 19             | 18 | 17              | 16 |
|                 |    |                | reg_t_xsnr_shdw |                |    |                 |    |
|                 |    |                | R/W-0h          |                |    |                 |    |
| 15              | 14 | 13             | 12              | 11             | 10 | 9               | 8  |
|                 |    |                | reg_t_xsrd_shdw |                |    |                 |    |
|                 |    |                | R/W-0h          |                |    |                 |    |
| 7               | 6  | 5              | 4               | 3              | 2  | 1               | 0  |
| reg_t_xsrd_shdw |    | reg_t_rtp_shdw |                 | reg_t_cke_shdw |    |                 |    |
| R/W-0h          |    | R/W-0h         |                 | R/W-0h         |    |                 |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-119. SDRAM\_TIM\_2\_SHDW Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | Reserved        | R    | 0h    |   |
| 30-28 | reg_t_xp_shdw   | R/W  | 0h    | Shadow field for reg_t_xp.<br>This field is loaded into reg_t_xp field in SDRAM Timing 2 register when SdleAck is asserted.     |
| 27-25 | reg_t_odt_shdw  | R/W  | 0h    | Shadow field for reg_t_odt.<br>This field is loaded into reg_t_odt field in SDRAM Timing 2 register when SdleAck is asserted.   |
| 24-16 | reg_t_xsnr_shdw | R/W  | 0h    | Shadow field for reg_t_xsnr.<br>This field is loaded into reg_t_xsnr field in SDRAM Timing 2 register when SdleAck is asserted. |
| 15-6  | reg_t_xsrd_shdw | R/W  | 0h    | Shadow field for reg_t_xsrd.<br>This field is loaded into reg_t_xsrd field in SDRAM Timing 2 register when SdleAck is asserted. |
| 5-3   | reg_t_rtp_shdw  | R/W  | 0h    | Shadow field for reg_t_rtp.<br>This field is loaded into reg_t_rtp field in SDRAM Timing 2 register when SdleAck is asserted.   |
| 2-0   | reg_t_cke_shdw  | R/W  | 0h    | Shadow field for reg_t_cke.<br>This field is loaded into reg_t_cke field in SDRAM Timing 2 register when SdleAck is asserted.   |

### 7.3.5.11 SDRAM\_TIM\_3 Register (offset = 28h) [reset = 0h]

SDRAM\_TIM\_3 is shown in Figure 7-101 and described in Table 7-120.

**Figure 7-101. SDRAM\_TIM\_3 Register**

|               |          |    |             |               |    |        |    |  |
|---------------|----------|----|-------------|---------------|----|--------|----|--|
| 31            | 30       | 29 | 28          | 27            | 26 | 25     | 24 |  |
| reg_t_pdll_ul |          |    |             | Reserved      |    |        |    |  |
| R/W-0h        |          |    |             |               |    | R-0h   |    |  |
| 23            | 22       | 21 | 20          | 19            | 18 | 17     | 16 |  |
| Reserved      |          |    | reg_zq_zqcs |               |    |        |    |  |
| R/W-0h        |          |    |             |               |    | R/W-0h |    |  |
| 15            | 14       | 13 | 12          | 11            | 10 | 9      | 8  |  |
| reg_zq_zqcs   | Reserved |    | reg_t_rfc   |               |    |        |    |  |
| R/W-0h        |          |    | R/W-0h      |               |    |        |    |  |
| 7             | 6        | 5  | 4           | 3             | 2  | 1      | 0  |  |
| reg_t_rfc     |          |    |             | reg_t_ras_max |    |        |    |  |
| R/W-0h        |          |    |             |               |    | R/W-0h |    |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

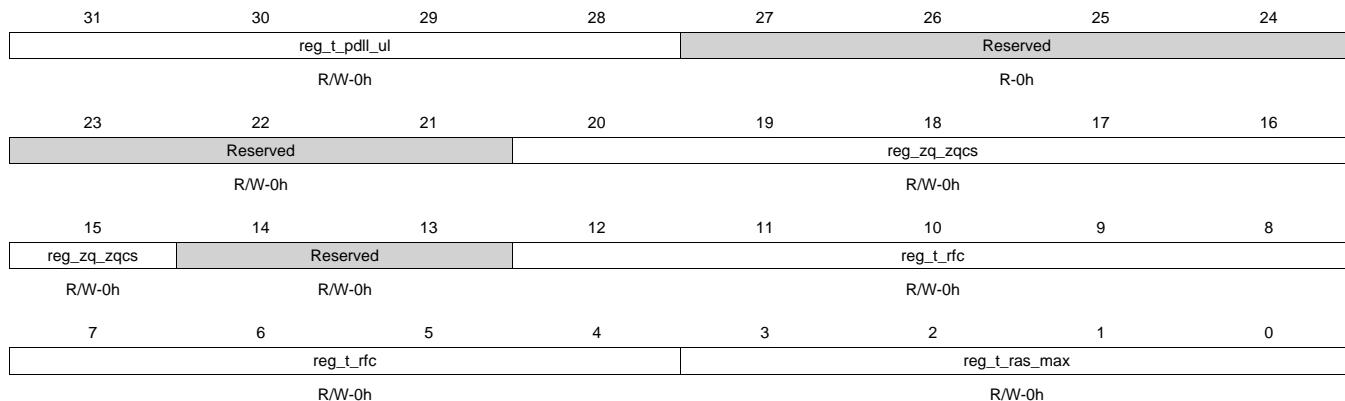
**Table 7-120. SDRAM\_TIM\_3 Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-28 | reg_t_pdll_ul | R/W  | 0h    | Minimum number of DDR clock cycles for PHY DLL to unlock. A value of N will be equal to N x 128 clocks.   |
| 27-24 | Reserved      | R    | 0h    |   |
| 23-21 | Reserved      | R/W  | 0h    | Reserved.   |
| 20-15 | reg_zq_zqcs   | R/W  | 0h    | Number of DDR clock clock cycles for a ZQCS command, minus one.   |
| 14-13 | Reserved      | R/W  | 0h    | Reserved.   |
| 12-4  | reg_t_rfc     | R/W  | 0h    | Minimum number of DDR clock cycles from Refresh or Load Mode to Refresh or Activate, minus one.   |
| 3-0   | reg_t_ras_max | R/W  | 0h    | Maximum number of reg_refresh_rate intervals from Activate to Precharge command.<br>This field must be equal to ((tRASmax / tREFI)-1) rounded down to the next lower integer.<br>This field is only applicable for mDDR.<br>This field must be programmed to 0xF for other SDRAM types. |

### 7.3.5.12 SDRAM\_TIM\_3\_SHDW Register (offset = 2Ch) [reset = 0h]

SDRAM\_TIM\_3\_SHDW is shown in [Figure 7-102](#) and described in [Table 7-121](#).

**Figure 7-102. SDRAM\_TIM\_3\_SHDW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

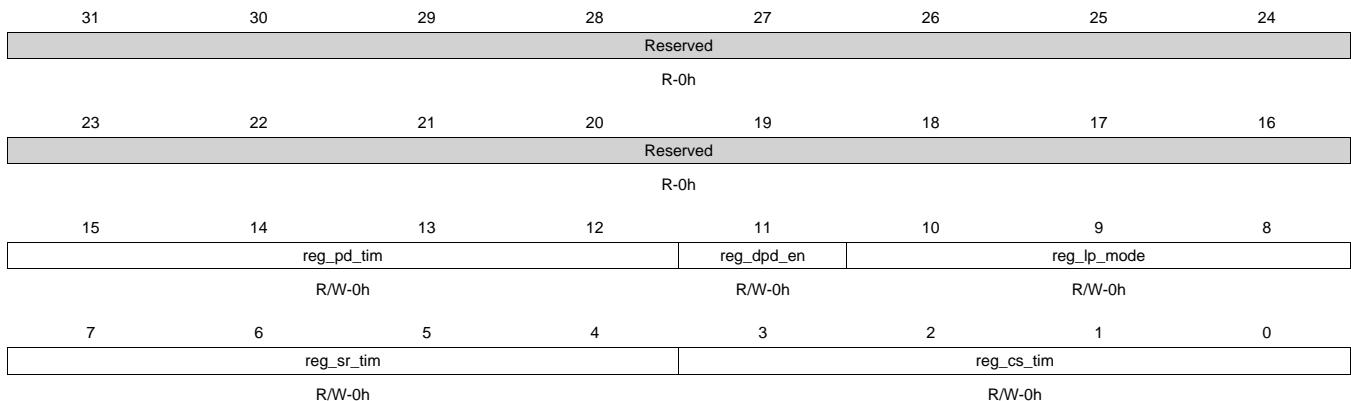
**Table 7-121. SDRAM\_TIM\_3\_SHDW Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-28 | reg_t_pdll_ul | R/W  | 0h    | Minimum number of DDR clock cycles for PHY DLL to unlock. A value of N will be equal to N x 128 clocks.   |
| 27-24 | Reserved      | R    | 0h    |   |
| 23-21 | Reserved      | R/W  | 0h    | Reserved.   |
| 20-15 | reg_zq_zqcs   | R/W  | 0h    | Number of DDR clock clock cycles for a ZQCS command, minus one.   |
| 14-13 | Reserved      | R/W  | 0h    | Reserved.   |
| 12-4  | reg_t_rfc     | R/W  | 0h    | Minimum number of DDR clock cycles from Refresh or Load Mode to Refresh or Activate, minus one.   |
| 3-0   | reg_t_ras_max | R/W  | 0h    | Maximum number of reg_refresh_rate intervals from Activate to Precharge command.<br>This field must be equal to ((tRASmax / tREFI)-1) rounded down to the next lower integer.<br>This field is only applicable for mDDR.<br>This field must be programmed to 0xF for other SDRAM types. |

### 7.3.5.13 PWR\_MGMT\_CTRL Register (offset = 38h) [reset = 0h]

PWR\_MGMT\_CTRL is shown in [Figure 7-103](#) and described in [Table 7-122](#).

**Figure 7-103. PWR\_MGMT\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-122. PWR\_MGMT\_CTRL Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-16 | Reserved    | R    | 0h    |  |
| 15-12 | reg_pd_tim  | R/W  | 0h    | Power Management timer for Power-Down.<br>The EMIF will put the external SDRAM in Power-Down mode after the EMIF is idle for these number of m_clk cycles and if reg_lp_mode field is set to 4.<br>Set to 0 to immediately enter Power-Down mode.<br>Set to 1 for 16 clocks.<br>Set to 2 for 32 clocks.<br>Set to 3 for 64 clocks.<br>Set to 4 for 128 clocks.<br>Set to 5 for 256 clocks.<br>Set to 6 for 512 clocks.<br>Set to 7 for 1024 clocks.<br>Set to 8 for 2048 clocks.<br>Set to 9 for 4096 clocks.<br>Set to 10 for 8192 clocks.<br>Set to 11 for 16384 clocks.<br>Set to 12 for 32768 clocks.<br>Set to 13 for 65536 clocks.<br>Set to 14 for 131072 clocks.<br>Set to 15 for 262144 clocks.<br>Note: After updating this field, at least one dummy read access to SDRAM is required for the new value to take affect. |
| 11    | reg_dpd_en  | R/W  | 0h    | Deep Power Down enable.<br>Set to 0 for normal operation.<br>Set to 1 to enter deep power down mode.<br>This mode will override the reg_lp_mode field setting.   |
| 10-8  | reg_lp_mode | R/W  | 0h    | Automatic Power Management enable.<br>Set to 1 for Clock Stop, set to 2 for Self Refresh, and set to 4 for Power-Down.<br>All other values will disable automatic power management.  |

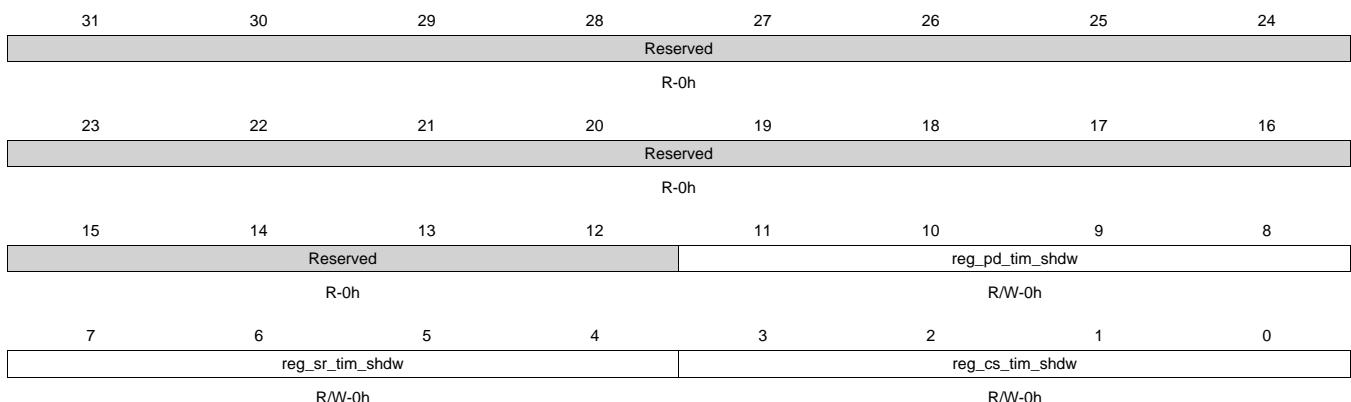
**Table 7-122. PWR\_MGMT\_CTRL Register Field Descriptions (continued)**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 7-4 | reg_sr_tim | R/W  | 0h    | <p>Power Management timer for Self Refresh.</p> <p>The EMIF will put the external SDRAM in Self Refresh mode after the EMIF is idle for these number of m_clk cycles and if reg_lp_mode field is set to 2.</p> <p>Set to 0 to immediately enter Self Refresh mode.</p> <p>Set to 1 for 16 clocks.</p> <p>Set to 2 for 32 clocks.</p> <p>Set to 3 for 64 clocks.</p> <p>Set to 4 for 128 clocks.</p> <p>Set to 5 for 256 clocks.</p> <p>Set to 6 for 512 clocks.</p> <p>Set to 7 for 1024 clocks.</p> <p>Set to 8 for 2048 clocks.</p> <p>Set to 9 for 4096 clocks.</p> <p>Set to 10 for 8192 clocks.</p> <p>Set to 11 for 16384 clocks.</p> <p>Set to 12 for 32768 clocks.</p> <p>Set to 13 for 65536 clocks.</p> <p>Set to 14 for 131072 clocks.</p> <p>Set to 15 for 262144 clocks.</p> <p>Note: After updating this field, at least one dummy read access to SDRAM is required for the new value to take affect.</p> |
| 3-0 | reg_cs_tim | R/W  | 0h    | <p>Power Management timer for Clock Stop.</p> <p>The EMIF will put the external SDRAM in Clock Stop mode after the EMIF is idle for these number of m_clk cycles and if reg_lp_mode field is set to 1.</p> <p>Set to 0 to immediately enter Clock Stop mode.</p> <p>Set to 1 for 16 clocks.</p> <p>Set to 2 for 32 clocks.</p> <p>Set to 3 for 64 clocks.</p> <p>Set to 4 for 128 clocks.</p> <p>Set to 5 for 256 clocks.</p> <p>Set to 6 for 512 clocks.</p> <p>Set to 7 for 1024 clocks.</p> <p>Set to 8 for 2048 clocks.</p> <p>Set to 9 for 4096 clocks.</p> <p>Set to 10 for 8192 clocks.</p> <p>Set to 11 for 16384 clocks.</p> <p>Set to 12 for 32768 clocks.</p> <p>Set to 13 for 65536 clocks.</p> <p>Set to 14 for 131072 clocks.</p> <p>Set to 15 for 262144 clocks.</p> <p>Note: After updating this field, at least one dummy read access to SDRAM is required for the new value to take affect.</p>       |

### 7.3.5.14 PWR\_MGMT\_CTRL\_SHDW Register (offset = 3Ch) [reset = 0h]

PWR\_MGMT\_CTRL\_SHDW is shown in [Figure 7-104](#) and described in [Table 7-123](#).

**Figure 7-104. PWR\_MGMT\_CTRL\_SHDW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-123. PWR\_MGMT\_CTRL\_SHDW Register Field Descriptions**

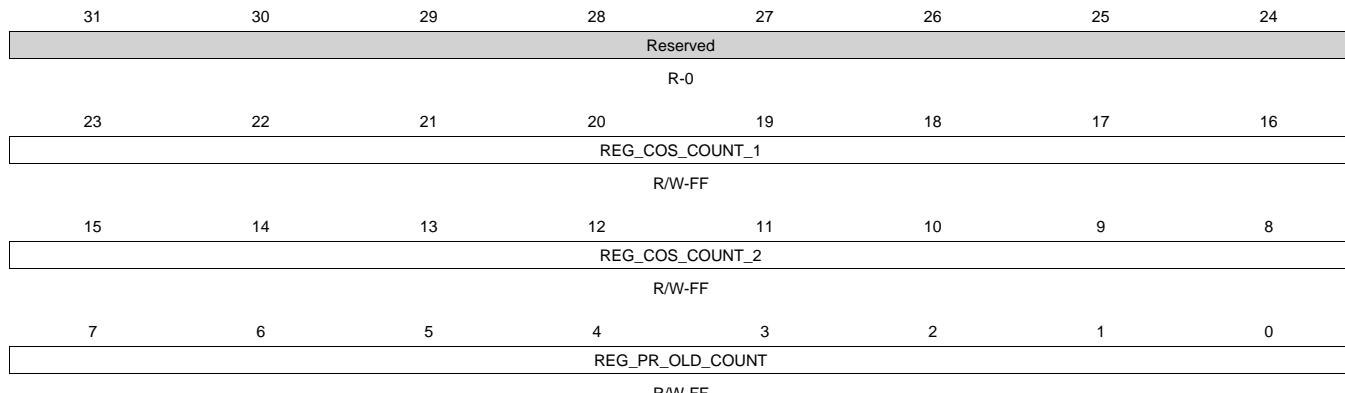
| Bit   | Field           | Type | Reset | Description  |
|-------|-----------------|------|-------|--|
| 31-12 | Reserved        | R    | 0h    |  |
| 11-8  | reg_pd_tim_shdw | R/W  | 0h    | Shadow field for reg_pd_tim.<br>This field is loaded into reg_pd_tim field in Power Management Control register when SldleAck is asserted. |
| 7-4   | reg_sr_tim_shdw | R/W  | 0h    | Shadow field for reg_sr_tim.<br>This field is loaded into reg_sr_tim field in Power Management Control register when SldleAck is asserted. |
| 3-0   | reg_cs_tim_shdw | R/W  | 0h    | Shadow field for reg_cs_tim.<br>This field is loaded into reg_cs_tim field in Power Management Control register when SldleAck is asserted. |

### 7.3.5.15 INT\_CONFIG Register (offset = 54h) [reset = 0h]

Interface Configuration Register

Interface Configuration Register is shown in [Figure 7-105](#) and described in [Table 7-124](#).

**Figure 7-105. Interface Configuration Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-124. Interface Configuration Register Field Descriptions**

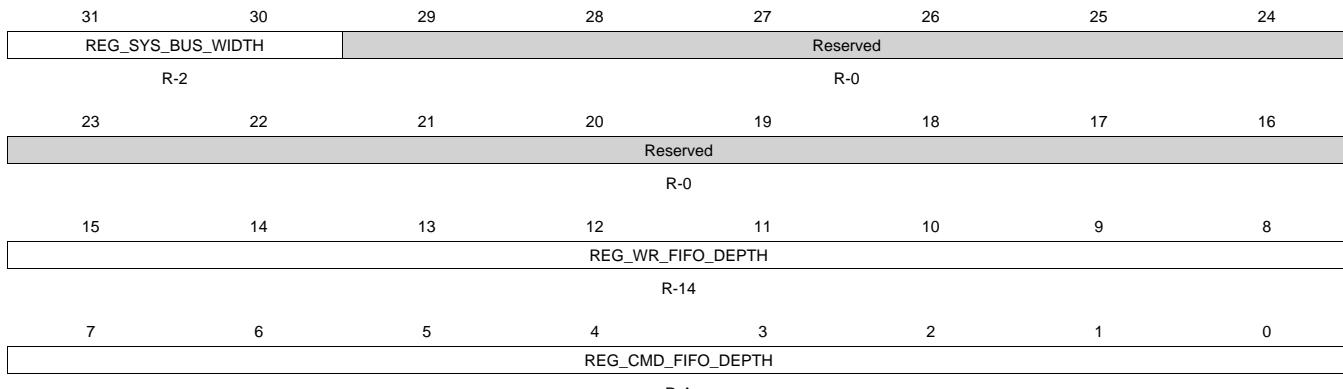
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | Reserved         | R    | R     | Reserved for future use.  |
| 23-16 | REG_COS_COUNT_1  | R/W  | 0xFF  | Priority Raise Counter for class of service 1.<br>Number of m_clk cycles after which the EMIF momentarily raises the priority of the class of service 1 commands in the Command FIFO.<br>A value of N will be equal to N x 16 clocks. |
| 15-8  | REG_COS_COUNT_2  | R/W  | 0xFF  | Priority Raise Counter for class of service 2.<br>Number of m_clk cycles after which the EMIF momentarily raises the priority of the class of service 2 commands in the Command FIFO.<br>A value of N will be equal to N x 16 clocks. |
| 7-0   | REG_PR_OLD_COUNT | R/W  | 0xFF  | Priority Raise Old Counter.<br>Number of m_clk cycles after which the EMIF momentarily raises the priority of the oldest command in the OCP Command FIFO.<br>A value of N will be equal to N x 16 clocks.                             |

### 7.3.5.16 INT\_CFG\_VAL\_1 Register (offset = 58h) [reset = 0h]

Interface Configuration Value 1 Register

Interface Configuration Value 1 Register is shown in [Figure 7-106](#) and described in [Table 7-125](#).

**Figure 7-106. Interface Configuration Value 1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-125. Interface Configuration Value 1 Register Field Descriptions**

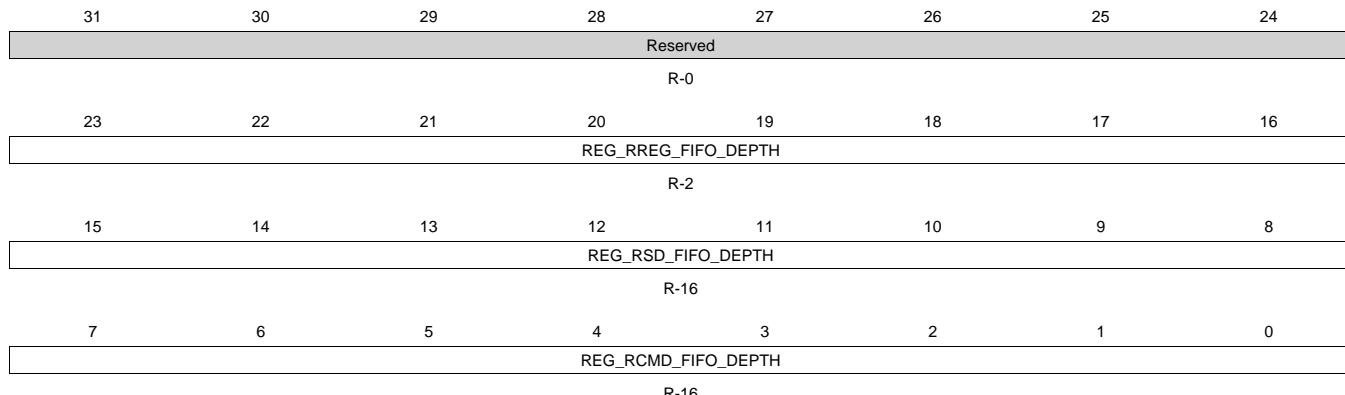
| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-30 | REG_SYS_BUS_WIDTH  | R    | 2     | L3 OCP data bus width for a particular configuration.<br>0 = 32 bit wide.<br>1 = 64 bit wide.<br>2 = 128 bit wide.<br>3 = 256 bit wide. |
| 29-16 | Reserved           | R    | 0     | Reserved for future use.  |
| 15-8  | REG_WR_FIFO_DEPTH  | R    | 0x14  | Write Data FIFO depth for a particular configuration.   |
| 7-0   | REG_CMD_FIFO_DEPTH | R    | 0xA   | Command FIFO depth for a particular configuration.  |

### 7.3.5.17 INT\_CFG\_VAL\_2 Register (offset = 5Ch) [reset = 0h]

Interface Configuration Value 2 Register

Interface Configuration Value 2 Register is shown in [Figure 7-107](#) and described in [Table 7-126](#).

**Figure 7-107. Interface Configuration Value 2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

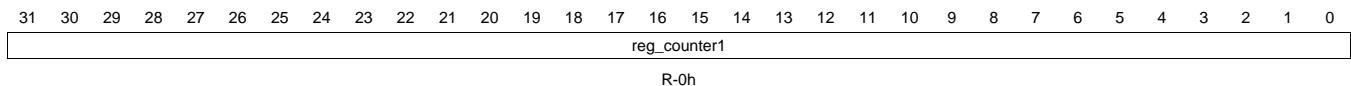
**Table 7-126. Interface Configuration Value 2 Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-24 | Reserved            | R    | 0     | Reserved for future use.                                      |
| 23-16 | REG_RREG_FIFO_DEPTH | R    | 0x2   | Register Read Data FIFO depth for a particular configuration. |
| 15-8  | REG_RSD_FIFO_DEPTH  | R    | 0x16  | SDRAM Read Data FIFO depth for a particular configuration.    |
| 7-0   | REG_RCMD_FIFO_DEPTH | R    | 0x16  | Read Command FIFO depth for a particular configuration.       |

### 7.3.5.18 PERF\_CNT\_1 Register (offset = 80h) [reset = 0h]

PERF\_CNT\_1 is shown in [Figure 7-108](#) and described in [Table 7-127](#).

**Figure 7-108. PERF\_CNT\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

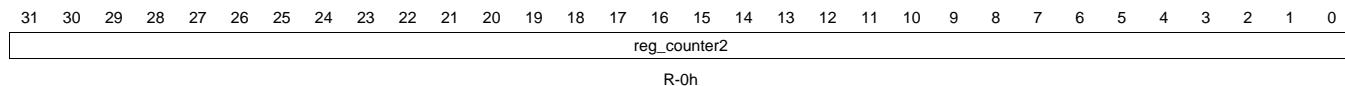
**Table 7-127. PERF\_CNT\_1 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-0 | reg_counter1 | R    | 0h    | 32-bit counter that can be configured as specified in the Performance Counter Config Register and Performance Counter Master Region Select Register. |

### 7.3.5.19 PERF\_CNT\_2 Register (offset = 84h) [reset = 0h]

PERF\_CNT\_2 is shown in [Figure 7-109](#) and described in [Table 7-128](#).

**Figure 7-109. PERF\_CNT\_2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

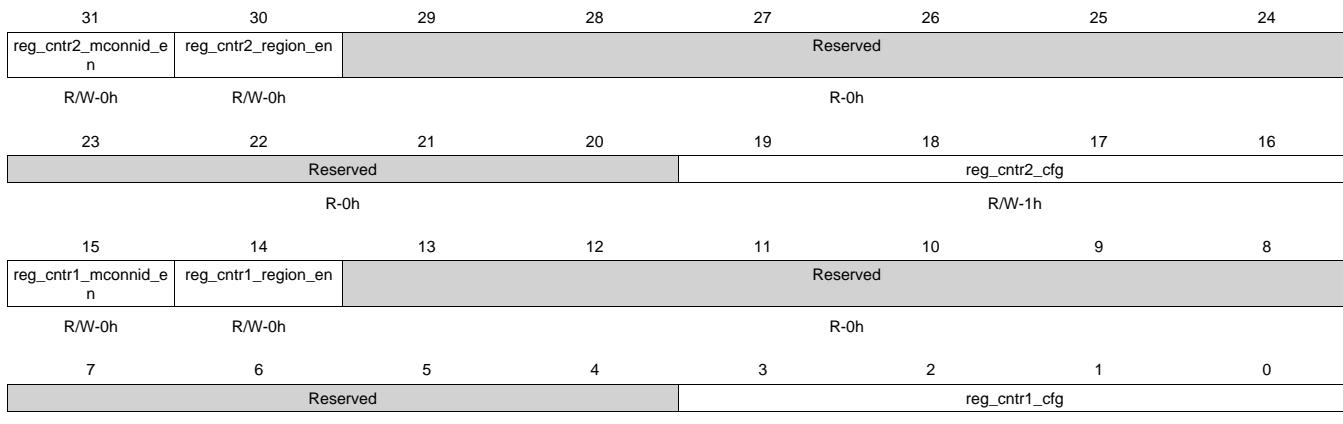
**Table 7-128. PERF\_CNT\_2 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-0 | reg_counter2 | R    | 0h    | 32-bit counter that can be configured as specified in the Performance Counter Config Register and Performance Counter Master Region Select Register. |

### 7.3.5.20 PERF\_CNT\_CFG Register (offset = 88h) [reset = 10000h]

PERF\_CNT\_CFG is shown in [Figure 7-110](#) and described in [Table 7-129](#).

**Figure 7-110. PERF\_CNT\_CFG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

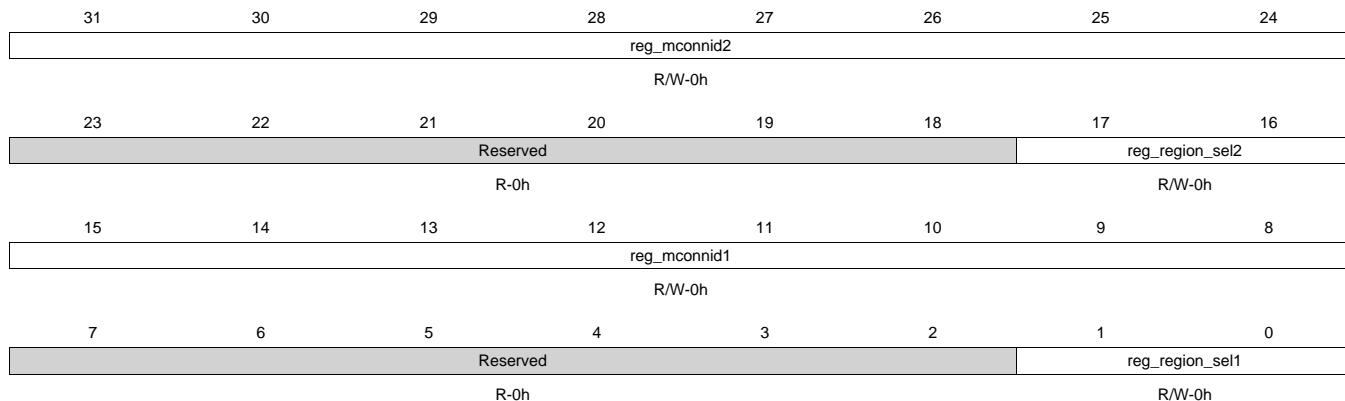
**Table 7-129. PERF\_CNT\_CFG Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31    | reg_cntr2_mconnid_en | R/W  | 0h    | MConnID filter enable for Performance Counter 2 register.                      |
| 30    | reg_cntr2_region_en  | R/W  | 0h    | Chip Select filter enable for Performance Counter 2 register.                  |
| 29-20 | Reserved             | R    | 0h    |  |
| 19-16 | reg_cntr2_cfg        | R/W  | 1h    | Filter configuration for Performance Counter 2.<br>Refer to table for details. |
| 15    | reg_cntr1_mconnid_en | R/W  | 0h    | MConnID filter enable for Performance Counter 1 register.                      |
| 14    | reg_cntr1_region_en  | R/W  | 0h    | Chip Select filter enable for Performance Counter 1 register.                  |
| 13-4  | Reserved             | R    | 0h    |  |
| 3-0   | reg_cntr1_cfg        | R/W  | 0h    | Filter configuration for Performance Counter 1.<br>Refer to table for details. |

### 7.3.5.21 PERF\_CNT\_SEL Register (offset = 8Ch) [reset = 0h]

PERF\_CNT\_SEL is shown in [Figure 7-111](#) and described in [Table 7-130](#).

**Figure 7-111. PERF\_CNT\_SEL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

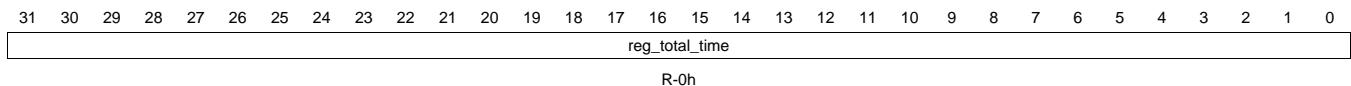
**Table 7-130. PERF\_CNT\_SEL Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description                                    |
|-------|-----------------|------|-------|--|
| 31-24 | reg_mconnid2    | R/W  | 0h    | MConnID for Performance Counter 2 register.    |
| 23-18 | Reserved        | R    | 0h    |  |
| 17-16 | reg_region_sel2 | R/W  | 0h    | MAddrSpace for Performance Counter 2 register. |
| 15-8  | reg_mconnid1    | R/W  | 0h    | MConnID for Performance Counter 1 register.    |
| 7-2   | Reserved        | R    | 0h    |  |
| 1-0   | reg_region_sel1 | R/W  | 0h    | MAddrSpace for Performance Counter 1 register. |

### 7.3.5.22 PERF\_CNT\_TIM Register (offset = 90h) [reset = 0h]

PERF\_CNT\_TIM is shown in [Figure 7-112](#) and described in [Table 7-131](#).

**Figure 7-112. PERF\_CNT\_TIM Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

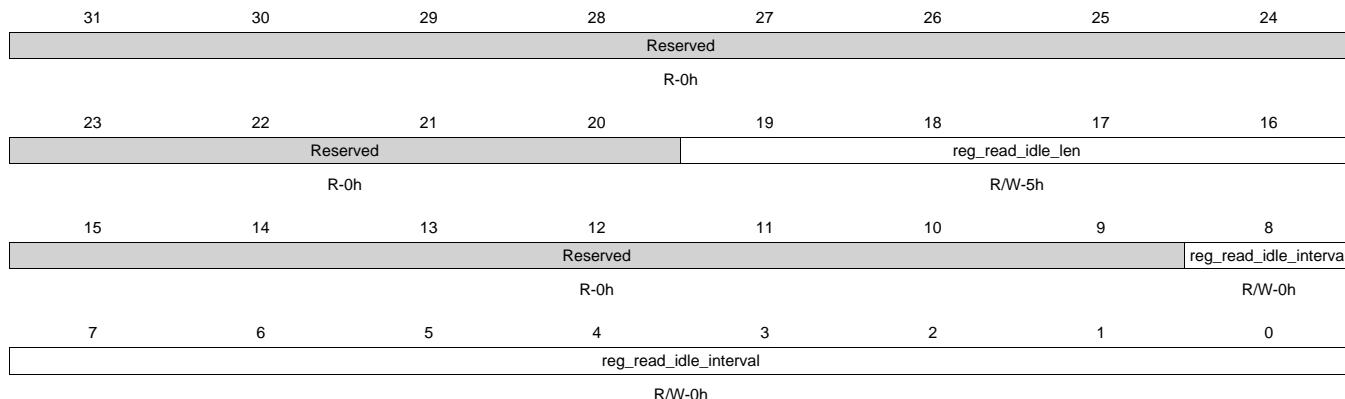
**Table 7-131. PERF\_CNT\_TIM Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-0 | reg_total_time | R    | 0h    | 32-bit counter that continuously counts number for m_clk cycles elapsed after EMIF is brought out of reset. |

### 7.3.5.23 READ\_IDLE\_CTRL Register (offset = 98h) [reset = 50000h]

READ\_IDLE\_CTRL is shown in [Figure 7-113](#) and described in [Table 7-132](#).

**Figure 7-113. READ\_IDLE\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

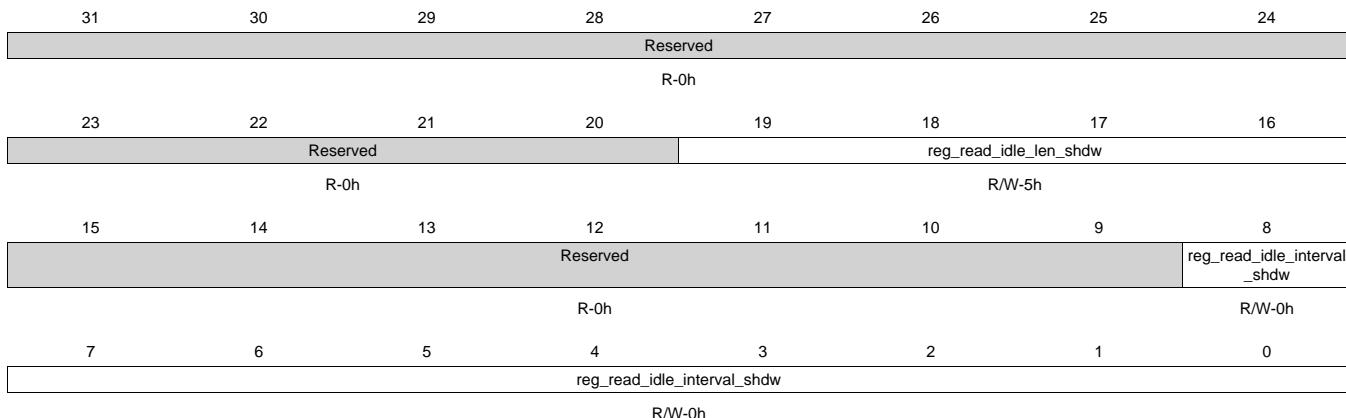
**Table 7-132. READ\_IDLE\_CTRL Register Field Descriptions**

| Bit   | Field                  | Type | Reset | Description   |
|-------|------------------------|------|-------|---|
| 31-20 | Reserved               | R    | 0h    |   |
| 19-16 | reg_read_idle_len      | R/W  | 5h    | The Read Idle Length field determines the minimum size (reg_read_idle_len-1 clock cycles) of Read Idle window for the read idle detection as well as the force read idle time.                    |
| 15-9  | Reserved               | R    | 0h    |   |
| 8-0   | reg_read_idle_interval | R/W  | 0h    | The Read Idle Interval field determines the maximum interval ((reg_read_idle_interval-1)*64 clock cycles) between read idle detections or force. A value of zero disables the read idle function. |

### 7.3.5.24 READ\_IDLE\_CTRL\_SHDW Register (offset = 9Ch) [reset = 50000h]

READ\_IDLE\_CTRL\_SHDW is shown in [Figure 7-114](#) and described in [Table 7-133](#).

**Figure 7-114. READ\_IDLE\_CTRL\_SHDW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

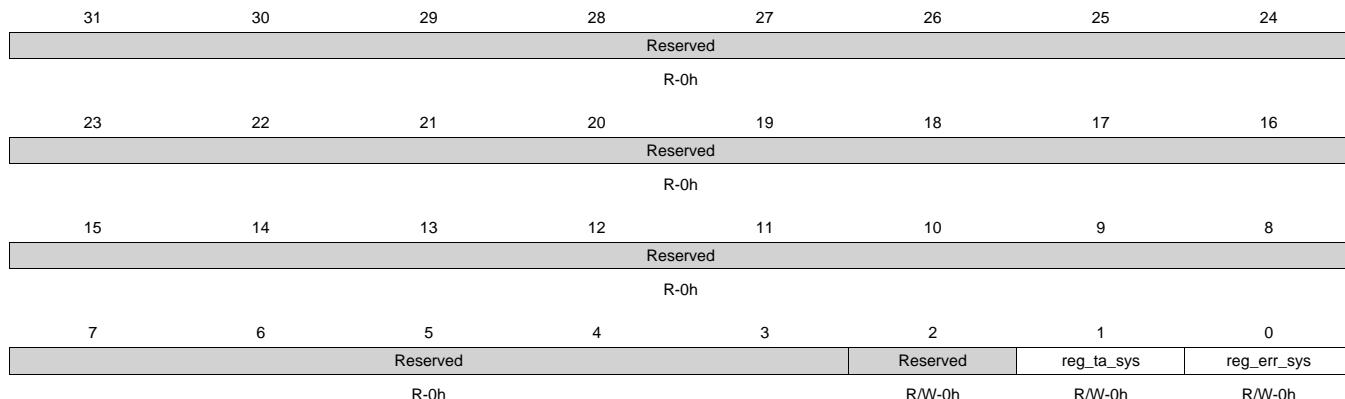
**Table 7-133. READ\_IDLE\_CTRL\_SHDW Register Field Descriptions**

| Bit   | Field                       | Type | Reset | Description   |
|-------|-----------------------------|------|-------|---|
| 31-20 | Reserved                    | R    | 0h    |   |
| 19-16 | reg_read_idle_len_shdw      | R/W  | 5h    | Shadow field for reg_read_idle_len.<br>This field is loaded into reg_read_idle_len field in Read Idle Control register when SIdleAck is asserted            |
| 15-9  | Reserved                    | R    | 0h    |   |
| 8-0   | reg_read_idle_interval_shdw | R/W  | 0h    | Shadow field for reg_read_idle_interval.<br>This field is loaded into reg_read_idle_interval field in Read Idle Control register when SIdleAck is asserted. |

### 7.3.5.25 IRQSTATUS\_RAW\_SYS Register (offset = A4h) [reset = 0h]

IRQSTATUS\_RAW\_SYS is shown in [Figure 7-115](#) and described in [Table 7-134](#).

**Figure 7-115. IRQSTATUS\_RAW\_SYS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

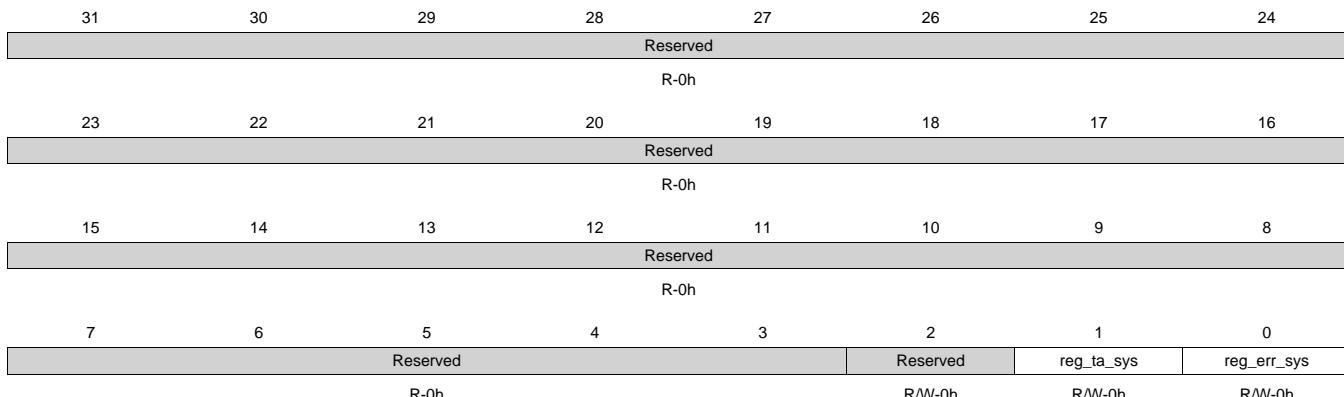
**Table 7-134. IRQSTATUS\_RAW\_SYS Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description   |
|------|-------------|------|-------|---|
| 31-3 | Reserved    | R    | 0h    |   |
| 2    | Reserved    | R/W  | 0h    | Reserved.   |
| 1    | reg_ta_sys  | R/W  | 0h    | Raw status of system OCP interrupt.<br>Write 1 to set the (raw) status, mostly for debug.<br>Writing a 0 has no effect. |
| 0    | reg_err_sys | R/W  | 0h    | Raw status of system OCP interrupt.<br>Write 1 to set the (raw) status, mostly for debug.<br>Writing a 0 has no effect. |

### 7.3.5.26 IRQSTATUS\_SYS Register (offset = ACh) [reset = 0h]

IRQSTATUS\_SYS is shown in [Figure 7-116](#) and described in [Table 7-135](#).

**Figure 7-116. IRQSTATUS\_SYS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

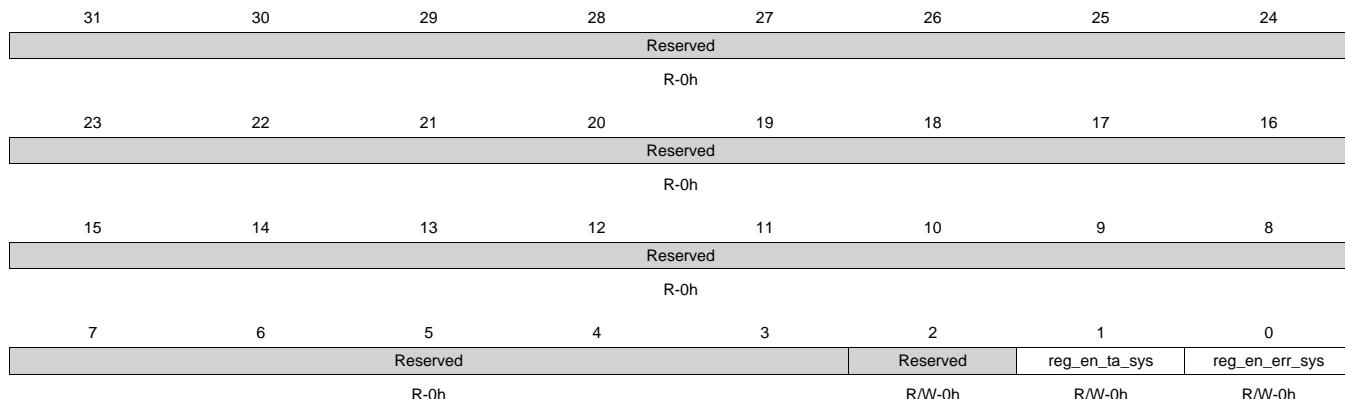
**Table 7-135. IRQSTATUS\_SYS Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description   |
|------|-------------|------|-------|---|
| 31-3 | Reserved    | R    | 0h    |   |
| 2    | Reserved    | R/W  | 0h    | Reserved.   |
| 1    | reg_ta_sys  | R/W  | 0h    | Enabled status of system OCP interrupt.<br>Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, i.e. even if not enabled).<br>Writing a 0 has no effect. |
| 0    | reg_err_sys | R/W  | 0h    | Enabled status of system OCP interrupt.<br>Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, i.e. even if not enabled).<br>Writing a 0 has no effect. |

### 7.3.5.27 IRQENABLE\_SET\_SYS Register (offset = B4h) [reset = 0h]

IRQENABLE\_SET\_SYS is shown in [Figure 7-117](#) and described in [Table 7-136](#).

**Figure 7-117. IRQENABLE\_SET\_SYS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

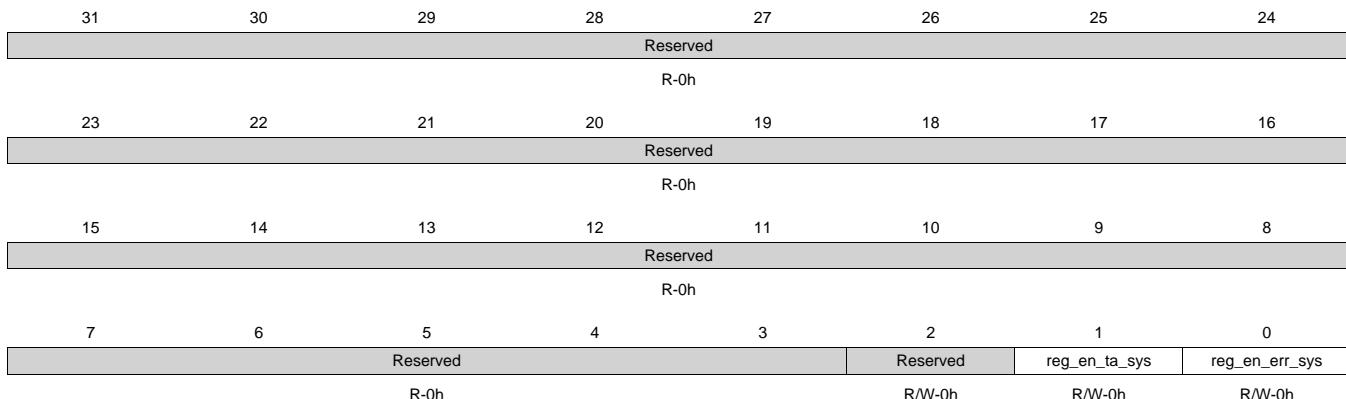
**Table 7-136. IRQENABLE\_SET\_SYS Register Field Descriptions**

| Bit  | Field                       | Type | Reset | Description   |
|------|-----------------------------|------|-------|---|
| 31-3 | Reserved                    | R    | 0h    |   |
| 2    | Reserved                    | R/W  | 0h    | Reserved.   |
| 1    | <code>reg_en_ta_sys</code>  | R/W  | 0h    | Enable set for system OCP interrupt.<br>Writing a 1 will enable the interrupt, and set this bit as well as the corresponding Interrupt Enable Clear Register.<br>Writing a 0 has no effect. |
| 0    | <code>reg_en_err_sys</code> | R/W  | 0h    | Enable set for system OCP interrupt.<br>Writing a 1 will enable the interrupt, and set this bit as well as the corresponding Interrupt Enable Clear Register.<br>Writing a 0 has no effect. |

### 7.3.5.28 IRQENABLE\_CLR\_SYS Register (offset = BCh) [reset = 0h]

IRQENABLE\_CLR\_SYS is shown in [Figure 7-118](#) and described in [Table 7-137](#).

**Figure 7-118. IRQENABLE\_CLR\_SYS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-137. IRQENABLE\_CLR\_SYS Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description  |
|------|----------------|------|-------|--|
| 31-3 | Reserved       | R    | 0h    |  |
| 2    | Reserved       | R/W  | 0h    | Reserved.  |
| 1    | reg_en_ta_sys  | R/W  | 0h    | Enable clear for system OCP interrupt.<br>Writing a 1 will disable the interrupt, and clear this bit as well as the corresponding Interrupt Enable Set Register.<br>Writing a 0 has no effect. |
| 0    | reg_en_err_sys | R/W  | 0h    | Enable clear for system OCP interrupt.<br>Writing a 1 will disable the interrupt, and clear this bit as well as the corresponding Interrupt Enable Set Register.<br>Writing a 0 has no effect. |

### 7.3.5.29 ZQ\_CONFIG Register (offset = C8h) [reset = 0h]

ZQ\_CONFIG is shown in [Figure 7-119](#) and described in [Table 7-138](#).

**Figure 7-119. ZQ\_CONFIG Register**

| 31           | 30           | 29                 | 28              | 27                 | 26   | 25               | 24 |
|--------------|--------------|--------------------|-----------------|--------------------|------|------------------|----|
| reg_zq_cs1en | reg_zq_cs0en | reg_zq_dualcalen   | reg_zq_sfexiten | Reserved           |      |                  |    |
| R/W-0h       | R/W-0h       | R/W-0h             | R/W-0h          |                    | R-0h |                  |    |
| 23           | 22           | 21                 | 20              | 19                 | 18   | 17               | 16 |
|              | Reserved     |                    |                 | reg_zq_zqinit_mult |      | reg_zq_zqcl_mult |    |
|              |              | R-0h               |                 | R/W-0h             |      | R/W-0h           |    |
| 15           | 14           | 13                 | 12              | 11                 | 10   | 9                | 8  |
|              |              | reg_zq_refinterval |                 | R/W-0h             |      | R/W-0h           |    |
| 7            | 6            | 5                  | 4               | 3                  | 2    | 1                | 0  |
|              |              | reg_zq_refinterval |                 | R/W-0h             |      | R/W-0h           |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-138. ZQ\_CONFIG Register Field Descriptions**

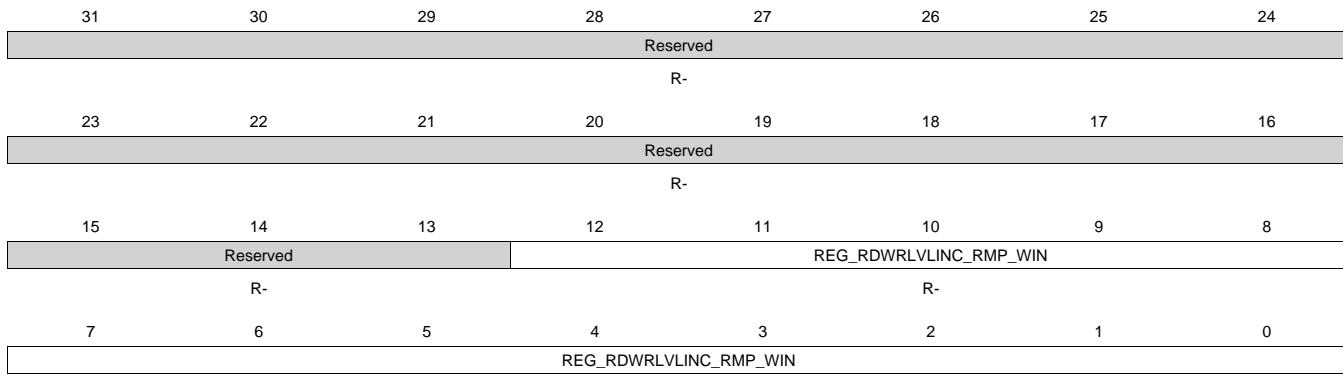
| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31    | reg_zq_cs1en       | R/W  | 0h    | Writing a 1 enables ZQ calibration for CS1.   |
| 30    | reg_zq_cs0en       | R/W  | 0h    | Writing a 1 enables ZQ calibration for CS0.   |
| 29    | reg_zq_dualcalen   | R/W  | 0h    | ZQ Dual Calibration enable.<br>Allows both ranks to be ZQ calibrated simultaneously.<br>Setting this bit requires both chip selects to have a seerate calibration resistor per device.  |
| 28    | reg_zq_sfexiten    | R/W  | 0h    | ZQCL on Self Refresh, Active Power-Down, and Precharge Power-Down exit enable.<br>Writing a 1 enables the issuing of ZQCL on Self-Refresh, Active Power-Down, and Precharge Power-Down exit.  |
| 27-20 | Reserved           | R    | 0h    |   |
| 19-18 | reg_zq_zqinit_mult | R/W  | 0h    | Indicates the number of ZQCL intervals that make up a ZQINIT interval, minus one.   |
| 17-16 | reg_zq_zqcl_mult   | R/W  | 0h    | Indicates the number of ZQCS intervals that make up a ZQCL interval, minus one.<br>ZQCS interval is defined by reg_zq_zqcs in SDRAM Timing 3 Register.  |
| 15-0  | reg_zq_refinterval | R/W  | 0h    | Number of refresh periods between ZQCS commands.<br>This field supports between one refresh period to 256 ms between ZQCS calibration commands.<br>Refresh period is defined by reg_refresh_rate in SDRAM Refresh Control register. |

### 7.3.5.30 RDWR\_LVL\_RMP\_WIN Register (offset = D4h) [reset = 0h]

Read-Write Leveling Ramp Window Register

Read-Write Leveling Ramp Window Register is shown in Figure 7-120 and described in Table 7-139.

**Figure 7-120. Read-Write Leveling Ramp Window Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-139. Read-Write Leveling Ramp Window Register Field Descriptions**

| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-13 | Reserved                | R    |       | Reserved.   |
| 12-0  | REG_RDWRLVLINC_RM_P_WIN | R    |       | Incremental leveling ramp window in number of refresh periods. The value programmed is minus one the required value. Refresh period is defined by reg_refresh_rate in SDRAM Refresh Control register. |

### 7.3.5.31 RDWR\_LVL\_RMP\_CTRL Register (offset = D8h) [reset = 0h]

Read-Write Leveling Ramp Control Register

Read-Write Leveling Ramp Control Register is shown in [Figure 7-121](#) and described in [Table 7-140](#).

**Figure 7-121. Read-Write Leveling Ramp Control Register**

| 31                       | 30                     | 29 | 28 | 27 | 26 | 25 | 24   |
|--------------------------|------------------------|----|----|----|----|----|------|
| REG_RDWRLVL_EN           | REG_RDWRLVLINC_RMP_PRE |    |    |    |    |    |      |
| R/W-                     |                        |    |    |    |    |    |      |
| 23                       | 22                     | 21 | 20 | 19 | 18 | 17 | 16   |
| REG_RDLVLINC_RMP_INT     |                        |    |    |    |    |    | R/W- |
| 15                       | 14                     | 13 | 12 | 11 | 10 | 9  | 8    |
| REG_RDLVLGATEINC_RMP_INT |                        |    |    |    |    |    | R/W- |
| 7                        | 6                      | 5  | 4  | 3  | 2  | 1  | 0    |
| REG_WRLVLINC_RMP_INT     |                        |    |    |    |    |    |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-140. Read-Write Leveling Ramp Control Register Field Descriptions**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 31    | REG_RDWRLVL_EN           | R/W  |       | Read-Write Leveling enable.<br>Set 1 to enable leveling.<br>Set 0 to disable leveling.   |
| 30-24 | REG_RDWRLVLINC_RM_P_PRE  |      |       | Incremental leveling pre-scalar in number of refresh periods during ramp window.<br>The value programmed is minus one the required value.<br>Refresh period is defined by reg_refresh_rate in SDRAM Refresh Control register.                |
| 23-16 | REG_RDLVLINC_RMP_INT     | R/W  |       | Incremental read data eye training interval during ramp window.<br>Number of reg_rdwrvlinc_rmp_pre intervals between incremental read data eye training.<br>A value of 0 will disable incremental read data eye training during ramp window. |
| 15-8  | REG_RDLVLGATEINC_RMP_INT | R/W  |       | Incremental read DQS gate training interval during ramp window.<br>Number of reg_rdwrvlinc_rmp_pre intervals between incremental read DQS gate training.<br>A value of 0 will disable incremental read DQS gate training during ramp window. |
| 7-0   | REG_WRLVLINC_RMP_INT     |      |       | Incremental write leveling interval during ramp window.<br>Number of reg_rdwrvlinc_rmp_pre intervals between incremental write leveling.<br>A value of 0 will disable incremental write leveling during ramp window.                         |

### 7.3.5.32 RDWR\_LVL\_CTRL Register (offset = DCh) [reset = 0h]

Read-Write Leveling Control Register

Read-Write Leveling Control Register is shown in [Figure 7-122](#) and described in [Table 7-141](#).

Read-Write Leveling Control Register

**Figure 7-122. Read-Write Leveling Control Register**

| 31                    | 30 | 29 | 28 | 27 | 26                 | 25 | 24 |
|-----------------------|----|----|----|----|--------------------|----|----|
| REG_RDWRLVLFULL_START |    |    |    |    | REG_RDWRLVLINC_PRE |    |    |
| R/W-                  |    |    |    |    |                    |    |    |
| 23                    | 22 | 21 | 20 | 19 | 18                 | 17 | 16 |
| REG_RDLVLINC_INT      |    |    |    |    |                    |    |    |
| R/W-                  |    |    |    |    |                    |    |    |
| 15                    | 14 | 13 | 12 | 11 | 10                 | 9  | 8  |
| REG_RDLVLGATEINC_INT  |    |    |    |    |                    |    |    |
| R/W-                  |    |    |    |    |                    |    |    |
| 7                     | 6  | 5  | 4  | 3  | 2                  | 1  | 0  |
| REG_WRLVLINC_INT      |    |    |    |    |                    |    |    |
| R/W-                  |    |    |    |    |                    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-141. Read-Write Leveling Control Register Field Descriptions**

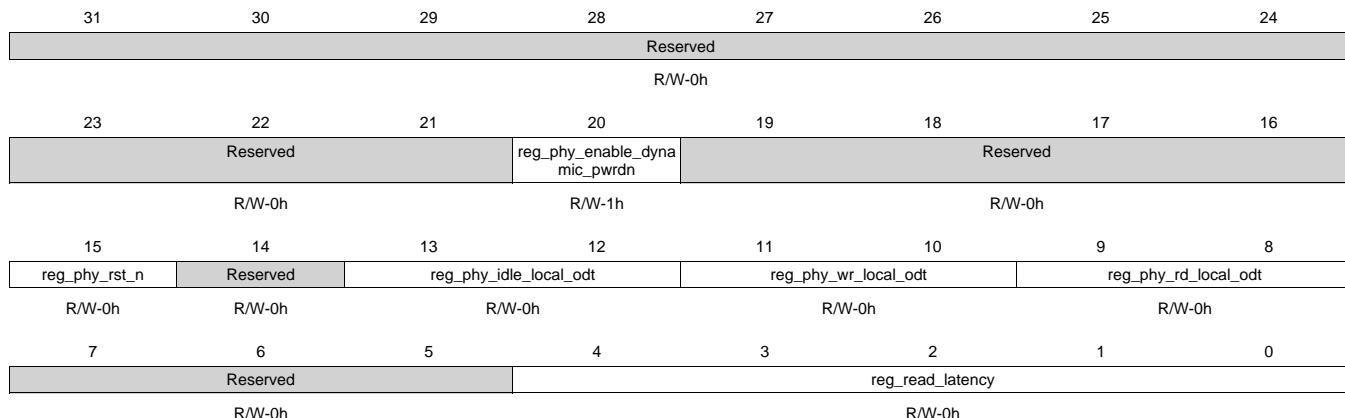
| Bit   | Field                  | Type | Reset | Description  |
|-------|------------------------|------|-------|--|
| 31    | REG_RDWRLVLFULL_S TART | R/W  |       | Full leveling trigger.<br>Writing a 1 to this field triggers full read and write leveling.<br>This bit will self-clear to 0.   |
| 30-24 | REG_RDWRLVLINC_PR E    |      |       | Incremental leveling pre-scalar in number of refresh periods.<br>The value programmed is minus one the required value.<br>Refresh period is defined by reg_refresh_rate in SDRAM Refresh Control register. |
| 23-16 | REG_RDLVLINC_INT       | R/W  |       | Incremental read data eye training interval.<br>Number of reg_rdwrllinc_pre intervals between incremental read data eye training.<br>A value of 0 will disable incremental read data eye training.         |
| 15-8  | REG_RDLVLGATEINC_I NT  | R/W  |       | Incremental read DQS gate training interval.<br>Number of reg_rdwrllinc_pre intervals between incremental read DQS gate training.<br>A value of 0 will disable incremental read DQS gate training.         |
| 7-0   | REG_WRLVLINC_INT       | R/W  |       | Incremental write leveling interval.<br>Number of reg_rdwrllinc_pre intervals between incremental write leveling.<br>A value of 0 will disable incremental write leveling.                                 |

### 7.3.5.33 DDR\_PHY\_CTRL\_1 Register (offset = E4h) [reset = 0h]

DDR\_PHY\_CTRL\_1 is shown in [Figure 7-123](#) and described in [Table 7-142](#).

A write to the DDR PHY Control 1 register must be followed by a write to the SDRAM\_CONFIG register to ensure that the control update/acknowledge protocol is performed on the DID. If CAS latency = 5, the minimum read latency =  $5 + 2 = 7$  and reg\_read\_latency must be programmed as  $7 - 1 = 6$ . The maximum read latency =  $5 + 7 = 12$  and reg\_read\_latency must be programmed as  $12 - 1 = 11$ .

**Figure 7-123. DDR\_PHY\_CTRL\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-142. DDR\_PHY\_CTRL\_1 Register Field Descriptions**

| Bit   | Field                       | Type | Reset | Description  |
|-------|-----------------------------|------|-------|--|
| 31-21 | Reserved                    | R/W  | 0h    | Reserved.  |
| 20    | reg_phy_enable_dynamic_pwrn | R/W  | 1h    | Dynamically enables powering down the IO receiver when not performing a read.<br>0 = IO receivers always powered up.<br>1 = IO receivers only powered up during a read.  |
| 19-16 | Reserved                    | R/W  | 0h    | Reserved.  |
| 15    | reg_phy_rst_n               | R/W  | 0h    | Writing a 1 to this bit will hold the PHY macros in reset.<br>Writing a 0 will bring PHY macros out of reset.  |
| 14    | Reserved                    | R/W  | 0h    | Reserved.  |
| 13-12 | reg_phy_idle_local_odt      | R/W  | 0h    | Value to drive on the 2-bit local_odt PHY outputs when reg_phy_dynamic_pwrn_enable is asserted and a read is not in progress and reg_phy_dynamic_pwrn_enable. Typically this is the value required to disable termination (00) to save power when idle.  |
| 11-10 | reg_phy_wr_local_odt        | R/W  | 0h    | This bit controls the value assigned to the reg_phy_wr_local_odt input on the data macros.<br>Always set to 00.  |
| 9-8   | reg_phy_rd_local_odt        | R/W  | 0h    | Value to drive on the 2-bit local_odt PHY outputs when output enable is not asserted and a read is in progress (where in progress is defined as after a read command is issued and until all read data has been returned all the way to the controller.) Typically this is set to the value required to enable termination at the desired strength for read usage.<br>00 = ODT off.<br>01 = ODT off.<br>10 = Full thevenin load.<br>11 = Half thevenin load. |
| 7-5   | Reserved                    | R/W  | 0h    | Reserved.  |

**Table 7-142. DDR\_PHY\_CTRL\_1 Register Field Descriptions (continued)**

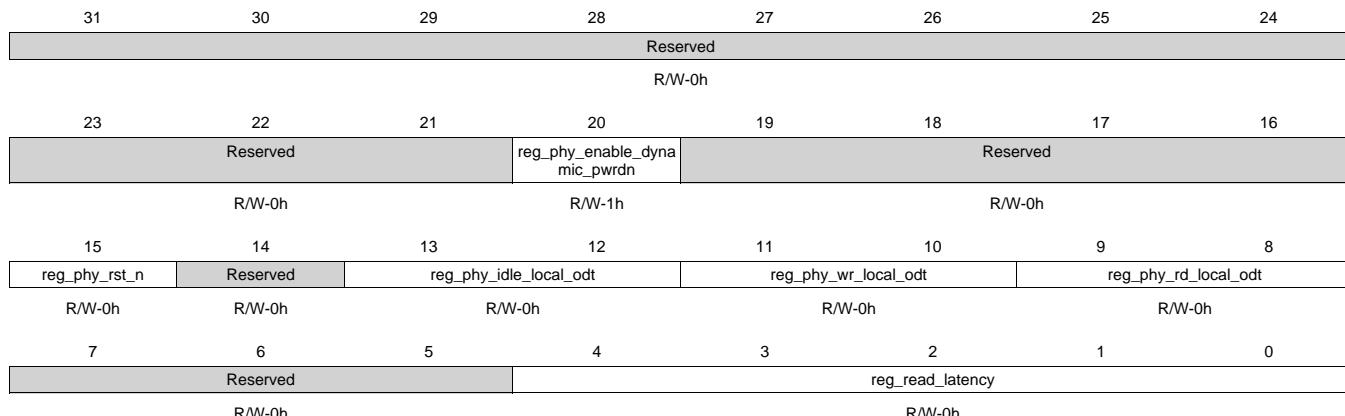
| Bit | Field            | Type | Reset | Description   |
|-----|------------------|------|-------|---|
| 4-0 | reg_read_latency | R/W  | 0h    | This field defines the latency for read data from DDR SDRAM in number of DDR clock cycles.<br>The value applied should be equal to the required value minus one.<br>The maximum read latency supported by the DDR PHY is equal to CAS latency plus 7 clock cycles.<br>The minimum read latency must be equal to CAS latency plus 2 clock cycle. |

### 7.3.5.34 DDR\_PHY\_CTRL\_1\_SHDW Register (offset = E8h) [reset = 0h]

DDR\_PHY\_CTRL\_1\_SHDW is shown in [Figure 7-124](#) and described in [Table 7-143](#).

A write to the DDR PHY Control 1 register must be followed by a write to the SDRAM\_CONFIG register to ensure that the control update/acknowledge protocol is performed on the DID. If CAS latency = 5, the minimum read latency =  $5 + 2 = 7$  and reg\_read\_latency must be programmed as  $7 - 1 = 6$ . The maximum read latency =  $5 + 7 = 12$  and reg\_read\_latency must be programmed as  $12 - 1 = 11$ .

**Figure 7-124. DDR\_PHY\_CTRL\_1\_SHDW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-143. DDR\_PHY\_CTRL\_1\_SHDW Register Field Descriptions**

| Bit   | Field                        | Type | Reset | Description  |
|-------|------------------------------|------|-------|--|
| 31-21 | Reserved                     | R/W  | 0h    | Reserved.  |
| 20    | reg_phy_enable_dynamic_pwrnd | R/W  | 1h    | Dynamically enables powering down the IO receiver when not performing a read.<br>0 = IO receivers always powered up.<br>1 = IO receivers only powered up during a read.  |
| 19-16 | Reserved                     | R/W  | 0h    | Reserved.  |
| 15    | reg_phy_rst_n                | R/W  | 0h    | Writing a 1 to this bit will hold the PHY macros in reset.<br>Writing a 0 will bring PHY macros out of reset.  |
| 14    | Reserved                     | R/W  | 0h    | Reserved.  |
| 13-12 | reg_phy_idle_local_odt       | R/W  | 0h    | Value to drive on the 2-bit local_odt PHY outputs when reg_phy_dynamic_pwrnd_enable is asserted and a read is not in progress and reg_phy_dynamic_pwrnd_enable. Typically this is the value required to disable termination (00) to save power when idle.  |
| 11-10 | reg_phy_wr_local_odt         | R/W  | 0h    | This bit controls the value assigned to the reg_phy_wr_local_odt input on the data macros.<br>Always set to 00.  |
| 9-8   | reg_phy_rd_local_odt         | R/W  | 0h    | Value to drive on the 2-bit local_odt PHY outputs when output enable is not asserted and a read is in progress (where in progress is defined as after a read command is issued and until all read data has been returned all the way to the controller.) Typically this is set to the value required to enable termination at the desired strength for read usage.<br>00 = ODT off.<br>01 = ODT off.<br>10 = Full thevenin load.<br>11 = Half thevenin load. |
| 7-5   | Reserved                     | R/W  | 0h    | Reserved.  |

**Table 7-143. DDR\_PHY\_CTRL\_1\_SHDW Register Field Descriptions (continued)**

| Bit | Field            | Type | Reset | Description   |
|-----|------------------|------|-------|---|
| 4-0 | reg_read_latency | R/W  | 0h    | This field defines the latency for read data from DDR SDRAM in number of DDR clock cycles.<br>The value applied should be equal to the required value minus one.<br>The maximum read latency supported by the DDR PHY is equal to CAS latency plus 7 clock cycles.<br>The minimum read latency must be equal to CAS latency plus 2 clock cycle. |

### 7.3.5.35 PRI\_COS\_MAP Register (offset = 100h) [reset = 0h]

Priority to Class of Service Mapping Register

Priority to Class of Service Mapping Register is shown in [Figure 7-125](#) and described in [Table 7-144](#).

**Figure 7-125. Priority to Class of Service Mapping Register**

|                    |    |               |    |               |    |               |    |
|--------------------|----|---------------|----|---------------|----|---------------|----|
| 31                 | 30 | 29            | 28 | 27            | 26 | 25            | 24 |
| REG_PRI_COS_MAP_EN |    |               |    | Reserved      |    |               |    |
| R/W-               |    |               |    | R-            |    |               |    |
| 23                 | 22 | 21            | 20 | 19            | 18 | 17            | 16 |
|                    |    |               |    | Reserved      |    |               |    |
|                    |    |               |    | R-            |    |               |    |
| 15                 | 14 | 13            | 12 | 11            | 10 | 9             | 8  |
| REG_PRI_7_COS      |    | REG_PRI_6_COS |    | REG_PRI_5_COS |    | REG_PRI_4_COS |    |
| R/W-               |    | R/W-          |    | R/W-          |    | R/W-          |    |
| 7                  | 6  | 5             | 4  | 3             | 2  | 1             | 0  |
| REG_PRI_3_COS      |    | REG_PRI_2_COS |    | REG_PRI_1_COS |    | REG_PRI_0_COS |    |
| R/W-               |    | R/W-          |    | R/W-          |    | R/W-          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-144. Priority to Class of Service Mapping Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31    | REG_PRI_COS_MAP_EN | R/W  |       | Set 1 to enable priority to class of service mapping.<br>Set 0 to disable mapping.   |
| 30-16 | Reserved           | R    |       | Reserved.  |
| 15-14 | REG_PRI_7_COS      | R/W  |       | Class of service for commands with priority of 7.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 13-12 | REG_PRI_6_COS      | R/W  |       | Class of service for commands with priority of 6.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 11-10 | REG_PRI_5_COS      | R/W  |       | Class of service for commands with priority of 5.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 9-8   | REG_PRI_4_COS      | R/W  |       | Class of service for commands with priority of 4.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 7-6   | REG_PRI_3_COS      | R/W  |       | Class of service for commands with priority of 3.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 5-4   | REG_PRI_2_COS      | R/W  |       | Class of service for commands with priority of 2.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 3-2   | REG_PRI_1_COS      | R/W  |       | Class of service for commands with priority of 1.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |
| 1-0   | REG_PRI_0_COS      | R/W  |       | Class of service for commands with priority of 0.<br>Value can be 1 or 2.<br>Setting a value of 0 or 3 will not assign any class of service. |

### 7.3.5.36 CONNID\_COS\_1\_MAP Register (offset = 104h) [reset = 0h]

Connection ID to Class of Service 1 Mapping Register

Connection ID to Class of Service 1 Mapping Register is shown in Figure 7-126 and described in Table 7-145.

**Figure 7-126. Connection ID to Class of Service 1 Mapping Register**

| 31                      | 30                 | 29 | 28 | 27                 | 26                 | 25   | 24 |
|-------------------------|--------------------|----|----|--------------------|--------------------|------|----|
| REG_CONNID_COS_1_MAP_EN | REG_CONNID_1_COS_1 |    |    |                    |                    |      |    |
| R/W-                    | R/W-               |    |    |                    |                    |      |    |
| 23                      | 22                 | 21 | 20 | 19                 | 18                 | 17   | 16 |
| REG_CONNID_1_CO_S_1     | REG_MSK_1_COS_1    |    |    | REG_CONNID_2_COS_1 |                    |      |    |
| R/W-                    | R/W-               |    |    | R/W-               |                    |      |    |
| 15                      | 14                 | 13 | 12 | 11                 | 10                 | 9    | 8  |
| REG_CONNID_2_COS_1      |                    |    |    | REG_MSK_2_COS_1    | REG_CONNID_3_COS_1 |      |    |
| R/W-                    | R/W-               |    |    | R/W-               | R/W-               |      |    |
| 7                       | 6                  | 5  | 4  | 3                  | 2                  | 1    | 0  |
| REG_CONNID_3_COS_1      |                    |    |    |                    | REG_MSK_3_COS_1    |      |    |
| R/W-                    | R/W-               |    |    |                    |                    | R/W- |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-145. Connection ID to Class of Service 1 Mapping Register Field Descriptions**

| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31    | REG_CONNID_COS_1_MAP_EN | R/W  |       | Set 1 to enable connection ID to class of service 1 mapping.<br>Set 0 to disable mapping.   |
| 30-23 | REG_CONNID_1_COS_1      | R/W  |       | Connection ID value 1 for class of service 1.   |
| 22-20 | REG_MSK_1_COS_1         | R/W  |       | Mask for connection ID value 1 for class of service 1.<br>0 = disable masking.<br>1 = mask connection ID bit 0.<br>2 = mask connection ID bits 1:0.<br>3 = mask connection ID bits 2:0.<br>4 = mask connection ID bits 3:0.<br>5 = mask connection ID bits 4:0.<br>6 = mask connection ID bits 5:0.<br>7 = mask connection ID bits 6:0. |
| 19-12 | REG_CONNID_2_COS_1      | R/W  |       | Connection ID value 2 for class of service 1.   |
| 11-10 | REG_MSK_2_COS_1         | R/W  |       | Mask for connection ID value 2 for class of service 1.<br>0 = disable masking.<br>1 = mask connection ID bit 0.<br>2 = mask connection ID bits 1:0.<br>3 = mask connection ID bits 2:0.   |
| 9-2   | REG_CONNID_3_COS_1      | R/W  |       | Connection ID value 3 for class of service 1.   |
| 1-0   | REG_MSK_3_COS_1         | R/W  |       | Mask for connection ID.<br>Value 3 for class of service 1.<br>0 = disable masking.<br>1 = mask connection ID bit 0.<br>2 = mask connection ID bits 1:0.<br>3 = mask connection ID bits 2:0.   |

### 7.3.5.37 CONNID\_COS\_2\_MAP Register (offset = 108h) [reset = 0h]

Connection ID to Class of Service 2 Mapping Register

Connection ID to Class of Service 2 Mapping Register is shown in Figure 7-127 and described in Table 7-146.

**Figure 7-127. Connection ID to Class of Service 2 Mapping Register**

| 31                      | 30                 | 29 | 28 | 27                 | 26              | 25                 | 24 |
|-------------------------|--------------------|----|----|--------------------|-----------------|--------------------|----|
| REG_CONNID_COS_2_MAP_EN | REG_CONNID_1_COS_2 |    |    |                    |                 |                    |    |
| R/W-                    | R/W-               |    |    |                    |                 |                    |    |
| 23                      | 22                 | 21 | 20 | 19                 | 18              | 17                 | 16 |
| REG_CONNID_1_CO_S_2     | REG_MSK_1_COS_2    |    |    | REG_CONNID_2_COS_2 |                 |                    |    |
| R/W-                    | R/W-               |    |    | R/W-               |                 |                    |    |
| 15                      | 14                 | 13 | 12 | 11                 | 10              | 9                  | 8  |
| REG_CONNID_2_COS_2      |                    |    |    | REG_MSK_2_COS_2    |                 | REG_CONNID_3_COS_2 |    |
| R/W-                    | R/W-               |    |    | R/W-               | R/W-            |                    |    |
| 7                       | 6                  | 5  | 4  | 3                  | 2               | 1                  | 0  |
| REG_CONNID_3_COS_2      |                    |    |    |                    | REG_MSK_3_COS_2 |                    |    |
| R/W-                    | R/W-               |    |    | R/W-               | R/W-            |                    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-146. Connection ID to Class of Service 2 Mapping Register Field Descriptions**

| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31    | REG_CONNID_COS_2_MAP_EN | R/W  |       | Set 1 to enable connection ID to class of service 2 mapping.<br>Set 0 to disable mapping.   |
| 30-23 | REG_CONNID_1_COS_2      | R/W  |       | Connection ID value 1 for class of service 2.   |
| 22-20 | REG_MSK_1_COS_2         | R/W  |       | Mask for connection ID.<br>Value 1 for class of service 2.<br>0 = disable masking.<br>1 = mask connection ID bit 0.<br>2= mask connection ID bits 1:0.<br>3= mask connection ID bits 2:0.<br>4 = mask connection ID bits 3:0.<br>5 = mask connection ID bits 4:0.<br>6 = mask connection ID bits 5:0.<br>7 = mask connection ID bits 6:0. |
| 19-12 | REG_CONNID_2_COS_2      | R/W  |       | Connection ID value 2 for class of service 2.   |
| 11-10 | REG_MSK_2_COS_2         | R/W  |       | Mask for connection ID.<br>Value 2 for class of service 2.<br>0 = disable masking.<br>1 = mask connection ID bit 0.<br>2 = mask connection ID bits 1:0.<br>3 = mask connection ID bits 2:0.   |
| 9-2   | REG_CONNID_3_COS_2      | R/W  |       | Connection ID value 3 for class of service 2.   |

**Table 7-146. Connection ID to Class of Service 2 Mapping Register Field Descriptions (continued)**

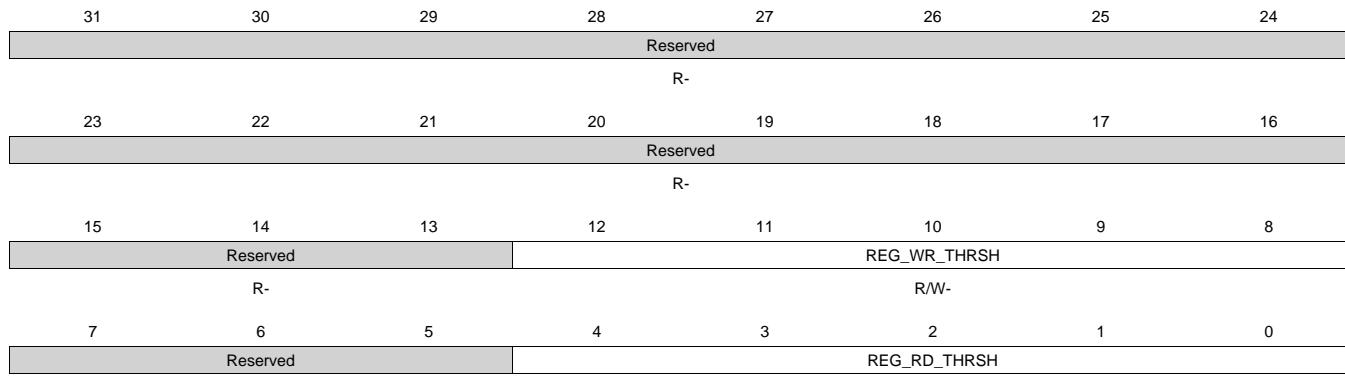
| Bit | Field           | Type | Reset | Description  |
|-----|-----------------|------|-------|--|
| 1-0 | REG_MSK_3_COS_2 | R/W  |       | <p>Mask for connection ID.</p> <p>Value 3 for class of service 2.</p> <p>0 = disable masking.</p> <p>1 = mask connection ID bit 0.</p> <p>2 = mask connection ID bits 1:0.</p> <p>3 = mask connection ID bits 2:0.</p> |

### 7.3.5.38 RD\_WR\_EXEC\_THRSH Register (offset = 120h) [reset = 0h]

Read Write Execution Threshold Register

Read Write Execution Threshold Register is shown in [Figure 7-128](#) and described in [Table 7-147](#).

**Figure 7-128. Read Write Execution Threshold Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 7-147. Read Write Execution Threshold Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description  |
|-------|--------------|------|-------|--|
| 31-13 | Reserved     | R    |       | Reserved.  |
| 12-8  | REG_WR_THRSH | R/W  |       | Write Threshold.<br>Number of SDRAM write bursts after which the EMIF arbitration will switch to executing read commands.<br>The value programmed is always minus one the required number. |
| 7-5   | Reserved     | R    |       | Reserved.  |
| 4-0   | REG_RD_THRSH | R/W  |       | Read Threshold.<br>Number of SDRAM read bursts after which the EMIF arbitration will switch to executing write commands.<br>The value programmed is always minus one the required number.  |

### 7.3.6 DDR2/3/mDDR PHY Registers

**Table 7-148** lists the memory-mapped registers for the DDR2/3 mDDR PHY. Configure the DDR PHY Control Register for calibration of board delay and other parameters to get the SDRAM device working at a different speed. See the device-specific data manual for the base memory address of these registers.

**Table 7-148. Memory-Mapped Registers for DDR2/3/mDDR PHY**

| Register Name                        | Type | Register  | Register   | Address |
|--------------------------------------|------|---|------------|---------|
|                                      |      | Description   | Reset      | Offset  |
| CMD0_REG_PHY_CTRL_SLAVE_RATIO_0      | RW   | DDR PHY Command 0 Address/Command Slave Ratio Register                    | 0x80       | 0x01C   |
| CMD0_REG_PHY_DLL_LOCK_DIFF_0         | RW   | DDR PHY Command 0 Address/Command DLL Lock Difference Register            | 0x4        | 0x028   |
| CMD0_REG_PHY_INVERT_CLKOUT_0         | RW   | DDR PHY Command 0 Invert Clockout Selection Register                      | 0x0        | 0x02C   |
| CMD1_REG_PHY_CTRL_SLAVE_RATIO_0      | RW   | DDR PHY Command 1 Address/Command Slave Ratio Register                    | 0x80       | 0x050   |
| CMD1_REG_PHY_DLL_LOCK_DIFF_0         | RW   | DDR PHY Command 1 Address/Command DLL Lock Difference Register            | 0x4        | 0x05C   |
| CMD1_REG_PHY_INVERT_CLKOUT_0         | RW   | DDR PHY Command 1 Invert Clockout Selection Register                      | 0x0        | 0x060   |
| CMD2_REG_PHY_CTRL_SLAVE_RATIO_0      | RW   | DDR PHY Command 2 Address/Command Slave Ratio Register                    | 0x80       | 0x084   |
| CMD2_REG_PHY_DLL_LOCK_DIFF_0         | RW   | DDR PHY Command 2 Address/Command DLL Lock Difference Register            | 0x4        | 0x090   |
| CMD2_REG_PHY_INVERT_CLKOUT_0         | RW   | DDR PHY Command 2 Invert Clockout Selection Register                      | 0x0        | 0x094   |
| DATA0_REG_PHY_RD_DQS_SLAVE_RATIO_0   | RW   | DDR PHY Data Macro 0 Read DQS Slave Ratio Register                        | 0x04010040 | 0x0C8   |
| DATA0_REG_PHY_WR_DQS_SLAVE_RATIO_0   | RW   | DDR PHY Data Macro 0 Write DQS Slave Ratio Register                       | 0x0        | 0x0DC   |
| DATA0_REG_PHY_WRLVL_INIT_RATIO_0     | RW   | DDR PHY Data Macro 0 Write Leveling Init Ratio Register                   | 0x0        | 0x0F0   |
| DATA0_REG_PHY_WRLVL_INIT_MODE_0      | RW   | DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register    | 0x0        | 0x0F8   |
| DATA0_REG_PHY_GATELEVEL_INIT_RATIO_0 | RW   | DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register                | 0x0        | 0x0FC   |
| DATA0_REG_PHY_GATELEVEL_INIT_MODE_0  | RW   | DDR PHY Data Macro 0 DQS Gate Training Init Mode Ratio Selection Register | 0x0        | 0x104   |
| DATA0_REG_PHY_FIFO_WE_SLAVE_RATIO_0  | RW   | DDR PHY Data Macro 0 DQS Gate Slave Ratio Register                        | 0x0        | 0x108   |

**Table 7-148. Memory-Mapped Registers for DDR2/3/mDDR PHY (continued)**

| Register Name                       | Type | Register  | Register   | Address |
|-------------------------------------|------|---|------------|---------|
| DATA0_REG_PHY_DQ_OFFSET_0           | RW   | Offset value from DQS to DQ for Data Macro 0                              | 0x40       | 0x11C   |
| DATA0_REG_PHY_WR_DATA_SLAVE_RATIO_0 | RW   | DDR PHY Data Macro 0 Write Data Slave Ratio Register                      | 0x04010040 | 0x120   |
| DATA0_REG_PHY_USE_RANK0_DELAYS      | RW   | DDR PHY Data Macro 0 Delay Selection Register                             | 0x0        | 0x134   |
| DATA0_REG_PHY_DLL_LOCK_DIFF_0       | RW   | DDR PHY Data Macro 0 DLL Lock Difference Register                         | 0x4        | 0x138   |
| DATA1_REG_PHY_RD_DQS_SLAVE_RATIO_0  | RW   | DDR PHY Data Macro 1 Read DQS Slave Ratio Register                        | 0x04010040 | 0x16C   |
| DATA1_REG_PHY_WR_DQS_SLAVE_RATIO_0  | RW   | DDR PHY Data Macro 1 Write DQS Slave Ratio Register                       | 0x0        | 0x180   |
| DATA1_REG_PHY_WRLVL_INIT_RATIO_0    | RW   | DDR PHY Data Macro 1 Write Leveling Init Ratio Register                   | 0x0        | 0x194   |
| DATA1_REG_PHY_WRLVL_INIT_MODE_0     | RW   | DDR PHY Data Macro 1 Write Leveling Init Mode Ratio Selection Register    | 0x0        | 0x19C   |
| DATA1_REG_PHY_GATELVL_INIT_RATIO_0  | RW   | DDR PHY Data Macro 1 DQS Gate Training Init Ratio Register                | 0x0        | 0x1A0   |
| DATA1_REG_PHY_GATELVL_INIT_MODE_0   | RW   | DDR PHY Data Macro 1 DQS Gate Training Init Mode Ratio Selection Register | 0x0        | 0x1A8   |
| DATA1_REG_PHY_FIFO_WE_SLAVE_RATIO_0 | RW   | DDR PHY Data Macro 1 DQS Gate Slave Ratio Register                        | 0x0        | 0x1AC   |
| DATA1_REG_PHY_DQ_OFFSET_1           | RW   | Offset value from DQS to DQ for Data Macro 1                              | 0x40       | 0x1C0   |
| DATA1_REG_PHY_WR_DATA_SLAVE_RATIO_0 | RW   | DDR PHY Data Macro 1 Write Data Slave Ratio Register                      | 0x04010040 | 0x1C4   |
| DATA1_REG_PHY_USE_RANK0_DELAYS      | RW   | DDR PHY Data Macro 1 Delay Selection Register                             | 0x0        | 0x1D8   |
| DATA1_REG_PHY_DLL_LOCK_DIFF_0       | RW   | DDR PHY Data Macro 1 DLL Lock Difference Register                         | 0x4        | 0x1DC   |

### 7.3.6.1 DDR PHY Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2\_REG\_PHY\_CTRL\_SLAVE\_RATIO\_0)

The DDR PHY Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2\_REG\_PHY\_CTRL\_SLAVE\_RATIO\_0) is shown in the figure and table below.

**Figure 7-129. DDR PHY Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2\_REG\_PHY\_CTRL\_SLAVE\_RATIO\_0)**

|    |          |   |                 |    |
|----|----------|---|-----------------|----|
| 31 | Reserved |   |                 | 16 |
|    | R-0      |   |                 |    |
| 15 | 10       | 9 | CMD_SLAVE_RATIO | 0  |
|    | Reserved |   | R/W-80h         |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-149. DDR PHY Command 0/1/2 Address/Command Slave Ratio Register (CMD0/1/2\_REG\_PHY\_CTRL\_SLAVE\_RATIO\_0) Field Descriptions**

| Bit   | Field           | Value | Description   |
|-------|-----------------|-------|---|
| 31-10 | Reserved        |       | Reserved  |
| 9-0   | CMD_SLAVE_RATIO | 0-80h | Ratio value for address/command launch timing in DDR PHY macro. This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line. |

### 7.3.6.2 DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register (CMD0/1/2\_REG\_PHY\_DLL\_LOCK\_DIFF\_0)

The DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register (CMD0/1/2\_REG\_PHY\_DLL\_LOCK\_DIFF\_0) is shown in the figure and table below.

**Figure 7-130. DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register (CMD0/1/2\_REG\_PHY\_DLL\_LOCK\_DIFF\_0)**

|    |          |   |               |    |
|----|----------|---|---------------|----|
| 31 | Reserved |   |               | 16 |
|    | R-0      |   |               |    |
| 15 | 4        | 3 | DLL_LOCK_DIFF | 0  |
|    | Reserved |   | R/W-4h        |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-150. DDR PHY Command 0/1/2 Address/Command DLL Lock Difference Register (CMD0/1/2\_REG\_PHY\_DLL\_LOCK\_DIFF\_0) Field Descriptions**

| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 31-4 | Reserved      |       | Reserved   |
| 3-0  | DLL_LOCK_DIFF | 0-4h  | The max number of delay line taps variation allowed while maintaining the master DLL lock. This is calculated as total jitter/ delay line tap size, where total jitter is half of (incoming clock jitter (pp) + delay line jitter (pp)). |

### 7.3.6.3 DDR PHY Command 0/1/2 Invert Clockout Selection Register( CMD0/1/2\_REG\_PHY\_INVERT\_CLKOUT\_0)

The CDDR PHY Command 0/1/2 Invert Clockout Selection Register(CMD0/1/2\_REG\_PHY\_INVERT\_CLKOUT\_0) is shown in the figure and table below.

**Figure 7-131. DDR PHY Command 0/1/2 Invert Clockout Selection Register(  
CMD0/1/2\_REG\_PHY\_INVERT\_CLKOUT\_0)**

|    |          |  |       |
|----|----------|--|-------|
| 31 |          |  | 16    |
|    | Reserved |  |       |
|    | R-0      |  |       |
| 15 |          |  | 1     |
|    | Reserved |  | 0     |
|    | R-0      |  | R/W-0 |
|    |          |  |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-151. DDR PHY Command 0/1/2 Invert Clockout Selection Register(  
CMD0/1/2\_REG\_PHY\_INVERT\_CLKOUT\_0) Field Descriptions**

| Bit  | Field           | Value | Description  |
|------|-----------------|-------|--|
| 31-1 | Reserved        | 0     | Reserved   |
| 0    | INVERT_CLK_SE_L | 0     | Inverts the polarity of DRAM clock.<br>Core clock is passed on to DRAM |
|      |                 | 1     | inverted core clock is passed on to DRAM                               |

### 7.3.6.4 DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register (DATA0/1\_REG\_PHY\_RD\_DQS\_SLAVE\_RATIO\_0)

The DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register(DATA0/1\_REG\_PHY\_RD\_DQS\_SLAVE\_RATIO\_0) is shown in the figure and table below.

**Figure 7-132. DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register  
(DATA0/1\_REG\_PHY\_RD\_DQS\_SLAVE\_RATIO\_0))**

|    |          |          |                        |          |
|----|----------|----------|------------------------|----------|
| 31 | i        |          | 19                     | 16       |
|    |          | Reserved |                        | Reserved |
|    |          | R/W-40h  |                        | R-0h     |
| 15 |          |          | 10                     | 9        |
|    | Reserved |          | RD_DQS_SLAVE_RATIO_CS0 |          |
|    | R-0h     |          | R/W-40h                |          |
|    |          |          |                        |          |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-152. DDR PHY Data Macro 0/1 Read DQS Slave Ratio Register  
(DATA0/1\_REG\_PHY\_RD\_DQS\_SLAVE\_RATIO\_0) Field Descriptions**

| Bit   | Field                  | Value | Description  |
|-------|------------------------|-------|--|
| 31-20 | Reserved               |       | Reserved   |
| 19-10 | Reserved               | 0h    | Reserved   |
| 9-0   | RD_DQS_SLAVE_RATIO_CS0 | 40h   | Ratio value for Read DQS slave DLL for CS0.<br>This is the fraction of a clock cycle represented by the shift to be applied to the read DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line. |

### 7.3.6.5 DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register (DATA0/1\_REG\_PHY\_WR\_DQS\_SLAVE\_RATIO\_0)

The DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register(DATA0/1\_REG\_PHY\_WR\_DQS\_SLAVE\_RATIO\_0) is shown in the figure and table below.

**Table 7-153. DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register (DATA0/1\_REG\_PHY\_WR\_DQS\_SLAVE\_RATIO\_0)**

|    |           |                        |    |        |
|----|-----------|------------------------|----|--------|
| 31 | Reserved  | 20                     | 19 | 16     |
|    | R-0       |                        |    | R-0h   |
| 15 | 10      9 |                        |    | 0      |
|    | Reserved  | WR_DQS_SLAVE_RATIO_CS0 |    | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-154. DDR PHY Data Macro 0/1 Write DQS Slave Ratio Register (DATA0/1\_REG\_PHY\_WR\_DQS\_SLAVE\_RATIO\_0) Field Descriptions**

| Bit   | Field                      | Value | Description  |
|-------|----------------------------|-------|--|
| 31-20 | Reserved                   | 0     | Reserved   |
| 19-10 | Reserved                   |       | Reserved   |
| 9-0   | WR_DQS_SLAVE_R<br>ATIO_CS0 |       | Ratio value for Write DQS slave DLL for CS0.<br><br>This is the fraction of a clock cycle represented by the shift to be applied to the write DQS in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line. |

### 7.3.6.6 DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_RATIO\_0)

The DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_RATIO\_0) is shown in the figure and table below.

**Figure 7-133. DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_RATIO\_0)**

|    |           |                      |    |       |
|----|-----------|----------------------|----|-------|
| 31 | Reserved  | 20                   | 19 | 16    |
|    | R-0       |                      |    | R-0   |
| 15 | 10      9 |                      |    | 0     |
|    | Reserved  | WRLVL_INIT_RATIO_CS0 |    | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

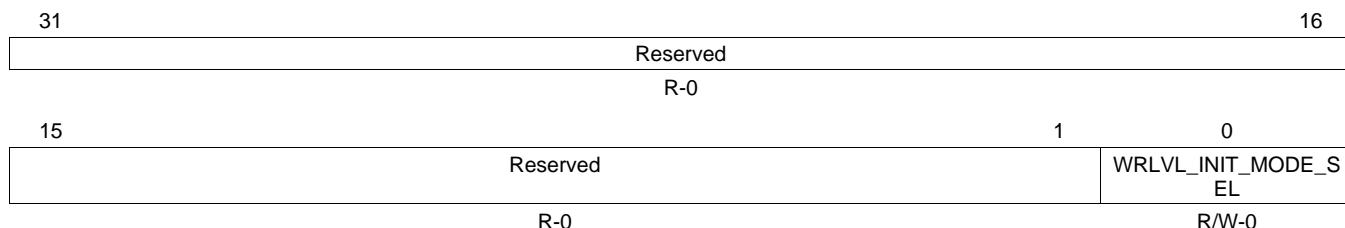
**Table 7-155. DDR PHY Data Macro 0/1 Write Leveling Init Ratio Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_RATIO\_0) Field Descriptions**

| Bit   | Field                    | Value | Description  |
|-------|--------------------------|-------|--|
| 31-20 | Reserved                 | 0     | Reserved   |
| 19-10 | Reserved                 | 0h    | Reserved   |
| 9-0   | WRLVL_INIT_RATIO<br>_CS0 | 0h    | The user programmable init ratio used by Write Leveling FSM when DATA0/1_REG_PHY_WRLVL_INIT_MODE_0 register value set to 1 |

### 7.3.6.7 DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_MODE\_0)

The DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register (DATA0/1\_REG\_PHY\_WRLVL\_INIT\_MODE\_0) is shown in the figure and table below..

**Figure 7-134. DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register  
(DATA0/1\_REG\_PHY\_WRLVL\_INIT\_MODE\_0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

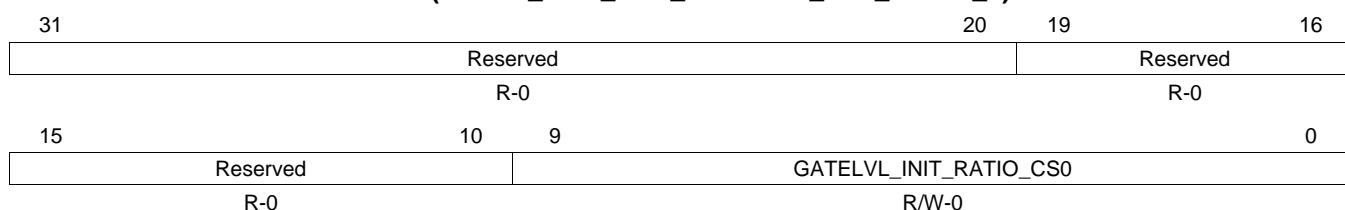
**Table 7-156. DDR PHY Data Macro 0 Write Leveling Init Mode Ratio Selection Register  
(DATA0/1\_REG\_PHY\_WRLVL\_INIT\_MODE\_0)**

| Bit  | Field               | Value  | Description  |
|------|---------------------|--------|--|
| 31-1 | Reserved            | 0      | Reserved   |
| 0    | WRLVL_INIT_MODE_SEL | 0<br>1 | The user programmable init ratio selection mode for Write Leveling FSM.<br>Selects a starting ratio value based on Write Leveling of previous data slice.<br>Selects a starting ratio value based in register DATA0/1_REG_PHY_WRLVL_INIT_RATIO_0 value programmed by the user. |

### 7.3.6.8 DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register (DATA0\_REG\_PHY\_GATELVL\_INIT\_RATIO\_0)

The DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register (DATA0\_REG\_PHY\_GATELVL\_INIT\_RATIO\_0) is shown in the figure and table below.

**Figure 7-135. DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register  
(DATA0\_REG\_PHY\_GATELVL\_INIT\_RATIO\_0)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-157. DDR PHY Data Macro 0 DQS Gate Training Init Ratio Register  
(DATA0\_REG\_PHY\_GATELVL\_INIT\_RATIO\_0) Field Descriptions**

| Bit   | Field                  | Value | Description  |
|-------|------------------------|-------|--|
| 31-20 | Reserved               |       | Reserved   |
| 19-10 | Reserved               | 0h    | Reserved   |
| 9-0   | GATELVL_INIT_RATIO_CS0 | 0h    | The user programmable init ratio used by DQS Gate Training FSM when DATA0/1_REG_PHY_GATELVL_INIT_MODE_0 register value set to 1. |

### 7.3.6.9 DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register (DATA0/1\_REG\_PHY\_GATELVL\_INIT\_MODE\_0)

The DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register(DATA0/1\_REG\_PHY\_GATELVL\_INIT\_MODE\_0) is shown in the figure and table below.

**Figure 7-136. DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register (DATA0/1\_REG\_PHY\_GATELVL\_INIT\_MODE\_0)**

|     |          |                                |    |
|-----|----------|--------------------------------|----|
| 31  | Reserved |                                | 16 |
| R-0 |          |                                |    |
| 15  | Reserved | 1                              | 0  |
| R-0 |          | GATELVL_INIT_MODE_SEL<br>R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-158. DDR PHY Data Macro 0/1 DQS Gate Training Init Mode Ratio Selection Register (DATA0/1\_REG\_PHY\_GATELVL\_INIT\_MODE\_0) Field Descriptions**

| Bit  | Field                 | Value  | Description   |
|------|-----------------------|--------|---|
| 31-1 | Reserved              | 0      | Reserved  |
| 0    | GATELVL_INIT_MODE_SEL | 0<br>1 | User programmable init ratio selection mode for DQS Gate Training FSM.<br>Selects a starting ratio value based on Write Leveling of the same data slice.<br>selects a starting ratio value based on<br>DATA0/1_REG_PHY_GATELVL_INIT_RATIO_0 value programmed by the user. |

### 7.3.6.10 DDR PHY Data Macro 0/1 DQS Gate Slave Ratio Register (DATA0/1\_REG\_PHY\_FIFO\_WE\_SLAVE\_RATIO\_0)

The DDR PHY Data Macro 0/1 DQS Gate Slave Ratio Register (DATA0/1\_REG\_PHY\_FIFO\_WE\_SLAVE\_RATIO\_0) is shown in the figure and table below.

**Figure 7-137. DDR PHY Data Macro 0/1 DQS Gate Slave Ratio Register(DATA0/1\_REG\_PHY\_FIFO\_WE\_SLAVE\_RATIO\_0)**

|          |          |                                  |    |    |
|----------|----------|----------------------------------|----|----|
| 31       | Reserved | 20                               | 19 | 16 |
| R-0      |          |                                  |    |    |
| 15       | 10       | 9                                | 0  |    |
| Reserved |          | FIFO_WE_SLAVE_RATIO_CS0<br>R/W-0 |    |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-159. DDR PHY Data Macro 0/1 DQS Gate Slave Ratio Register (DATA0/1\_REG\_PHY\_FIFO\_WE\_SLAVE\_RATIO\_0) Field Descriptions**

| Bit   | Field                       | Value | Description                      |
|-------|-----------------------------|-------|----------------------------------|
| 31-20 | Reserved                    | 0     | Reserved                         |
| 19-10 | Reserved                    | 0h    | Reserved                         |
| 9-0   | RD_DQS_GATE_SLAVE_RATIO_CS0 | 0h    | Ratio value for fifo we for CS0. |

### 7.3.6.11 DDR PHY Data Macro 0/1 Write Data Slave Ratio Register (DATA0/1\_REG\_PHY\_WR\_DATA\_SLAVE\_RATIO\_0)

The DDR PHY Data Macro 0/1 Write Data Slave Ratio Register (DATA0/1\_REG\_PHY\_WR\_DATA\_SLAVE\_RATIO\_0) is shown in the figure and table below.

**Figure 7-138. DDR PHY Data Macro 0/1 Write Data Slave Ratio Register  
(DATA0/1\_REG\_PHY\_WR\_DATA\_SLAVE\_RATIO\_0)**

| 31       | 20    | 19                      | 16      |
|----------|-------|-------------------------|---------|
| Reserved |       | Reserved                |         |
|          | R-40h |                         | R-0h    |
| 15       | 10    | 9                       | 0       |
| Reserved |       | WR_DATA_SLAVE_RATIO_CS0 |         |
|          | R-0h  |                         | R/W-40h |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

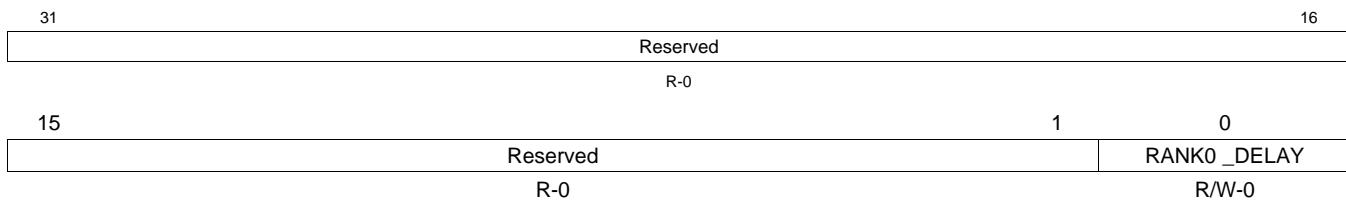
**Table 7-160. DDR PHY Data Macro 0/1 Write Data Slave Ratio Register  
(DATA0/1\_REG\_PHY\_WR\_DATA\_SLAVE\_RATIO\_0) Field Descriptions**

| Bit   | Field                   | Value | Description  |
|-------|-------------------------|-------|--|
| 31-20 | Reserved                | 40h   | Reserved   |
| 19-10 | Reserved                | 0     |  |
| 9-0   | WR_DATA_SLAVE_RATIO_CS0 | 40h   | Ratio value for write data slave DLL for CS0.<br>This is the fraction of a clock cycle represented by the shift to be applied to the write DQ muxes in units of 256ths. In other words, the full-cycle tap value from the master DLL will be scaled by this number over 256 to get the delay value for the slave delay line. |

### 7.3.6.12 DDR PHY Data Macro 0/1 Delay Selection Register (DATA0/1\_REG\_PHY\_USE\_RANK0\_DELAYS)

The DATA0/1\_REG\_PHY\_USE\_RANK0\_DELAYS is shown in [Figure 7-139](#) and described in [Table 7-161](#).

**Figure 7-139. DDR PHY Data Macro 0/1 Delay Selection Register (DATA0/1\_REG\_PHY\_USE\_RANK0\_DELAYS)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-161. DDR PHY Data Macro 0/1 Delay Selection Register (DATA0/1\_REG\_PHY\_USE\_RANK0\_DELAYS) Field Descriptions**

| Bit  | Field                  | Value | Description   |
|------|------------------------|-------|---|
| 31-1 | Reserved               | 0     | Reserved  |
| 0    | PHY_USE_RANK0_DELAYS_0 | 0     | Delay Selection<br>Each Rank uses its own delay. (Recommended). This is applicable only in case of DDR3 |
|      |                        | 1     | Rank 0 delays are used for all ranks. This must be set to 1 for DDR2 and mDDR.                          |

## 7.4 ELM

### 7.4.1 Introduction

Non-managed NAND flash memories can be dense and nonvolatile in their own nature, but error-prone. When reading from NAND flash memories, some level of error-correction is required. In the case of NAND modules with no internal correction capability, sometimes referred to as bare NANDs, the correction process is delegated to the memory controller.

The general-purpose memory controller (GPMC) probes data read from an external NAND flash and uses this to compute checksum-like information, called syndrome polynomials, on a per-block basis. Each syndrome polynomial gives a status of the read operations for a full block, including 512 bytes of data, parity bits, and an optional spare-area data field, with a maximum block size of 1023 bytes. Computation is based on a Bose-Chaudhuri-Hocquenghem (BCH) algorithm. The error-location module (ELM) extracts error addresses from these syndrome polynomials.

Based on the syndrome polynomial value, the ELM can detect errors, compute the number of errors, and give the location of each error bit. The actual data is not required to complete the error-correction algorithm. Errors can be reported anywhere in the NAND flash block, including in the parity bits.

The maximum acceptable number of errors that can be corrected depends on a programmable configuration parameter. 4-, 8-, and 16-bit error-correction levels are supported. The ELM relies on a static and fixed definition of the generator polynomial for each error-correction level that corresponds to the generator polynomials defined in the GPMC (there are three fixed polynomial for the three correction error levels). A larger number of errors than the programmed error-correction level may be detected, but the ELM cannot correct them all. The offending block is then tagged as uncorrectable in the associated computation exit status register. If the computation is successful, that is, if the number of errors detected does not exceed the maximum value authorized for the chosen correction capability, the exit status register contains the information on the number of detected errors.

When the error-location process completes, an interrupt is triggered to inform the central processing unit (CPU) that its status can be checked. The number of detected errors and their locations in the NAND block can be retrieved from the module through register accesses.

#### 7.4.1.1 ELM Features

The ELM has the following features:

- 4, 8, and 16 bits per 512-byte block error-location based on BCH algorithms
- Eight simultaneous processing contexts
- Page-based and continuous modes
- Interrupt generation on error-location process completion:
  - When the full page has been processed in page mode
  - For each syndrome polynomial in continuous mode

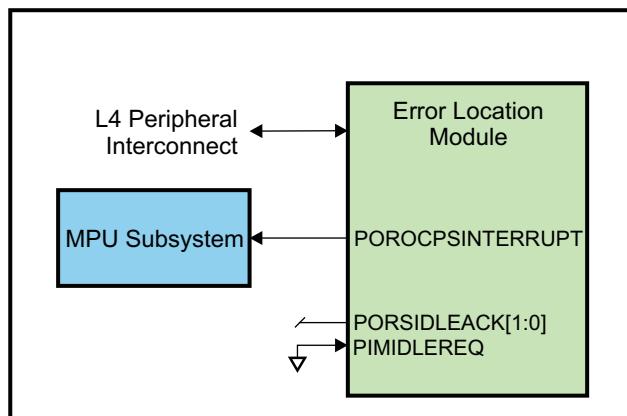
#### 7.4.1.2 Unsupported ELM Features

There are no unsupported ELM features in this device.

## 7.4.2 Integration

The error location module (ELM) is used to extract error addresses from syndrome polynomials generated using a BCH algorithm. Each of these polynomials gives a status of the read operations for a 512 bytes block from a NAND flash and its associated BCH parity bits, plus optionally some spare area information.

The ELM is intended to be used in conjunction with the GPMC. Syndrome polynomials generated on-the-fly when reading a NAND Flash page and stored in GPMC registers are passed to the ELM module. The MPU can then easily correct the data block by flipping the bits pointed to by the ELM error locations outputs.



**Figure 7-140. ELM Integration**

### 7.4.2.1 ELM Connectivity Attributes

The general connectivity for the ELM module in this device is summarized in [Table 7-162](#).

**Table 7-162. ELM Connectivity Attributes**

| Attributes          | Type                         |
|---------------------|------------------------------|
| Power Domain        | Peripheral Domain            |
| Clock Domain        | L4PER_L4LS_GCLK              |
| Reset Signals       | PER_DOM_RST_N                |
| Idle/Wakeup Signals | Smart Idle                   |
| Interrupt Requests  | 1 interrupt to MPU Subsystem |
| DMA Requests        | None                         |
| Physical Address    | L4 Peripheral slave port     |

### 7.4.2.2 ELM Clock and Reset Management

The ELM operates from a single OCP interface clock.

**Table 7-163. ELM Clock Signals**

| Clock Signal               | Max Freq | Reference / Source | Comments                      |
|----------------------------|----------|--------------------|-------------------------------|
| Functional/interface clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>From PRCM |

### 7.4.2.3 ELM Pin List

The ELM module does not include any external interface pins.

### 7.4.3 Functional Description

The ELM is designed around the error-location engine, which handles the computation based on the input syndrome polynomials.

The ELM maps the error-location engine to a standard interconnect interface by using a set of registers to control inputs and outputs.

#### 7.4.3.1 ELM Software Reset

To perform a software reset, write a 1 to the ELM\_SYSCONFIG[1] SOFTRESET bit. The ELM\_SYSSTATUS[0] RESETDONE bit indicates that the software reset is complete when its value is 1. When the software reset completes, the ELM\_SYSCONFIG[1] SOFTRESET bit is automatically reset.

#### 7.4.3.2 ELM Power Management

[Table 7-164](#) describes the power-management features available to the ELM module.

**Table 7-164. Local Power Management Features**

| Feature                | Registers                                 | Description  |
|------------------------|---|--|
| Clock autogating       | ELM_SYSCONFIG[0]<br>AUTOGATING bit        | This bit allows a local power optimization inside the module by gating the ELM_FCLK clock upon the interface activity. |
| Slave idle modes       | ELM_SYSCONFIG[4:3]<br>SIDLEMODE bit field | Force-idle, No-idle, and Smart-idle modes are available.   |
| Clock activity         | ELM_SYSCONFIG[8]<br>CLOCKACTIVITY bit     | The clock can be switched-off or maintained during the wake-up period.   |
| Master Standby modes   | N/A                                       |  |
| Global Wake-up Enable  | N/A                                       |  |
| Wake-up Sources Enable | N/A                                       |  |

**CAUTION**

The PRCM module has no hardware means of reading CLOCKACTIVITY settings. Thus, software must ensure consistent programming between the ELM CLOCKACTIVITY and ELM clock PRCM control bits.

#### 7.4.3.3 ELM Interrupt Requests

[Table 7-165](#) lists the event flags, and their masks, that can cause module interrupts.

**Table 7-165. Events**

| Event Flag                      | Event Mask                          | Map to  | Description  |
|---------------------------------|-------------------------------------|---------|--|
| ELM_IRQSTATUS[8]<br>PAGE_VALID  | ELM_IRQENABLE[8]<br>PAGE_MASK       | ELM_IRQ | Page interrupt                                     |
| ELM_IRQSTATUS[7]<br>LOC_VALID_7 | ELM_IRQENABLE[7]<br>LOCATION_MASK_7 | ELM_IRQ | Error-location interrupt for syndrome polynomial 7 |
| ELM_IRQSTATUS[6]<br>LOC_VALID_6 | ELM_IRQENABLE[6]<br>LOCATION_MASK_6 | ELM_IRQ | Error-location interrupt for syndrome polynomial 6 |
| ELM_IRQSTATUS[5]<br>LOC_VALID_5 | ELM_IRQENABLE[5]<br>LOCATION_MASK_5 | ELM_IRQ | Error-location interrupt for syndrome polynomial 5 |
| ELM_IRQSTATUS[4]<br>LOC_VALID_4 | ELM_IRQENABLE[4]<br>LOCATION_MASK_4 | ELM_IRQ | Error-location interrupt for syndrome polynomial 4 |
| ELM_IRQSTATUS[3]<br>LOC_VALID_3 | ELM_IRQENABLE[3]<br>LOCATION_MASK_3 | ELM_IRQ | Error-location interrupt for syndrome polynomial 3 |
| ELM_IRQSTATUS[2]<br>LOC_VALID_2 | ELM_IRQENABLE[2]<br>LOCATION_MASK_2 | ELM_IRQ | Error-location interrupt for syndrome polynomial 2 |

**Table 7-165. Events (continued)**

| Event Flag                      | Event Mask                          | Map to  | Description  |
|---------------------------------|-------------------------------------|---------|--|
| ELM_IRQSTATUS[1]<br>LOC_VALID_1 | ELM_IRQENABLE[1]<br>LOCATION_MASK_1 | ELM_IRQ | Error-location interrupt for syndrome polynomial 1 |
| ELM_IRQSTATUS[0]<br>LOC_VALID_0 | ELM_IRQENABLE[0]<br>LOCATION_MASK_0 | ELM_IRQ | Error-location interrupt for syndrome polynomial 0 |

#### 7.4.3.4 Processing Initialization

ELM\_LOCATION\_CONFIG global setting parameters must be set before using the error-location engine. The ELM\_LOCATION\_CONFIG[1:0] ECC\_BCH\_LEVEL bit defines the error-correction level used (4-, 8-, or 16-bit error-correction). The ELM\_LOCATION\_CONFIG[26:16] ECC\_SIZE bit field defines the maximum buffer length beyond which the engine processing no longer looks for errors.

The CPU can choose to use the ELM in continuous mode or page mode. If all ELM\_PAGE\_CTRL[i] SECTOR\_i bits are reset (i is the syndrome polynomial number, i = 0 to 7), continuous mode is used. In any other case, page mode is implicitly selected.

- Continuous mode: Each syndrome polynomial is processed independently – results for a syndrome can be retrieved and acknowledged at any time, whatever the status of the other seven processing contexts.
- Page mode: Syndrome polynomials are grouped into atomic entities – only one page can be processed at any given time, even if all eight contexts are not used for this page. Unused contexts are lost and cannot be affected to any other processing. The full page must be acknowledged and cleared before moving to the next page.

For completion interrupts to be generated correctly, all ELM\_IRQENABLE[i] LOCATION\_MASK\_i bits (i = 0 to 7) must be forced to 0 when in page mode, and set to 1 in continuous mode. Additionally, the ELM\_IRQENABLE[8] PAGE\_MASK bit must be set to 1 when in page mode.

The CPU initiates error-location processing by writing a syndrome polynomial into one of the eight possible register sets. Each of these register sets includes seven registers:

ELM\_SYNDROME\_FRAGMENT\_0\_i to ELM\_SYNDROME\_FRAGMENT\_6\_i. The first six registers can be written in any order, but ELM\_SYNDROME\_FRAGMENT\_6\_i must be written last because it includes the validity bit, which instructs the ELM that this syndrome polynomial must be processed (the ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID bit).

As soon as one validity bit is asserted (ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID = 0x1, with i = 0 to 7), error-location processing can start for the corresponding syndrome polynomial. The associated ELM\_LOCATION\_STATUS\_i and ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i registers are not reset (i = 0 to 7). The software must not consider them until the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit is set.

#### 7.4.3.5 Processing Sequence

While the error-location engine is busy processing one syndrome polynomial, further syndrome polynomials can be written. They are processed when the current processing completes.

The engine completes early when:

- No error is detected; that is, when the ELM\_LOCATION\_STATUS\_i[8] ECC\_CORRECTABLE bit is set to 1 and the ELM\_LOCATION\_STATUS\_i[4:0] ECC\_NB\_ERRORS bit field is set to 0x0.
- Too many errors are detected; that is, when the ELM\_LOCATION\_STATUS\_i[8] ECC\_CORRECTABLE bit is set to 0 while the ELM\_LOCATION\_STATUS\_i[4:0] ECC\_NB\_ERRORS bit field is set with the value output by the error-location engine. The reported number of errors is not ensured if ECC\_CORRECTABLE is 0.

If the engine completes early, the associated error-location registers ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i are not updated (i = 0 to 7).

In all other cases, the engine goes through the entire error-location process. Each time an error-location is found, it is logged in the associated ECC\_ERROR\_LOCATION bit field. The first error detected is logged in the ELM\_ERROR\_LOCATION\_0\_i[12:0] ECC\_ERROR\_LOCATION bit field; the second in the ELM\_ERROR\_LOCATION\_1\_i[12:0] ECC\_ERROR\_LOCATION bit field, and so on.

**Table 7-166. ELM\_LOCATION\_STATUS\_i Value Decoding Table**

| ECC_CORRECTABLE Value | ECC_NB_ERRORS Value | Status | Number of Errors Detected | Action Required  |
|-----------------------|---------------------|--------|---------------------------|--|
| 1                     | 0                   | OK     | 0                         | None   |
| 1                     | $\neq 0$            | OK     | ECC_NB_ERRORS             | Correct the data buffer read based on the ELM_ERROR_LOCATION_0_i to ELM_ERROR_LOCATION_15_i results. |
| 0                     | Any                 | Failed | Unknown                   | Software-dependant   |

#### 7.4.3.6 Processing Completion

When the processing for a given syndrome polynomial completes, its ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID bit is reset. It must not be set again until the exit status registers, ELM\_LOCATION\_STATUS\_i ( $i = 0$  to 7), for this processing are checked. Failure to comply with this rule leads to potential loss of the first polynomial process data output.

The error-location engine signals the process completion to the ELM. When this event is detected, the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit is set ( $i = 0$  to 7). The processing-exit status is available from the associated ELM\_LOCATION\_STATUS\_i register, and error locations are stored in order in the ECC\_ERROR\_LOCATION fields. The software must only read valid error-location registers based on the number of errors detected and located.

Immediately after the error-location engine completes, a new syndrome polynomial can be processed, if any is available, as reported by the ELM\_SYNDROME\_FRAGMENT\_6\_i[16] SYNDROME\_VALID validity bit, depending on the configured error-correction level. If several syndrome polynomials are available, a round-robin arbitration is used to select one for processing.

In continuous mode (that is, all bits in ELM\_PAGE\_CTRL are reset), an interrupt is triggered whenever a ELM\_IRQSTATUS[i] LOC\_VALID\_i bit is asserted. The CPU must read the ELM\_IRQSTATUS register to determine which polynomial is processed and retrieve the exit status and error locations (ELM\_LOCATION\_STATUS\_i and ELM\_ERROR\_LOCATION\_0\_i to ELM\_ERROR\_LOCATION\_15\_i). When done, the CPU must clear the corresponding ELM\_IRQSTATUS[i] LOC\_VALID\_i bit by writing it to 1. Other status bits must be written to 0 so that other interrupts are not unintentionally cleared. When using this mode, the ELM\_IRQSTATUS[8] PAGE\_VALID interrupt is never triggered.

In page mode, the module does not trigger interrupts for the processing completion of each polynomial because the ELM\_IRQENABLE[i] LOCATION\_MASK\_i bits are cleared. A page is defined using the ELM\_PAGE\_CTRL register. Each SECTOR\_i bit set means the corresponding polynomial i is part of the page processing. A page is fully processed when all tagged polynomials have been processed, as logged in the ELM\_IRQSTATUS[i] LOC\_VALID\_i bit fields. The module triggers an ELM\_IRQSTATUS[8] PAGE\_VALID interrupt whenever it detects that the full page has been processed. To make sure the next page can be correctly processed, all status bits in the ELM\_IRQSTATUS register must be cleared by using a single atomic-write access.

---

**NOTE:** Do not modify page setting parameters in the ELM\_PAGE\_CTRL register unless the engine is idle, no polynomial input is valid, and all interrupts have been cleared.

---

Because no polynomial-level interrupt is triggered in page mode, polynomials cleared in the ELM\_PAGE\_CTRL[i] SECTOR\_i bit fields ( $i = 0$  to 7) are processed as usual, but are essentially ignored. The CPU must manually poll the ELM\_IRQSTATUS bits to check for their status.

## 7.4.4 Basic Programming Model

### 7.4.4.1 ELM Low Level Programming Model

#### 7.4.4.1.1 Processing Initialization

**Table 7-167. ELM Processing Initialization**

| Step   | Register/ Bit Field / Programming Model  | Value  |
|--|--|--|
| Resets the module  | ELM_SYSCONFIG[1] SOFTRESET   | 0x1  |
| Wait until reset is done.  | ELM_SYSSTATUS[0] RESETDONE   | 0x1  |
| Configure the slave interface power management.                                | ELM_SYSCONFIG[4:3] SIDLEMODE   | Set value  |
| Defines the error-correction level used  | ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL   | Set value  |
| Defines the maximum buffer length  | ELM_LOCATION_CONFIG[26:16] ECC_SIZE  | Set value  |
| Sets the ELM in continuous mode or page mode                                   | ELM_PAGE_CTRL  | Set value  |
| <b>If</b> continuous mode is used  | All ELM_PAGE_CTRL[i] SECTOR_i (i = 0 to 7)   | 0x0  |
| Enables interrupt for syndrome polynomial i                                    | ELM_IRQENABLE[i] LOCATION_MASK_i   | 0x1  |
| <b>else</b> (page mode is used)  | One syndrome polynomial i is set ELM_PAGE_CTRL[i] SECTOR_i (i = 0 to 7)  | 0x1  |
| Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt. | All ELM_IRQENABLE[i] LOCATION_MASK_i = 0x0 and ELM_IRQENABLE[8] PAGE_MASK = 0x1                                  | Set value  |
| <b>endif</b>   |  | Set value  |
| Set the input syndrome polynomial i.   | ELM_SYNDROME_FRAGMENT_0_i<br>ELM_SYNDROME_FRAGMENT_1_i<br>ELM_SYNDROME_FRAGMENT_5_i<br>ELM_SYNDROME_FRAGMENT_6_i | Set value<br>Set value<br>Set value<br>Set value |
| Initiates the computation process  | ELM_SYNDROME_FRAGMENT_6_i[16]<br>SYNDROME_VALID  | 0x1  |

#### 7.4.4.1.2 Read Results

The engine goes through the entire error-location process and results can be read. [Table 7-168](#) and [Table 7-169](#) describe the processing completion for continuous and page modes, respectively.

**Table 7-168. ELM Processing Completion for Continuous Mode**

| Step   | Register/ Bit Field / Programming Model  | Value |
|--|--|-------|
| Wait until process is complete for syndrome polynomial i:<br>Wait until the ELM_IRQ interrupt is generated, or poll the status register.   |  |       |
| Read for which i the error-location process is complete.   | ELM_IRQSTATUS[i] LOC_VALID_i   | 0x1   |
| <b>if</b> the process fails (too many errors)  | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE   | 0x0   |
| It is software dependant.  |  |       |
| <b>else</b> (process successful, the engine completes)   | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE   | 0x1   |
| Read the number of errors.   | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS   |       |
| Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers.<br>It is the software responsibility to correct errors in the data buffer. | ELM_ERROR_LOCATION_0_i[12:0]<br>ECC_ERROR_LOCATION<br>ELM_ERROR_LOCATION_1_i[12:0]<br>ECC_ERROR_LOCATION<br>...<br>ELM_ERROR_LOCATION_15_i[12:0]<br>ECC_ERROR_LOCATION |       |

**Table 7-168. ELM Processing Completion for Continuous Mode (continued)**

| Step                                 | Register/ Bit Field / Programming Model | Value |
|--------------------------------------|---|-------|
| <b>endif</b>                         |   |       |
| Clear the corresponding i interrupt. | ELM_IRQSTATUS[i] LOC_VALID_i            | 0x1   |

A new syndrome polynomial can be processed after the end of processing (ELM\_SYNDRome\_FRAGMENT\_6\_i[16] SYNDROME\_VALID = 0x0) and after the exit status register check (ELM\_LOCATION\_STATUS\_i).

**Table 7-169. ELM Processing Completion for Page Mode**

| Step   | Register/ Bit Field / Programming Model             | Value |
|--|---|-------|
| Wait until process is complete for syndrome polynomial i:  |   |       |
| Wait until the ELM_IRQ interrupt is generated, or poll the status register.  |   |       |
| Wait for page completed interrupt:<br>All error locations are valid.   | ELM_IRQSTATUS[8] PAGE_VALID                         | 0x1   |
| <b>Repeat</b> the following actions the necessary number of times. That is, once for each valid defined block in the page. |   |       |
| Read the process exit status.  | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE            |       |
| <b>if</b> the process fails (too many errors)  | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE            | 0x0   |
| It is software dependant.  |   |       |
| <b>else</b> (process successful, the engine completes)   | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE            | 0x1   |
| Read the number of errors.   | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS            |       |
| Read the error-location bit addresses for syndrome polynomial i of the ECC_NB_ERRORS first registers.                      | ELM_ERROR_LOCATION_0_i[12:0]<br>ECC_ERROR_LOCATION  |       |
|  | ELM_ERROR_LOCATION_1_i[12:0]<br>ECC_ERROR_LOCATION  |       |
|  | ...   |       |
|  | ELM_ERROR_LOCATION_15_i[12:0]<br>ECC_ERROR_LOCATION |       |
| <b>endif</b>   |   |       |
| <b>End Repeat</b>  |   |       |
| Clear the ELM_IRQSTATUS register.  | ELM_IRQSTATUS                                       | 0x1FF |

Next page can be correctly processed after a page is fully processed, when all tagged polynomials have been processed (ELM\_IRQSTATUS[i] LOC\_VALID\_i = 0x1 for all syndrome polynomials i used in the page).

#### 7.4.4.2 Use Case: ELM Used in Continuous Mode

In this example, the ELM module is programmed for an 8-bit error-correction capability in continuous mode. After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, a non-zero polynomial syndrome is reported from the GPMC (Polynomial syndrome 0 is used in the ELM):

- P = 0xA16ABE115E44F767BFB0D0980

**Table 7-170. Use Case: Continuous Mode**

| Step   | Register/ Bit Field / Programming Model | Value |
|--|---|-------|
| Resets the module  | ELM_SYSConfig[1] SOFTRESET              | 0x1   |
| Wait until reset is done.  | ELM_SYSSTATUS[0] RESETDONE              | 0x1   |
| Configure the slave interface power management:<br>Smart idle is used. | ELM_SYSConfig[4:3] SIDLEMODE            | 0x2   |

**Table 7-170. Use Case: Continuous Mode (continued)**

| Step   | Register/ Bit Field / Programming Model  | Value  |
|--|--|--|
| Defines the error-correction level used: 8 bits  | ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL   | 0x1  |
| Defines the maximum buffer length: 528 bytes (2x528 = 1056)  | ELM_LOCATION_CONFIG[26:16] ECC_SIZE  | 0x420  |
| Sets the ELM in continuous mode  | ELM_PAGE_CTRL  | 0  |
| Enables interrupt for syndrome polynomial 0  | ELM_IRQENABLE[0] LOCATION_MASK_0   | 0x1  |
| Set the input syndrome polynomial 0.   | ELM_SYNDROME_FRAGMENT_0_i (i=0)<br>ELM_SYNDROME_FRAGMENT_1_i (i=0)<br>ELM_SYNDROME_FRAGMENT_2_i (i=0)<br>ELM_SYNDROME_FRAGMENT_3_i (i=0) | 0xFB0D0980<br>0xE44F767B<br>0x16ABE115<br>0x0000000A |
| Initiates the computation process  | ELM_SYNDROME_FRAGMENT_6_i[16]<br>SYNDROME_VALID (i=0)  | 0x1  |
| Wait until process is complete for syndrome polynomial 0:<br>IRQ_ELM is generated or poll the status register.   |  |  |
| Read that error-location process is complete for syndrome polynomial 0.  | ELM_IRQSTATUS[0] LOC_VALID_0   | 0x1  |
| Read the process exit status:<br>All errors were successfully located.   | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE (i=0)   | 0x1  |
| Read the number of errors:<br>Four errors detected.  | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS (i=0)   | 0x4  |
| Read the error-location bit addresses for syndrome polynomial 0 of the 4 first registers:<br>Errors are located in the data buffer at decimal addresses 431, 1062, 1909, 3452. | ELM_ERROR_LOCATION_0_i (i=0)<br>ELM_ERROR_LOCATION_1_i (i=0)<br>ELM_ERROR_LOCATION_2_i (i=0)<br>ELM_ERROR_LOCATION_3_i (i=0)             | 0x1AF<br>0x426<br>0x775<br>0xD7C                     |
| Clear the corresponding interrupt for polynomial 0.  | ELM_IRQSTATUS[0] LOC_VALID_0   | 0x1  |

The NAND flash data in the sector are seen as a polynomial of degree 4223 (number of bits in a 528 byte buffer minus 1), with each data bit being a coefficient in the polynomial. When reading from a NAND flash using the GPMC module, computation of the polynomial syndrome assumes that the first NAND word read at address 0x0 contains the highest-order coefficient in the message. Furthermore, in the 16-bit NAND word, bits are ordered from bit 7 to bit 0, then from bit 15 to bit 8. Based on this convention, an address table of the data buffer can be built. NAND memory addresses in [Table 7-171](#) are given in decimal format.

**Table 7-171. 16-bit NAND Sector Buffer Address Map**

| NAND<br>Memory<br>Address | Message bit addresses in the memory word |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|---------------------------|--|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
|                           | 15                                       | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| 0                         | 4215                                     | 4214 | 4213 | 4212 | 4211 | 4210 | 4209 | 4208 | 4223 | 4222 | 4221 | 4220 | 4219 | 4218 | 4217 | 4216 |
| 1                         | 4175                                     | 4174 | 4173 | 4172 | 4171 | 4170 | 4169 | 4168 | 4183 | 4182 | 4181 | 4180 | 4179 | 4178 | 4177 | 4176 |
| ...                       |  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| 47                        | 3463                                     | 3462 | 3461 | 3460 | 3459 | 3458 | 3457 | 3456 | 3471 | 3470 | 3469 | 3468 | 3467 | 3466 | 3465 | 3464 |
| 48                        | 3447                                     | 3446 | 3445 | 3444 | 3443 | 3442 | 3441 | 3440 | 3455 | 3454 | 3453 | 3452 | 3451 | 3450 | 3449 | 3448 |
| 49                        | 3431                                     | 3430 | 3429 | 3428 | 3427 | 3426 | 3425 | 3424 | 3439 | 3438 | 3437 | 3436 | 3435 | 3434 | 3433 | 3432 |
| 50                        | 3415                                     | 3414 | 3413 | 3412 | 3411 | 3410 | 3409 | 3408 | 3423 | 3422 | 3421 | 3420 | 3419 | 3418 | 3417 | 3416 |
| ...                       |  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| 255                       | 135                                      | 134  | 133  | 132  | 131  | 130  | 129  | 128  | 143  | 142  | 141  | 140  | 139  | 138  | 137  | 136  |
| 256                       | 119                                      | 118  | 117  | 116  | 115  | 114  | 113  | 112  | 127  | 126  | 125  | 124  | 123  | 122  | 121  | 120  |
| 257                       | 103                                      | 102  | 101  | 100  | 99   | 98   | 97   | 96   | 111  | 110  | 109  | 108  | 107  | 106  | 105  | 104  |
| 258                       | 87                                       | 86   | 85   | 84   | 83   | 82   | 81   | 80   | 95   | 94   | 93   | 92   | 91   | 90   | 89   | 88   |
| 259                       | 71                                       | 70   | 69   | 68   | 67   | 66   | 65   | 64   | 79   | 78   | 77   | 76   | 75   | 74   | 73   | 72   |
| 260                       | 55                                       | 54   | 53   | 52   | 51   | 50   | 49   | 48   | 63   | 62   | 61   | 60   | 59   | 58   | 57   | 56   |
| 261                       | 39                                       | 38   | 37   | 36   | 35   | 34   | 33   | 32   | 47   | 46   | 45   | 44   | 43   | 42   | 41   | 40   |
| 262                       | 23                                       | 22   | 21   | 20   | 19   | 18   | 17   | 16   | 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   |
| 263                       | 7  | 6    | 5    | 4    | 3    | 2    | 1    | 0    | 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    |

The table can now be used to determine which bits in the buffer were incorrect and must be flipped. In this example, the first bit to be flipped is bit 4 from the 49th byte read from memory. It is up to the processor to correctly map this word to the copied buffer and to flip this bit. The same process must be repeated for all detected errors.

#### 7.4.4.3 Use Case: ELM Used in Page Mode

In this example, the ELM module is programmed for an 16-bit error-correction capability in page mode. After reading a 528-byte NAND flash sector (512B data plus 16B spare area) with a 16-bit interface, four non-zero polynomial syndromes are reported from the GPMC (Polynomial syndrome 0, 1, 2, and 3 are used in the ELM):

- P0 = 0xE8B0 12ADD5A318E05BE B0693DB28330B5CC A329AA05E0B718EF
- P1 = 0xBAD0 49A0D932C22E6669 0948DF08BE093336 79C6BA10E5F935EB
- P2 = 0x69D9 B86ABCD5EC3697FA A6498FEE54556EA0 1579EF7D60BA3189
- P3 = 0x0

**Table 7-172. Use Case: Page Mode**

| Step   | Register/ Bit Field / Programming Model | Value |
|--|---|-------|
| Resets the module  | ELM_SYSCONFIG[1] SOFTRESET              | 0x1   |
| Wait until reset is done.  | ELM_SYSSTATUS[0] RESETDONE              | 0x1   |
| Configure the slave interface power management:<br>Smart idle is used. | ELM_SYSCONFIG[4:3] SIDLEMODE            | 0x2   |
| Defines the error-correction level used: 16 bits                       | ELM_LOCATION_CONFIG[1:0] ECC_BCH_LEVEL  | 0x2   |
| Defines the maximum buffer length: 528 bytes                           | ELM_LOCATION_CONFIG[26:16] ECC_SIZE     | 0x420 |
| Sets the ELM in page mode (4 blocks in a page)                         | ELM_PAGE_CTRL[0] SECTOR_0               | 0x1   |
|  | ELM_PAGE_CTRL[1] SECTOR_1               | 0x1   |
|  | ELM_PAGE_CTRL[2] SECTOR_2               | 0x1   |
|  | ELM_PAGE_CTRL[3] SECTOR_3               | 0x1   |

**Table 7-172. Use Case: Page Mode (continued)**

| Step   | Register/ Bit Field / Programming Model                 | Value      |
|--|---|------------|
| Disable all interrupts for syndrome polynomial and enable PAGE_MASK interrupt.   | ELM_IRQENABLE   | 0x100      |
| Set the input syndrome polynomial 0.   | ELM_SYNTHESIS_FRAGMENT_0_i (i=0)                        | 0xE0B718EF |
|  | ELM_SYNTHESIS_FRAGMENT_1_i (i=0)                        | 0xA329AA05 |
|  | ELM_SYNTHESIS_FRAGMENT_2_i (i=0)                        | 0x8330B5CC |
|  | ELM_SYNTHESIS_FRAGMENT_3_i (i=0)                        | 0xB0693DB2 |
|  | ELM_SYNTHESIS_FRAGMENT_4_i (i=0)                        | 0x318E05BE |
|  | ELM_SYNTHESIS_FRAGMENT_5_i (i=0)                        | 0x12ADDB5A |
|  | ELM_SYNTHESIS_FRAGMENT_6_i (i=0)                        | 0xE8B0     |
| Set the input syndrome polynomial 1.   | ELM_SYNTHESIS_FRAGMENT_0_i (i=1)                        | 0xE5F935EB |
|  | ELM_SYNTHESIS_FRAGMENT_1_i (i=1)                        | 0x79C6BA10 |
|  | ELM_SYNTHESIS_FRAGMENT_2_i (i=1)                        | 0xBE093336 |
|  | ELM_SYNTHESIS_FRAGMENT_3_i (i=1)                        | 0x0948DF08 |
|  | ELM_SYNTHESIS_FRAGMENT_4_i (i=1)                        | 0xC22E6669 |
|  | ELM_SYNTHESIS_FRAGMENT_5_i (i=1)                        | 0x49A0D932 |
|  | ELM_SYNTHESIS_FRAGMENT_6_i (i=1)                        | 0xBAD0     |
| Set the input syndrome polynomial 2.   | ELM_SYNTHESIS_FRAGMENT_0_i (i=2)                        | 0x60BA3189 |
|  | ELM_SYNTHESIS_FRAGMENT_1_i (i=2)                        | 0x1579EF7D |
|  | ELM_SYNTHESIS_FRAGMENT_2_i (i=2)                        | 0x54556EA0 |
|  | ELM_SYNTHESIS_FRAGMENT_3_i (i=2)                        | 0xA6498FEE |
|  | ELM_SYNTHESIS_FRAGMENT_4_i (i=2)                        | 0xEC3697FA |
|  | ELM_SYNTHESIS_FRAGMENT_5_i (i=2)                        | 0xB86ABCD5 |
|  | ELM_SYNTHESIS_FRAGMENT_6_i (i=2)                        | 0x69D9     |
| Set the input syndrome polynomial 3.   | ELM_SYNTHESIS_FRAGMENT_0_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_1_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_2_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_3_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_4_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_5_i (i=3)                        | 0x0        |
|  | ELM_SYNTHESIS_FRAGMENT_6_i (i=3)                        | 0x0        |
| Initiates the computation process for syndrome polynomial 0  | ELM_SYNTHESIS_FRAGMENT_6_i[16]<br>SYNTHESIS_VALID (i=0) | 0x1        |
| Initiates the computation process for syndrome polynomial 1  | ELM_SYNTHESIS_FRAGMENT_6_i[16]<br>SYNTHESIS_VALID (i=1) | 0x1        |
| Initiates the computation process for syndrome polynomial 2  | ELM_SYNTHESIS_FRAGMENT_6_i[16]<br>SYNTHESIS_VALID (i=2) | 0x1        |
| Initiates the computation process for syndrome polynomial 3  | ELM_SYNTHESIS_FRAGMENT_6_i[16]<br>SYNTHESIS_VALID (i=3) | 0x1        |
| Wait until process is complete for syndrome polynomial 0, 1, 2, and 3:<br>Wait until the ELM_IRQ interrupt is generated or poll the status register. |   |            |
| Wait for page completed interrupt:<br>All error locations are valid.   | ELM_IRQSTATUS[8] PAGE_VALID                             | 0x1        |
| Read the process exit status for syndrome polynomial 0:<br>All errors were successfully located.   | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE<br>(i=0)       | 0x1        |
| Read the process exit status for syndrome polynomial 1:<br>All errors were successfully located.   | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE<br>(i=1)       | 0x1        |

**Table 7-172. Use Case: Page Mode (continued)**

| <b>Step</b>  | <b>Register/ Bit Field / Programming Model</b>   | <b>Value</b>                     |
|--|--|----------------------------------|
| Read the process exit status for syndrome polynomial 2:<br>All errors were successfully located. | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE<br>(i=2)  | 0x1                              |
| Read the process exit status for syndrome polynomial 3:<br>All errors were successfully located. | ELM_LOCATION_STATUS_i[8] ECC_CORRECTABLE<br>(i=3)  | 0x1                              |
| Read the number of errors for syndrome polynomial 0:<br>4 errors detected.                       | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS<br>(i=0)  | 0x4                              |
| Read the number of errors for syndrome polynomial 1:<br>2 errors detected.                       | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS<br>(i=1)  | 0x2                              |
| Read the number of errors for syndrome polynomial 2:<br>1 error detected.                        | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS<br>(i=2)  | 0x1                              |
| Read the number of errors for syndrome polynomial 3:<br>0 errors detected.                       | ELM_LOCATION_STATUS_i[4:0] ECC_NB_ERRORS<br>(i=3)  | 0x0                              |
| Read the error-location bit addresses for syndrome polynomial 0 of the 4 first registers:        | ELM_ERROR_LOCATION_0_i (i=0)<br>ELM_ERROR_LOCATION_1_i (i=0)<br>ELM_ERROR_LOCATION_2_i (i=0)<br>ELM_ERROR_LOCATION_3_i (i=0) | 0x1FE<br>0x617<br>0x650<br>0xA83 |
| Read the error-location bit addresses for syndrome polynomial 1 of the 2 first registers:        | ELM_ERROR_LOCATION_0_i (i=1)<br>ELM_ERROR_LOCATION_1_i (i=1)   | 0x4<br>0x1036                    |
| Read the errors location bit addresses for syndrome polynomial 2 of the first registers:         | ELM_ERROR_LOCATION_0_i (i=1)   | 0x3E8                            |
| Clear the ELM_IRQSTATUS register.  | ELM_IRQSTATUS  | 0xFF                             |

## 7.4.5 ELM Registers

**Table 7-173. ELM Registers**

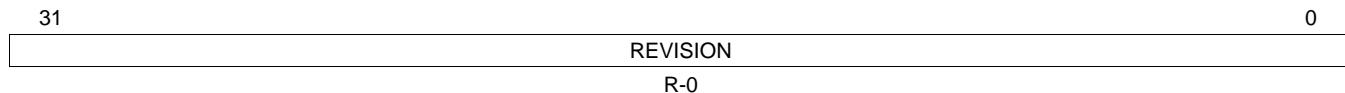
| Address Offset            | Acronym                                  | Description                         | Section                          |
|---------------------------|--|-------------------------------------|----------------------------------|
| 0h                        | ELM_REVISION                             | ELM Revision Register               | <a href="#">Section 7.4.5.1</a>  |
| 10h                       | ELM_SYSCONFIG                            | ELM System Configuration Register   | <a href="#">Section 7.4.5.2</a>  |
| 14h                       | ELM_SYSSTATUS                            | ELM System Status Register          | <a href="#">Section 7.4.5.3</a>  |
| 18h                       | ELM_IRQSTATUS                            | ELM Interrupt Status Register       | <a href="#">Section 7.4.5.4</a>  |
| 1Ch                       | ELM_IRQENABLE                            | ELM Interrupt Enable Register       | <a href="#">Section 7.4.5.5</a>  |
| 20h                       | ELM_LOCATION_CONFIG                      | ELM Location Configuration Register | <a href="#">Section 7.4.5.6</a>  |
| 80h                       | ELM_PAGE_CTRL                            | ELM Page Definition Register        | <a href="#">Section 7.4.5.7</a>  |
| 400h +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_0_i <sup>(1)</sup> | ELM_SYNDROME_FRAGMENT_0_i Register  | <a href="#">Section 7.4.5.8</a>  |
| 404h+<br>(40h × i)        | ELM_SYNDROME_FRAGMENT_1_i                | ELM_SYNDROME_FRAGMENT_1_i Register  | <a href="#">Section 7.4.5.9</a>  |
| 408h +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_2_i                | ELM_SYNDROME_FRAGMENT_2_i Register  | <a href="#">Section 7.4.5.10</a> |
| 40Ch +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_3_i                | ELM_SYNDROME_FRAGMENT_3_i Register  | <a href="#">Section 7.4.5.11</a> |
| 410h +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_4_i                | ELM_SYNDROME_FRAGMENT_4_i Register  | <a href="#">Section 7.4.5.12</a> |
| 414h +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_5_i                | ELM_SYNDROME_FRAGMENT_5_i Register  | <a href="#">Section 7.4.5.13</a> |
| 418h +<br>(40h × i)       | ELM_SYNDROME_FRAGMENT_6_1                | ELM_SYNDROME_FRAGMENT_6_1 Register  | <a href="#">Section 7.4.5.14</a> |
| 800h +<br>(100h × i)      | ELM_LOCATION_STATUS_i                    | ELM_LOCATION_STATUS_i Register      | <a href="#">Section 7.4.5.15</a> |
| 880h-8BCh +<br>(100h × i) | ELM_ERROR_LOCATION_0-15_i                | ELM_ERROR_LOCATION_0-15_i Registers | <a href="#">Section 7.4.5.16</a> |

<sup>(1)</sup> i = 0 to 8

#### 7.4.5.1 ELM Revision Register (ELM\_REVISION)

This register contains the IP revision code. The ELM Revision Register (ELM\_REVISION) is shown in [Figure 7-141](#) and described in [Table 7-174](#).

**Figure 7-141. ELM Revision Register (ELM\_REVISION)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

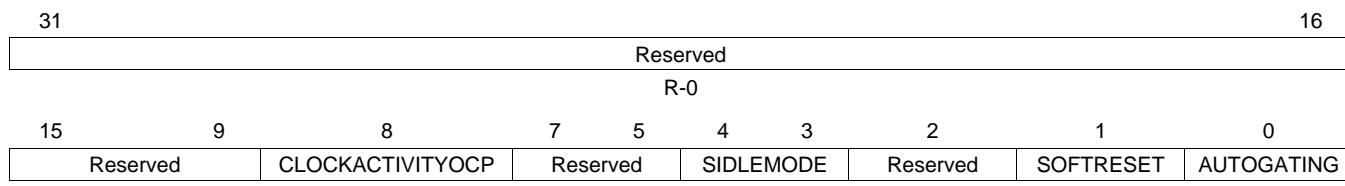
**Table 7-174. ELM Revision Register (ELM\_REVISION) Field Descriptions**

| Bit  | Field    | Value        | Description |
|------|----------|--------------|-------------|
| 31-0 | REVISION | 0xFFFF FFFFh | IP Revision |

#### 7.4.5.2 ELM System Configuration Register (ELM\_SYSCONFIG)

This register allows controlling various parameters of the OCP interface. The ELM System Configuration Register (ELM\_SYSCONFIG) is shown in [Figure 7-142](#) and described in [Table 7-175](#).

**Figure 7-142. ELM System Configuration Register (ELM\_SYSCONFIG)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-175. ELM System Configuration Register (ELM\_SYSCONFIG) Field Descriptions**

| Bit  | Field            | Value               | Description  |
|------|------------------|---------------------|--|
| 31-9 | Reserved         | 0                   | Reserved   |
| 8    | CLOCKACTIVITYOCP | 0<br>1              | OCP Clock activity when module is in IDLE mode (during wake up mode period).<br>OCP Clock can be switch-off<br>OCP Clock is maintained during wake up period   |
| 7-5  | Reserved         | 0                   | Reserved   |
| 4-3  | SIDLEMODE        | 0<br>1h<br>2h<br>3h | Slave interface power management (IDLE req/ack control).<br>FORCE Idle. IDLE request is acknowledged unconditionally and immediately. (Default <i>Dumb</i> mode for safety)<br>NO idle. IDLE request is never acknowledged.<br>SMART Idle. The acknowledgment to an IDLE request is given based on the internal activity.<br>Reserved - do not use |
| 2    | Reserved         | 0                   | Reserved   |
| 1    | SOFTRESET        | 0<br>1              | Module software reset. This bit is automatically reset by hardware (During reads, it always returns 0.). It has same effect as the OCP hardware reset.<br>Normal mode.<br>Start soft reset sequence.   |
| 0    | AUTOGATING       | 0<br>1              | Internal OCP clock gating strategy. (No module visible impact other than saving power.)<br>OCP clock is free-running.<br>Automatic internal OCP clock gating strategy is applied based on the OCP interface activity.  |

#### 7.4.5.3 ELM System Status Register (ELM\_SYSSTATUS)

The ELM System Status Register (ELM\_SYSSTATUS) is shown in Figure 7-143 and described in Table 7-176.

**Figure 7-143. ELM System Status Register (ELM\_SYSSTATUS)**

|    |          |          |
|----|----------|----------|
| 31 | Reserved | 1      0 |
|    | R-0      | R-0      |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-176. ELM System Status Register (ELM\_SYSSTATUS) Field Descriptions**

| Bit  | Field     | Value  | Description  |
|------|-----------|--------|--|
| 31-1 | Reserved  | 0      | Reserved   |
| 0    | RESETDONE | 0<br>1 | Internal reset monitoring (OCP domain). Undefined since:<br>From hardware perspective, the reset state is 0.<br>From software user perspective, when the accessible module is 1.<br><br>Reset is on-going<br>Reset is done (completed) |

#### 7.4.5.4 ELM Interrupt Status Register (ELM\_IRQSTATUS)

This register doubles as a status register for the error-location processes. The ELM Interrupt Status Register (ELM\_IRQSTATUS) is shown in [Figure 7-144](#) and described in [Table 7-177](#).

**Figure 7-144. ELM Interrupt Status Register (ELM\_IRQSTATUS)**

|             |             |             |             |             |             |             |             |  |    |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|--|----|
| 31          | Reserved    |             |             |             |             |             |             |  | 16 |
| R-0         |             |             |             |             |             |             |             |  |    |
| 15          | Reserved    |             |             |             |             |             |             |  | 8  |
| R-0         |             |             |             |             |             |             |             |  |    |
| 7           | 6           | 5           | 4           | 3           | 2           | 1           | 0           |  |    |
| LOC_VALID_7 | LOC_VALID_6 | LOC_VALID_5 | LOC_VALID_4 | LOC_VALID_3 | LOC_VALID_2 | LOC_VALID_1 | LOC_VALID_0 |  |    |
| R/W-0       |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-177. ELM Interrupt Status Register (ELM\_IRQSTATUS) Field Descriptions**

| Bit  | Field       | Value | Description   |
|------|-------------|-------|---|
| 31-9 | Reserved    | 0     | Reserved  |
| 8    | PAGE_VALID  | 0     | Error-location status for a full page, based on the mask definition.                          |
|      |             | 0     | Read: Error locations invalid for all polynomials enabled in the ECC_INTERRUPT_MASK register. |
|      |             | 1     | Read: All error locations valid.  |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |
| 7    | LOC_VALID_7 | 0     | Error-location status for syndrome polynomial 7.  |
|      |             | 0     | Read: No syndrome processed or process in progress.   |
|      |             | 1     | Read: Error-location process completed.   |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |
| 6    | LOC_VALID_6 | 0     | Error-location status for syndrome polynomial 6.  |
|      |             | 0     | Read: No syndrome processed or process in progress.   |
|      |             | 1     | Read: Error-location process completed.   |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |
| 5    | LOC_VALID_5 | 0     | Error-location status for syndrome polynomial 5.  |
|      |             | 0     | Read: No syndrome processed or process in progress.   |
|      |             | 1     | Read: Error-location process completed.   |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |
| 4    | LOC_VALID_4 | 0     | Error-location status for syndrome polynomial 4.  |
|      |             | 0     | Read: No syndrome processed or process in progress.   |
|      |             | 1     | Read: Error-location process completed.   |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |
| 3    | LOC_VALID_3 | 0     | Error-location status for syndrome polynomial 3.  |
|      |             | 0     | Read: No syndrome processed or process in progress.   |
|      |             | 1     | Read: Error-location process completed.   |
|      |             | 0     | Write: No effect.   |
|      |             | 1     | Write: Clear interrupt.   |

**Table 7-177. ELM Interrupt Status Register (ELM\_IRQSTATUS) Field Descriptions (continued)**

| Bit | Field       | Value | Description   |
|-----|-------------|-------|---|
| 2   | LOC_VALID_2 | 0     | Error-location status for syndrome polynomial 2.<br>Read: No syndrome processed or process in progress. |
|     |             | 1     | Read: Error-location process completed.   |
| 1   | LOC_VALID_1 | 0     | Write: No effect.   |
|     |             | 1     | Write: Clear interrupt.   |
| 0   | LOC_VALID_0 | 0     | Error-location status for syndrome polynomial 0.<br>Read: No syndrome processed or process in progress. |
|     |             | 1     | Read: Error-location process completed.   |
|     |             | 0     | Write: No effect.   |
|     |             | 1     | Write: Clear interrupt.   |

#### 7.4.5.5 ELM Interrupt Enable Register (ELM\_IRQENABLE)

The ELM Interrupt Enable Register (ELM\_IRQENABLE) is shown in [Figure 7-145](#) and described in [Table 7-178](#).

**Figure 7-145. ELM Interrupt Enable Register (ELM\_IRQENABLE)**

|    |                 |                 |                 |                 |                 |                 |                 |                 |    |
|----|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----|
| 31 | Reserved        |                 |                 |                 |                 |                 |                 |                 | 16 |
|    | R-0             |                 |                 |                 |                 |                 |                 |                 |    |
| 15 | Reserved        |                 |                 |                 |                 |                 |                 |                 |    |
|    | PAGE_MASK       |                 |                 |                 |                 |                 |                 |                 |    |
|    | R/W-0           |                 |                 |                 |                 |                 |                 |                 |    |
| 7  | LOCATION_MASK_7 | LOCATION_MASK_6 | LOCATION_MASK_5 | LOCATION_MASK_4 | LOCATION_MASK_3 | LOCATION_MASK_2 | LOCATION_MASK_1 | LOCATION_MASK_0 | 0  |
|    | R/W-0           |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-178. ELM Interrupt Enable Register (ELM\_IRQENABLE) Field Descriptions**

| Bit  | Field           | Value | Description  |
|------|-----------------|-------|--|
| 31-9 | Reserved        | 0     | Reserved   |
| 8    | PAGE_MASK       | 0     | Page interrupt mask bit                                      |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 7    | LOCATION_MASK_7 | 0     | Error-location interrupt mask bit for syndrome polynomial 7. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 6    | LOCATION_MASK_6 | 0     | Error-location interrupt mask bit for syndrome polynomial 6. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 5    | LOCATION_MASK_5 | 0     | Error-location interrupt mask bit for syndrome polynomial 5. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 4    | LOCATION_MASK_4 | 0     | Error-location interrupt mask bit for syndrome polynomial 4. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 3    | LOCATION_MASK_3 | 0     | Error-location interrupt mask bit for syndrome polynomial 3. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 2    | LOCATION_MASK_2 | 0     | Error-location interrupt mask bit for syndrome polynomial 2. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 1    | LOCATION_MASK_1 | 0     | Error-location interrupt mask bit for syndrome polynomial 1. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |
| 0    | LOCATION_MASK_0 | 0     | Error-location interrupt mask bit for syndrome polynomial 0. |
|      |                 | 1     | Disable interrupt.   |
|      |                 | 1     | Enable interrupt.  |

#### 7.4.5.6 ELM Location Configuration Register (ELM\_LOCATION\_CONFIG)

The ELM Location Configuration Register (ELM\_LOCATION\_CONFIG) is shown in [Figure 7-146](#) and described in [Table 7-179](#).

**Figure 7-146. ELM Location Configuration Register (ELM\_LOCATION\_CONFIG)**

| 31       | 27 | 16       | 16 | 15       | 2 | 1             | 0 |
|----------|----|----------|----|----------|---|---------------|---|
| Reserved |    | ECC_SIZE |    | Reserved |   | ECC_BCH_LEVEL |   |
| R-0      |    | R/W-0    |    | R-0      |   | R/W-0         |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

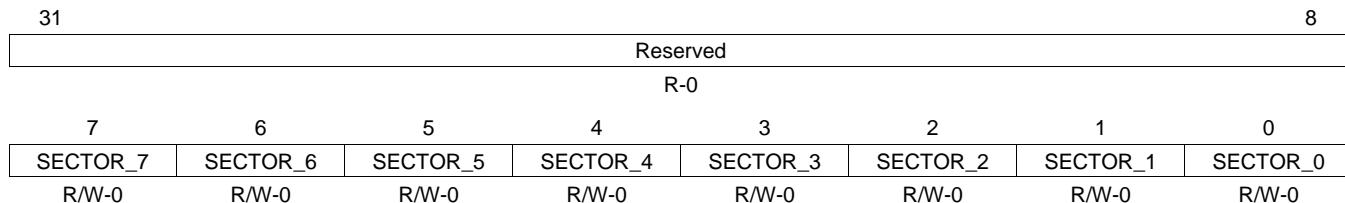
**Table 7-179. ELM Location Configuration Register (ELM\_LOCATION\_CONFIG) Field Descriptions**

| Bit   | Field         | Value               | Description   |
|-------|---------------|---------------------|---|
| 31-27 | Reserved      | 0                   | Reserved  |
| 26-16 | ECC_SIZE      | 0-7FFh              | Maximum size of the buffers for which the error-location engine is used, in number of nibbles (4-bits entities) |
| 15-2  | Reserved      | 0                   | Reserved  |
| 1-0   | ECC_BCH_LEVEL | 0<br>1h<br>2h<br>3h | Error correction level.<br>0 4 bits.<br>1h 8 bits.<br>2h 16 bits.<br>3h Reserved.                               |

#### 7.4.5.7 ELM Page Definition Register (ELM\_PAGE\_CTRL)

The ELM Page Definition Register (ELM\_PAGE\_CTRL) is shown in [Figure 7-147](#) and described in [Table 7-180](#).

**Figure 7-147. ELM Page Definition Register (ELM\_PAGE\_CTRL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-180. ELM Page Definition Register (ELM\_PAGE\_CTRL) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved.   |
| 7    | SECTOR_7 | 0-1   | Set to 1 if syndrome polynomial 7 is part of the page in page mode. Must be 0 in continuous mode. |
| 6    | SECTOR_6 | 0-1   | Set to 1 if syndrome polynomial 6 is part of the page in page mode. Must be 0 in continuous mode. |
| 5    | SECTOR_5 | 0-1   | Set to 1 if syndrome polynomial 5 is part of the page in page mode. Must be 0 in continuous mode. |
| 4    | SECTOR_4 | 0-1   | Set to 1 if syndrome polynomial 4 is part of the page in page mode. Must be 0 in continuous mode. |
| 3    | SECTOR_3 | 0-1   | Set to 1 if syndrome polynomial 3 is part of the page in page mode. Must be 0 in continuous mode. |
| 2    | SECTOR_2 | 0-1   | Set to 1 if syndrome polynomial 2 is part of the page in page mode. Must be 0 in continuous mode. |
| 1    | SECTOR_1 | 0-1   | Set to 1 if syndrome polynomial 1 is part of the page in page mode. Must be 0 in continuous mode. |
| 0    | SECTOR_0 | 0-1   | Set to 1 if syndrome polynomial 0 is part of the page in page mode. Must be 0 in continuous mode. |

#### 7.4.5.8 ELM\_SYNDROME\_FRAGMENT\_0\_i Register

The ELM\_SYNDROME\_FRAGMENT\_0\_i Register is shown in [Figure 7-148](#) and described in [Table 7-181](#).

**Figure 7-148. ELM\_SYNDROME\_FRAGMENT\_0\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_0 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-181. ELM\_SYNDROME\_FRAGMENT\_0\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description            |
|------|------------|--------------|------------------------|
| 31-0 | SYNDROME_0 | 0xFFFF FFFFh | Syndrome bits 0 to 31. |

#### 7.4.5.9 ELM\_SYNDROME\_FRAGMENT\_1\_i Register

The ELM\_SYNDROME\_FRAGMENT\_1\_i Register is shown in [Figure 7-149](#) and described in [Table 7-182](#).

**Figure 7-149. ELM\_SYNDROME\_FRAGMENT\_1\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_1 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-182. ELM\_SYNDROME\_FRAGMENT\_1\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description             |
|------|------------|--------------|-------------------------|
| 31-0 | SYNDROME_1 | 0xFFFF FFFFh | Syndrome bits 32 to 63. |

#### 7.4.5.10 ELM\_SYNDROME\_FRAGMENT\_2\_i Register

The ELM\_SYNDROME\_FRAGMENT\_2\_i Register is shown in [Figure 7-150](#) and described in [Table 7-183](#).

**Figure 7-150. ELM\_SYNDROME\_FRAGMENT\_2\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_2 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-183. ELM\_SYNDROME\_FRAGMENT\_2\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description             |
|------|------------|--------------|-------------------------|
| 31-0 | SYNDROME_2 | 0xFFFF FFFFh | Syndrome bits 64 to 95. |

#### **7.4.5.11 ELM\_SYNDROME\_FRAGMENT\_3\_i Register**

The ELM\_SYNDROME\_FRAGMENT\_3\_i Register is shown in [Figure 7-151](#) and described in [Table 7-184](#).

**Figure 7-151. ELM\_SYNDROME\_FRAGMENT\_3\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_3 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-184. ELM\_SYNDROME\_FRAGMENT\_3\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description              |
|------|------------|--------------|--------------------------|
| 31-0 | SYNDROME_3 | 0xFFFF FFFFh | Syndrome bits 96 to 127. |

#### **7.4.5.12 ELM\_SYNDROME\_FRAGMENT\_4\_i Register**

The ELM\_SYNDROME\_FRAGMENT\_4\_i Register is shown in [Figure 7-152](#) and described in [Table 7-185](#).

**Figure 7-152. ELM\_SYNDROME\_FRAGMENT\_4\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_4 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-185. ELM\_SYNDROME\_FRAGMENT\_4\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description               |
|------|------------|--------------|---------------------------|
| 31-0 | SYNDROME_4 | 0xFFFF FFFFh | Syndrome bits 128 to 159. |

#### **7.4.5.13 ELM\_SYNDROME\_FRAGMENT\_5\_i Register**

The ELM\_SYNDROME\_FRAGMENT\_5\_i Register is shown in [Figure 7-153](#) and described in [Table 7-186](#).

**Figure 7-153. ELM\_SYNDROME\_FRAGMENT\_5\_i Register**

|       |            |   |
|-------|------------|---|
| 31    | SYNDROME_5 | 0 |
| R/W-0 |            |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-186. ELM\_SYNDROME\_FRAGMENT\_5\_i Register Field Descriptions**

| Bit  | Field      | Value        | Description               |
|------|------------|--------------|---------------------------|
| 31-0 | SYNDROME_5 | 0xFFFF FFFFh | Syndrome bits 160 to 191. |

#### 7.4.5.14 ELM\_SYNDROME\_FRAGMENT\_6\_i Register

The ELM\_SYNDROME\_FRAGMENT\_6\_i Register is shown in [Figure 7-154](#) and described in [Table 7-187](#).

**Figure 7-154. ELM\_SYNDROME\_FRAGMENT\_6\_i Register**

| 31       | 17 | 16             | 15 | 0          |
|----------|----|----------------|----|------------|
| Reserved |    | SYNDROME_VALID |    | SYNDROME_6 |
| R-0      |    | R/W-0          |    | R/W-0      |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-187. ELM\_SYNDROME\_FRAGMENT\_6\_i Register Field Descriptions**

| Bit   | Field          | Value   | Description                                       |
|-------|----------------|---------|---|
| 31-17 | Reserved       | 0       | Reserved  |
| 16    | SYNDROME_VALID |         | Syndrome valid bit.                               |
|       |                | 0       | This syndrome polynomial should not be processed. |
|       |                | 1       | This syndrome polynomial must be processed.       |
| 15-0  | SYNDROME_6     | 0xFFFFh | Syndrome bits 192 to 207.                         |

#### 7.4.5.15 ELM\_LOCATION\_STATUS\_i Register

The ELM\_LOCATION\_STATUS\_i Register is shown in [Figure 7-155](#) and described in [Table 7-188](#).

**Figure 7-155. ELM\_LOCATION\_STATUS\_i Register**

| 31       | 9 | 8              | 7 | 5        | 4 | 0             |
|----------|---|----------------|---|----------|---|---------------|
| Reserved |   | ECC_CORRECTABL |   | Reserved |   | ECC_NB_ERRORS |
| R-0      |   | R-0            |   | R-0      |   | R-0           |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-188. ELM\_LOCATION\_STATUS\_i Register Field Descriptions**

| Bit  | Field          | Value | Description   |
|------|----------------|-------|---|
| 31-9 | Reserved       | 0     | Reserved  |
| 8    | ECC_CORRECTABL |       | Error-location process exit status.   |
|      |                | 0     | ECC error-location process failed. Number of errors and error locations are invalid.  |
|      |                | 1     | All errors were successfully located. Number of errors and error locations are valid. |
| 7-5  | Reserved       | 0     | Reserved  |
| 4-0  | ECC_NB_ERRORS  | 0-1Fh | Number of errors detected and located.  |

#### 7.4.5.16 ELM\_ERROR\_LOCATION\_0-15\_i Registers

The ELM\_ERROR\_LOCATION\_0-15\_i Registers is shown in [Figure 7-156](#) and described in [Table 7-189](#).

**Figure 7-156. ELM\_ERROR\_LOCATION\_0-15\_i Registers**

| 31       | 13 | 12 | 0                  |
|----------|----|----|--------------------|
| Reserved |    |    | ECC_ERROR_LOCATION |
| R-0      |    |    | R-0                |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 7-189. ELM\_ERROR\_LOCATION\_0-15\_i Registers Field Descriptions**

| Bit   | Field              | Value   | Description                 |
|-------|--------------------|---------|-----------------------------|
| 31-13 | Reserved           | 0       | Reserved                    |
| 12-0  | ECC_ERROR_LOCATION | 0-1FFFh | Error-location bit address. |

## **Power, Reset, and Clock Management (PRCM)**

This chapter describes the PRCM of the device.

| Topic   | Page       |
|---|------------|
| <b>8.1 Power, Reset, and Clock Management .....</b> | <b>499</b> |

## 8.1 Power, Reset, and Clock Management

### 8.1.1 Introduction

The device power-management architecture ensures maximum performance and operation time for user satisfaction (audio/video support) while offering versatile power-management techniques for maximum design flexibility, depending on application requirements. This introduction contains the following information:

- Power-management architecture building blocks for the device
- State-of-the-art power-management techniques supported by the power-management architecture of the device

### 8.1.2 Device Power-Management Architecture Building Blocks

To provide a versatile architecture supporting multiple power-management techniques, the power-management framework is built with three levels of resource management: clock, power, and voltage management.

These management levels are enforced by defining the managed entities or building blocks of the power-management architecture, called the clock, power, and voltage domains. A domain is a group of modules or subsections of the device that share a common entity (for example, common clock source, common voltage source, or common power switch). The group forming the domain is managed by a policy manager. For example, a clock for a clock domain is managed by a dedicated clock manager within the power, reset, and clock management (PRCM) module. The clock manager considers the joint clocking constraints of all the modules belonging to that clock domain (and, hence, receiving that clock).

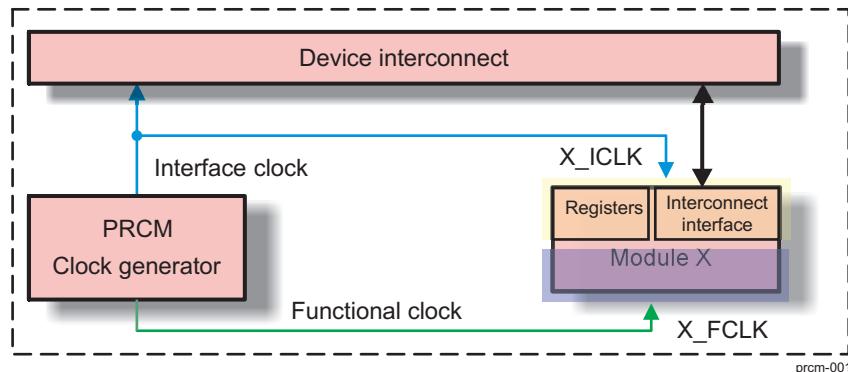
### 8.1.3 Clock Management

The PRCM module along with the control module manages the gating (that is, switching off) and enabling of the clocks to the device modules. The clocks are managed based on the requirement constraints of the associated modules. The following sections identify the module clock characteristics, management policy, clock domains, and clock domain management.

#### 8.1.3.1 Module Interface and Functional Clocks

Each module within the device has specific clock input characteristic requirements. Based on the characteristics of the clocks delivered to the modules, the clocks are divided into two categories: interface clocks and functional clocks.

**Figure 8-1. Functional and Interface Clocks**



The interface clocks have the following characteristics:

- They ensure proper communication between any module/subsystem and interconnect.
- In most cases, they supply the system interconnect interface and registers of the module.
- A typical module has one interface clock, but modules with multiple interface clocks may also exist (that is, when connected to multiple interconnect buses).

- Interface clock management is done at the device level.
- From the standpoint of the PRCM module, an interface clock is identified by an \_ICLK suffix.

Functional clocks have the following characteristics:

- They supply the functional part of a module or subsystem.
- A module can have one or more functional clocks. Some functional clocks are mandatory, while others are optional. A module needs its mandatory clock(s) to be operational. The optional clocks are used for specific features and can be shut down without stopping the module activity
- From the standpoint of the PRCM module, a functional clock is distributed directly to the related modules through a dedicated clock tree. It is identified with an \_FCLK suffix

### 8.1.3.2 Module-Level Clock Management

Each module in the device may also have specific clock requirements. Certain module clocks must be active when operating in specific modes, or may be gated otherwise. Globally, the activation and gating of the module clocks are managed by the PRCM module. Hence, the PRCM module must be aware of when to activate and when to gate the module clocks. The PRCM module differentiates the clock-management behavior for device modules based on whether the module can initiate transactions on the device interconnect (called master module or initiators) or cannot initiate transactions and only responds to the transactions initiated by the master (called slave module or targets). Thus, two hardware-based power-management protocols are used:

- Master standby protocol: Clock-management protocol between the PRCM and master modules.
- Slave idle protocol: Clock-management protocol between the PRCM and slave modules.

#### 8.1.3.2.1 Master Standby Protocol

Master standby protocol is used to indicate that a master module must initiate a transaction on the device interconnect and requests specific (functional and interface) clocks for the purpose. The PRCM module ensures that the required clocks are active when the master module requests the PRCM module to enable them. This is called a module wake-up transition and the module is said to be functional after this transition completes. Similarly, when the master module no longer requires the clocks, it informs the PRCM module, which can then gate the clocks to the module. The master module is then said to be in standby mode. Although the protocol is completely hardware-controlled, software must configure the clock-management behavior for the module. This is done by setting the module register bit field <Module>\_SYSCONFIG.MIDLEMODE or <Module>\_SYSCONFIG.STANDBYMODE. The behavior, identified by standby mode values, must be configured.

**Table 8-1. Master Module Standby-Mode Settings**

| Standby Mode Value | Selected Mode | Description   |
|--------------------|---------------|---|
| 0x0                | Force-standby | The module unconditionally asserts the standby request to the PRCM module, regardless of its internal operations. The PRCM module may gate the functional and interface clocks to the module. This mode must be used carefully because it does not prevent the loss of data at the time the clocks are gated. |
| 0x1                | No-standby    | The module never asserts the standby request to the PRCM module. This mode is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient from a power-saving perspective because it never allows the output clocks of the PRCM module to be gated        |

**Table 8-1. Master Module Standby-Mode Settings (continued)**

| Standby Mode Value | Selected Mode                    | Description  |
|--------------------|----------------------------------|--|
| 0x2                | Smart-standby                    | The module asserts the standby request based on its internal activity status. The standby signal is asserted only when all ongoing transactions are complete and the module is idled. The PRCM module can then gate the clocks to the module.  |
| 0x3                | Smart-standbywakeup-capable mode | The module asserts the standby request based on its internal activity status. The standby signal is asserted only when all ongoing transactions are complete and the module is idle. The PRCM module can then gate the clocks to the module. The module may generate (master-related) wake-up events when in STANDBY state. The mode is relevant only if the appropriate module mwakeup output is implemented. |

The standby status of a master module is indicated by the CM\_<Power\_domain>\_<Module>\_CLKCTRL[x]. STBYST bit in the PRCM module.

**Table 8-2. Master Module Standby Status**

| STBYST Bit Value | Description                   |
|------------------|-------------------------------|
| 0x0              | The module is functional.     |
| 0x1              | The module is in standby mode |

#### 8.1.3.2.2 Slave Idle Protocol

This hardware protocol allows the PRCM module to control the state of a slave module. The PRCM module informs the slave module, through assertion of an idle request, when its clocks (interface and functional) can be gated. The slave can then acknowledge the request from the PRCM module and the PRCM module is then allowed to gate the clocks to the module. A slave module is said to be in IDLE state when its clocks are gated by the PRCM module. Similarly, an idled slave module may need to be wakened because of a service request from a master module or as a result of an event (called a wake-up event; for example, interrupt or DMA request) received by the slave module. In this situation the PRCM module enables the clocks to the module and then deasserts the idle request to signal the module to wake up. Although the protocol is completely hardware-controlled, software must configure the clock-management behavior for the slave module. This is done by setting the module register bit field <Module>\_SYSCONFIG. SIDLEMODE or <Module>\_SYSCONFIG. IDLEMODE. The behavior, listed in the Idle Mode Value column, must be configured by software.

**Table 8-3. Module Idle Mode Settings**

| Idle Mode Value | Selected Mode | Description  |
|-----------------|---------------|--|
| 0x0             | Force-idle    | The module unconditionally acknowledges the idle request from the PRCM module, regardless of its internal operations. This mode must be used carefully because it does not prevent the loss of data at the time the clock is switched off.   |
| 0x1             | No-idle       | The module never acknowledges any idle request from the PRCM module. This mode is safe from a module point of view because it ensures that the clocks remain active. However, it is not efficient from a power-saving perspective because it does not allow the PRCM module output clock to be shut off, and thus the power domain to be set to a lower power state. |

**Table 8-3. Module Idle Mode Settings (continued)**

| Idle Mode Value | Selected Mode                  | Description  |
|-----------------|--------------------------------|--|
| 0x2             | Smart-idle                     | The module acknowledges the idle request basing its decision on its internal activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or direct memory access (DMA) requests are processed. This is the best approach to efficient system power management.  |
| 0x3             | Smart-idle wakeup-capable mode | The module acknowledges the idle request basing its decision on its internal wakeup-capable mode activity. Namely, the acknowledge signal is asserted only when all pending transactions, interrupts, or DMA requests are processed. This is the best approach to efficient system power management. The module may generate (IRQ- or DMA-request-related) wake-up events when in IDLE state. The mode is relevant only if the appropriate module wakeup output(s) is implemented. |

The idle status of a slave module is indicated by the CM\_<Powerdomain>\_<Module>\_CLKCTRL[x] IDLEST bit field in the PRCM module.

**Table 8-4. Idle States for a Slave Module**

| IDLEST Bit VALUE | Idle Status    | Description   |
|------------------|----------------|---|
| 0x0              | Functional     | The module is fully functional. The interface and functional clocks are active.                             |
| 0x1              | In transition  | The module is performing a wake-up or a sleep transition.   |
| 0x2              | Interface idle | The module interface clock is idled. The module may remain functional if using a separate functional clock. |
| 0x3              | Full idle      | The module is fully idle. The interface and functional clocks are gated in the module.                      |

For the idle protocol management on the PRCM module side, the behavior of the PRCM module is configured in the CM\_<Power domain>\_<module>\_CLKCTRL[x] MODULEMODE bit field. Based on the configured behavior, the PRCM module asserts the idle request to the module unconditionally (that is, immediately when the software requests).

**Table 8-5. Slave Module Mode Settings in PRCM**

| MODULEMODE Bit VALUE | Selected Mode | Description   |
|----------------------|---------------|---|
| 0x0                  | Disabled      | The PRCM module unconditionally asserts the module idle request. This request applies to the gating of the functional and interface clocks to the module. If acknowledged by the module, the PRCM module can gate all clocks to the module (that is, the module is completely disabled).. |
| 0x1                  | Reserved      | NA  |

**Table 8-5. Slave Module Mode Settings in PRCM (continued)**

| MODULEMODE Bit VALUE | Selected Mode | Description   |
|----------------------|---------------|---|
| 0x2                  | Enabled       | This mode applies to a module when the PRCM module manages its interface and functional clocks. The functional clock to the module remains active unconditionally, while the PRCM module automatically asserts/deasserts the module idle request based on the clock-domain transitions. If acknowledged by the module, the PRCM module can gate only the interface clock to the module. |
| 0x3                  | Reserved      | NA  |

In addition to the IDLE and STANDBY protocol, PRCM offers also the possibility to manage optional clocks, through a direct SW control: "OptFclken" bit from programming register.

**Table 8-6. Module Clock Enabling Condition**

| Clock Enabling   |     |                       |                         |
|--|-----|-----------------------|-------------------------|
| Clock associated with STANDBY protocol                   | AND | Clock Domain is ready |                         |
|  |     | OR                    | MStandby is de-asserted |
|  |     |                       | Mwakeup is asserted     |
| Clock associated with IDLE protocol, as interface clock  | AND | Clock Domain is ready |                         |
|  |     | OR                    | Idle status = FUNCT     |
|  |     |                       | Idle status = TRANS     |
| Clock associated with IDLE protocol, as functional clock | AND | Clock Domain is ready |                         |
|  |     | OR                    | Idle status = FUNCT     |
|  |     |                       | Idle status = IDLE      |
| Optional clock   | AND | Clock domain is ready |                         |
|  |     |                       | OptFclken=Enabled ('1') |

### 8.1.3.3 Clock Domain

A clock domain is a group of modules fed by clock signals controlled by the same clock manager in the PRCM module. By gating the clocks in a clock domain, the clocks to all the modules belonging to that clock domain can be cut to lower their active power consumption (that is, the device is on and the clocks to the modules are dynamically switched to ACTIVE or INACTIVE (GATED) states). Thus, a clock domain allows control of the dynamic power consumption of the device. The device is partitioned into multiple clock domains, and each clock domain is controlled by an associated clock manager within the PRCM module. This allows the PRCM module to individually activate and gate each clock domain of the device.

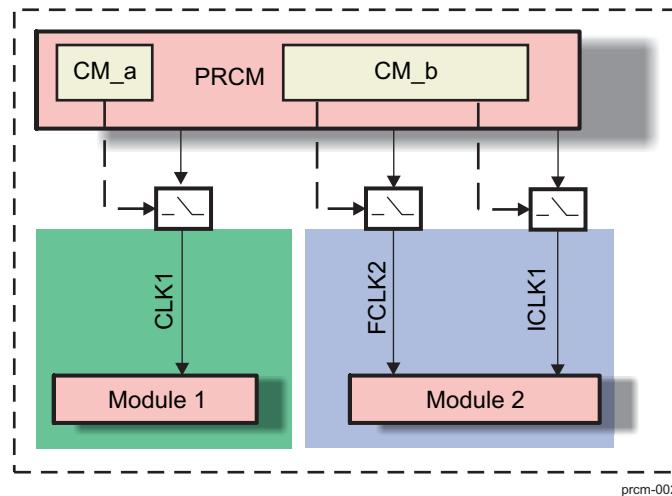
**Figure 8-2. Generic Clock Domain**

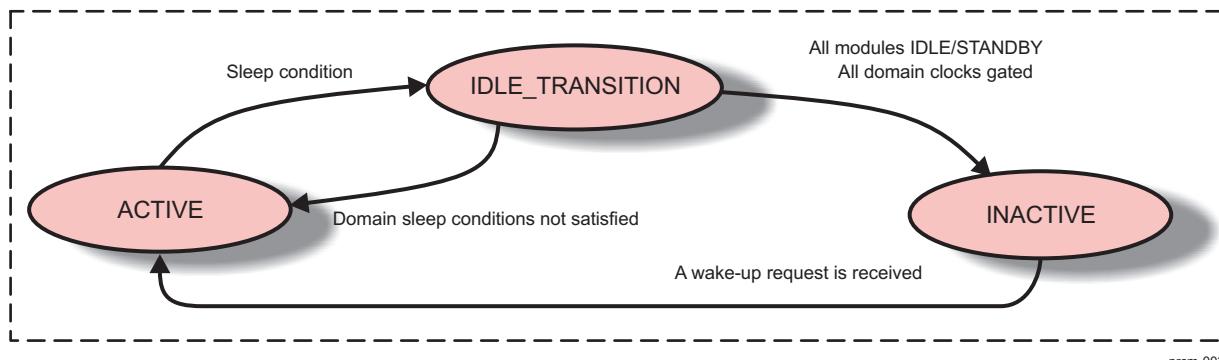
Figure above is an example of two clock managers: CM\_a and CM\_b. Each clock manager manages a clock domain. The clock domain of CM\_b is composed of two clocks: a functional clock (FCLK2) and an interface clock (ICLK1), while the clock domain of CM\_a consists of a clock (CLK1) that is used by the module as a functional and interface clock. The clocks to Module 2 can be gated independently of the clock to Module 1, thus ensuring power savings when Module 2 is not in use. The PRCM module lets software check the status of the clock domain functional clocks. The CM\_<Clock domain>\_CLKSTCTRL[x] CLKACTIVITY\_<FCLK/Clock name\_FCLK> bit in the PRCM module identifies the state of the functional clock(s) within the clock domain. Table shows the possible states of the functional clock.

**Table 8-7. Clock Domain Functional Clock States**

| CLKACTIVITY BIT Value | Status | Description  |
|-----------------------|--------|--|
| 0x0                   | Gated  | The functional clock of the clock domain is inactive |
| 0x1                   | Active | The functional clock of the clock domain is running  |

#### 8.1.3.3.1 Clock Domain-Level Clock Management

The domain clock manager can automatically (that is, based on hardware conditions) and jointly manage the interface clocks within the clock domain. The functional clocks within the clock domain are managed through software settings. A clock domain can switch between three possible states: ACTIVE, IDLE\_TRANSITION, and INACTIVE. Figure 8-3 shows the sleep and wake-up transitions of the clock domain between ACTIVE and INACTIVE states.

**Figure 8-3. Clock Domain State Transitions**

**Table 8-8. Clock Domain States**

| State           | Description   |
|-----------------|---|
| ACTIVE          | <p>Every nondisabled slave module (that is, those whose MODULEMODE value is not set to disabled) is put out of IDLE state.</p> <p>All interface clocks to the nondisabled slave modules in the clock domain are provided. All functional and interface clocks to the active master modules (that is, not in STANDBY) in the clock domain are provided. All enabled optional clocks to the modules in the clock domain are provided.</p> |
| IDLE_TRANSITION | <p>This is a transitory state.</p> <p>Every master module in the clock domain is in STANDBY state.</p> <p>Every idle request to all the slave modules in the clock domain is asserted. The functional clocks to the slave module in enabled state (that is, those whose MODULEMODE values are set to enabled) remain active.</p> <p>All enabled optional clocks to the modules in the clock domain are provided.</p>                    |
| INACTIVE        | <p>All clocks within the clock domain are gated.</p> <p>Every slave module in the clock is in IDLE state and set to disabled.</p> <p>Every slave module in the clock domain (that is, those whose MODULEMODE is set to disabled) is in IDLE state and set to disabled.</p> <p>Every optional functional clock in the clock domain is gated</p>  |

Each clock domain transition behavior is managed by an associated register bit field in the CM\_<Clock domain>\_CLKSTCTRL[x] CLKTRCTRL PRCM module

**Table 8-9. Clock Transition Mode Settings**

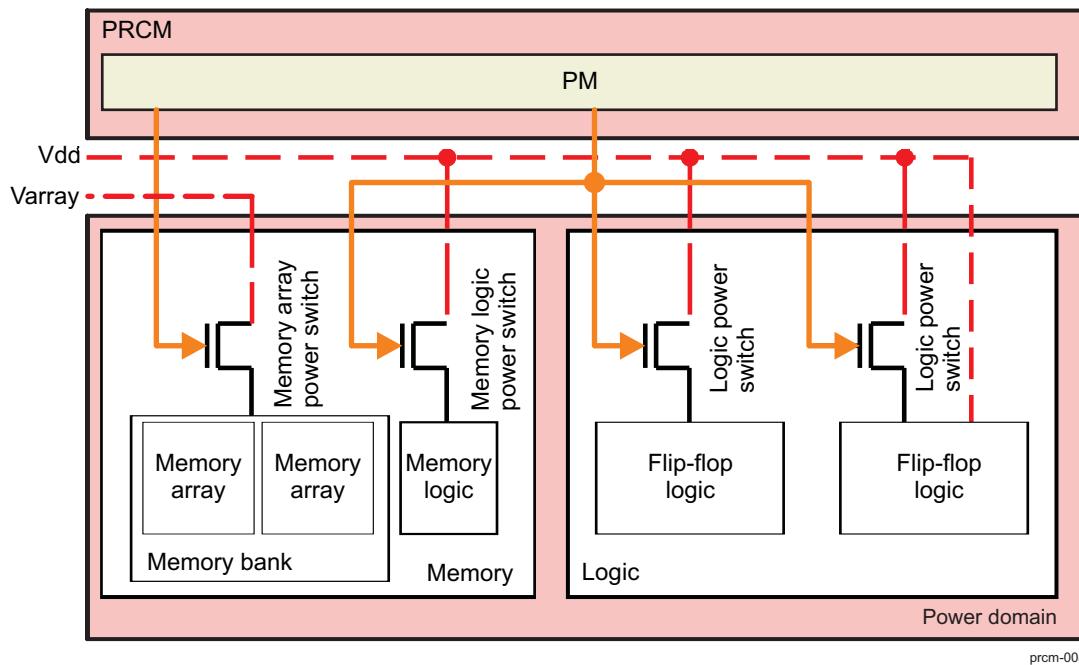
| CLKTRCTRL Bit Value | Selected Mode | Description   |
|---------------------|---------------|---|
| 0x0                 | NO_SLEEP      | Sleep transition cannot be initiated. Wakeup transition may however occur.  |
| 0x1                 | SW_SLEEP      | A software-forced sleep transition. The transition is initiated when the associated hardware conditions are satisfied |
| 0x2                 | SW_WKUP       | A software-forced clock domain wake-up transition is initiated  |
| 0x3                 | Reserved      | NA  |

### 8.1.4 Power Management

The PRCM module manages the switching on and off of the power supply to the device modules. To minimize device power consumption, the power to the modules can be switched off when they are not in use. Independent power control of sections of the device allows the PRCM module to turn on and off specific sections of the device without affecting the others.

#### 8.1.4.1 Power Domain

A power domain is a section (that is, a group of modules) of the device with an independent and dedicated power manager (see Figure). A power domain can be turned on and off without affecting the other parts of the device.

**Figure 8-4. Generic Power Domain Architecture**

To minimize device power consumption, the modules are grouped into power domains. A power domain can be split into a logic area and a memory area.

**Table 8-10. States of a Memory Area in a Power Domain**

| State | Description                                      |
|-------|--|
| ON    | The memory array is powered and fully functional |
| OFF   | The memory array is powered down                 |

**Table 8-11. States of a Logic Area in a Power Domain**

| State | Description   |
|-------|---|
| ON    | Logic is fully powered                                    |
| OFF   | Logic power switches are off. All the logic (DFF) is lost |

#### 8.1.4.2 Power Domain Management

The power manager associated with each power domain is assigned the task of managing the domain power transitions. It ensures that all hardware conditions are satisfied before it can initiate a power domain transition from a source to a target power state.

**Table 8-12. Power Domain Control and Status Registers**

| Register/Bit Field                             | Type    | Description  |
|--|---------|--|
| PM_<Power domain>_PWRSTCTRL[1:0]<br>POWERSTATE | Control | Selects the target power state of the power domain among OFF, ON, or RETENTION.          |
| PM_<Power domain>_PWRSTST[1:0]<br>POWERSTATEST | Status  | Identifies the current state of the power domain. It can be OFF, ON, or RETENTION.       |
| PM_<Power domain>_PWRSTST[2]<br>LOGICSTATEST   | Status  | Identifies the current state of the logic area in the power domain. It can be OFF or ON. |

**Table 8-12. Power Domain Control and Status Registers (continued)**

| Register/Bit Field                           | Type   | Description  |
|--|--------|--|
| PM_<Power domain>_PWRSTST[5:4]<br>MEMSTATEST | Status | Identifies the current state of the memory area in the power domain. It can be OFF, ON, or RETENTION |

#### 8.1.4.2.1 Power-Management Techniques

The following section describes the state-of-the-art power-management techniques supported by the device.

##### 8.1.4.2.1.1 Adaptive Voltage Scaling

AVS is a power-management technique based in Smart Reflex that is used for automatic control of the operating voltages of the device to reduce active power consumption. With Smart Reflex, power-supply voltage is adapted to silicon performance, either statically (based on performance points predefined in the manufacturing process of a given device) or dynamically (based on the temperature-induced real-time performance of the device). A comparison of these predefined performance points to the real-time on-chip measured performance determines whether to raise or lower the power-supply voltage. AVS achieves the optimal performance/power trade-off for all devices across the technology process spectrum and across temperature variation. The device voltage is automatically adapted to maintain performance of the device

#### 8.1.4.3 Power Modes

The following power modes are for easy reference applicable for typical use cases. The design does not restrict these to be the only modes. In order of the lowest power to the highest power, these modes are named RTC-Only, DeepSleep0, DeepSleep1, DeepSleep2, Standby and Active. All voltage supplies must be maintained for the each of the Deep Sleep, Standby and Active modes. In Active mode, all power domains are ON. In RTC mode, only the supplies to the RTC subsystem are supplied.

The contents of SDRAM are preserved in any of the Deep Sleep/Standby modes. This is done by placing SDRAM in self-refresh prior to entering Deep Sleep.

**Table 8-13. Typical Power Modes**

| Use Case   | Power Mode | Application state   | Power Domain States& Supply Voltages  |
|------------|------------|---|---|
| RTC-Only   |            | Only RTC voltage domain is alive.<br>Optionally, SDRAM can be kept in self-refresh which will reduce the boot time.   | VDD_MPU = 0v<br>VDD_CORE = 0v<br>VDDS_RTC supply is active.<br>Optionally, IO supplies can be left ON   |
| DeepSleep0 |            | PD_PER peripheral & CortexA8 / MPU register information will be lost.<br>On-chip peripheral register (context) information of PD_PER domain needs to be saved by application to SDRAM before entering this mode.<br>SDRAM is in self-refresh. For wakeup, boot ROM executes and branches to peripheral context restore followed by system resume. | Master Oscillator = OFF<br>VDD_MPU = 0.95v<br>VDD_CORE = 0.95v<br>PD_WKUP = ON<br>PD_MPU = OFF<br>PD_PER = OFF<br>PD_GFX = OFF<br>All IO supplies & RTC supplies are ON |
| DeepSleep1 |            | On-chip peripheral registers are preserved.<br>CortexA8 context/registers are lost and hence application needs to save them to MPU Subsystem or L3 OCMC RAM or SDRAM before entering DeepSleep.<br>SDRAM is in self-refresh.<br>For wakeup, boot ROM executes and branch to system resume.  | Master Oscillator = OFF<br>VDD_MPU = 0.95v<br>VDD_CORE = 0.95v<br>PD_WKUP = ON<br>PD_MPU = OFF<br>PD_PER = ON<br>PD_GFX = OFF<br>All IO supplies & RTC supplies are ON  |

**Table 8-13. Typical Power Modes (continued)**

| Use Case   | Power Mode | Application state  | Power Domain States & Supply Voltages   |
|------------|------------|--|---|
| DeepSleep2 |            | Everything is preserved including SDRAM.   | Master Oscillator = OFF<br>VDD_MPU = 0.95v<br>VDD_CORE = 0.95v<br>PD_WKUP = ON<br>PD_MPU = ON<br>PD_PER = ON<br>PD_GFX = OFF<br>All IO supplies & RTC supplies are ON   |
| Standby    |            | Everything is preserved including SDRAM.<br>Only required module clocks are enabled, rest are clock gated. | Master Oscillator = ON<br>One PLL is ON.<br>VDD_MPU = 0.95v<br>VDD_CORE = OPP50<br>PD_WKUP = ON<br>PD_MPU = ON<br>PD_PER = ON<br>PD_GFX = OFF<br>All IO supplies & RTC supplies are ON<br>If more PLLs /power domains/ modules are required, the power will increase accordingly. |
| Active     |            | All Features   | Master Oscillator = ON<br>One or more PLLs are ON<br>VDD_MPU = ON<br>VDD_CORE = ON<br>PD_WKUP = ON<br>PD_MPU = ON<br>PD_PER = ON<br>PD_GFX = ON   |

#### 8.1.4.3.1 Active

In Active mode, the supply to all voltage rails must be maintained. All power domains come up in ON state and the device is fully functional.

#### 8.1.4.3.2 Standby

In Standby mode, the supply to all voltage rails must be maintained. The power consumption can be minimized if the supply voltage placed at its minimum operating voltage. Using Software the required PLL's are made active depending upon wakeup or use case requirements and the rest are placed in low power bypass modes. All non-essential IP blocks are powered down. A peripheral interrupt to CortexM3, will cause it to un-clock gate the A8 after which A8 services the pending interrupt. SDRAM contents are preserved. This is useful for quick standby/resume kind of application.

#### 8.1.4.3.3 DeepSleep0

In DeepSleep0 mode, the supply to all voltage rails are maintained, only the PD\_RTC and PD\_WKUP power domains are ON. The master crystal oscillator is disabled. PD\_MPU is OFF. The contents of OCMC RAM are preserved. Additionally the contents of the SDRAM are preserved by placing the SDRAM in self-refresh. Activity on wake up peripherals via wake up events enables the master crystal oscillator using the oscillator control circuit. Also, the wake up events interrupt CortexM3.

#### 8.1.4.3.4 Deepsleep1

In DeepSleep1 mode, the supply to all voltage rails are maintained, the PD\_RTC, PD\_WKUP & PD\_PER power domains are ON. The PD\_GFX power domain is OFF (but could be optionally ON at the cost of more power). The master crystal oscillator is disabled. The contents of internal SRAM are preserved. Additionally the contents of the SDRAM are preserved by placing the SDRAM in self-refresh. Activity on wake up peripherals via wake up events enables the master crystal oscillator using the oscillator control circuit. Also, the wake up events interrupt CortexM3.

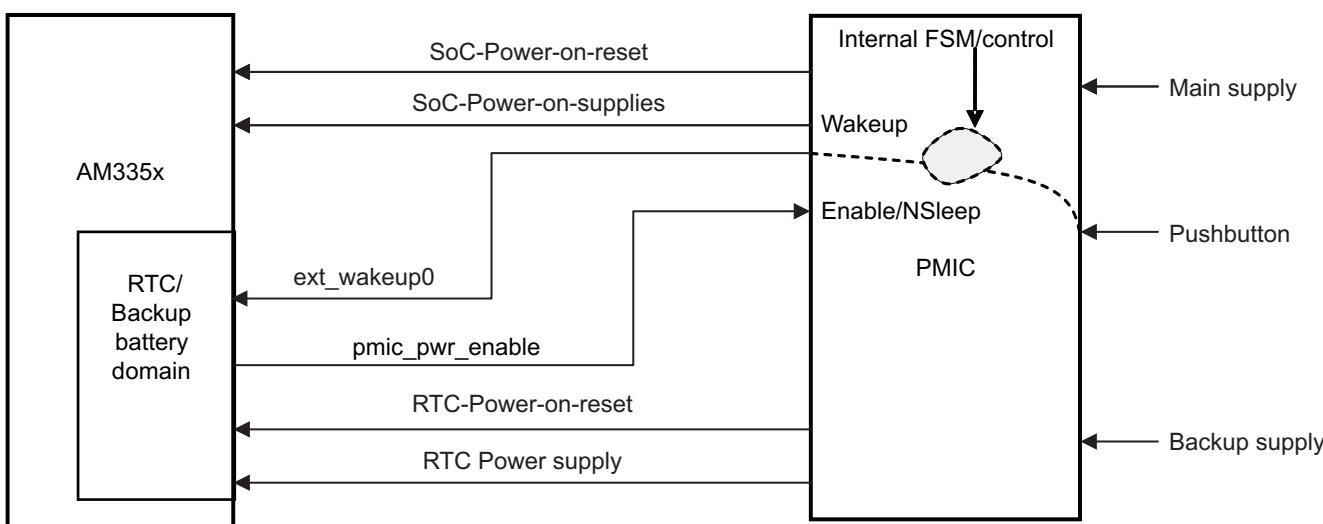
#### 8.1.4.3.5 Deepsleep2

In DeepSleep2 mode, the supply to all voltage rails are maintained, all power domains are ON. The PD\_GFX power domain is OFF (but could be optionally ON at the cost of more power). The master crystal oscillator is disabled. The contents of internal SRAM are preserved. Additionally the contents of the SDRAM are preserved by placing the SDRAM in self-refresh. Activity on wake up peripherals via wake events enables the master crystal oscillator using the oscillator control circuit. Also, the wake events interrupt CortexM3.

#### 8.1.4.3.6 RTC-Only

The following diagram gives a high level view of system which implements the RTC-only mode.

**Figure 8-5. High Level System View for RTC-only Mode**



If RTC-only mode is not required or implemented by the end application, RTC power supply (CAP\_VDD\_RTC) may be connected to VDD\_CORE or a separate 1.1-V power source. In addition, the RTC power-on reset (RTC\_PWRONRSTn) can be connected to the SoC power-on reset (PWRONRSTn) if their power domains are at the same voltage level (i.e., VDDSHV6 is the same voltage as VDDS\_RTC).

The RTC battery backup domain consists of RTCSS (subsystem), a dedicated on-chip 32.768 Hz crystal oscillator and I/Os associated with RTCSS — pmic\_power\_en and ext\_wakeup.

In RTC-only mode, only the RTC power supply is expected to be ON. All the remaining supplies are expected to be OFF. For SDRAM to be in self-refresh during this mode, the CKE pin of the DDR interface needs to have a pulldown on the board.

#### 8.1.4.4 Main Oscillator Control During Deep Sleep

The Deepsleep oscillator circuit is used to control the main oscillator by disabling it during deep sleep and enabling during active/wakeup. By default during reset, the oscillator is enabled and the oscillator control circuit comes up disabled (in-active). In order to activate the oscillator control circuit for deepsleep, DSENABLE bit of DEEPSLEEP\_CTRL register must set. Once this is set and whenever wake M3 enters standby, the oscillator control will disable the oscillator causing the clock to be shut OFF. Any async event from the wakeup sources will cause the oscillator control to re-enable the oscillator after a period of DSCOUNT configured in DEEPSLEEP\_CTRL register.

#### 8.1.4.5 Wakeup Sources/Events

Following events will wake up the device from Deep sleep(low power) modes. These are part of the Wakeup Power domain and remain always ON.

- GPIO0 bank
- dmtimer1\_1ms (timer based wakeup)
- USB2PHY (USB resume signaling from suspend) – Both USB ports supported.
- TSC (touch screen controller, ADC monitor functions )
- UART0 (Infra-red support)
- I2C0

These wake events apply on any of the deep sleep modes and standby mode.

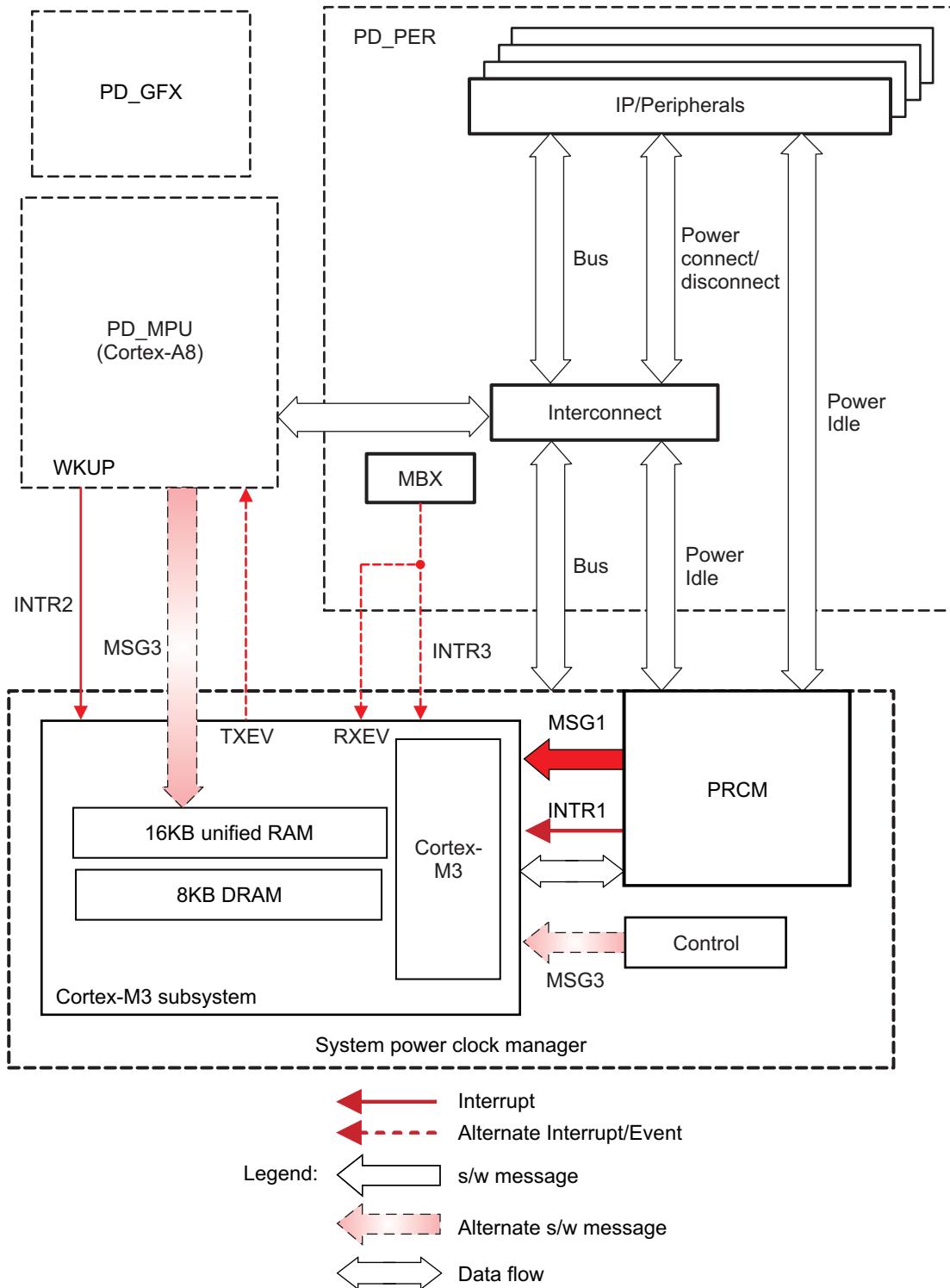
#### 8.1.4.6 Functional Sequencing for Power Management with Cortex M3

The AM335x device contains a dedicated Cortex M3 processor to handle the power management transitions. It is part of the Wake up Power domain (PD\_WKUP). Implementing the Power modes are part of the MPU and Cortex A8 processors.

The power management sequence kicks off with Cortex A8 MPU executing a WFI instruction with the following steps:

1. During Active power mode, the Cortex A8 MPU executes a WFI instruction to enter IDLE mode.
2. Cortex M3 gets an interrupt and gets active, It powers down the MPU power domain( if required).
3. Registers interrupt for the Wake up peripheral(which is listed in Wake up sources in previous section).
4. Executes WFI and goes into idle state.
5. The wake up event triggers an interrupt to Cortex M3 system and it wakes up the Cortex A8 MPU.

Generally, A8 and CortexM3 are not expected to be active at the same time CortexM3 along with PRCM is the power manager primarily for PD\_MPU and PD\_PER. Other power domains (e.g., PD\_GFX) may be handled directly using Cortex A8 MPU software. [Figure 8-6](#) gives a system level view of the Power management system between Cortex A8 MPU and Cortex M3.

**Figure 8-6. System Level View of Power Management of Cortex A8 MPU and Cortex M3**

**Table 8-14. M3 Interrupts 1–3**

| Interrupt Name       | Interrupt Number in M3 | Explanation   |
|----------------------|------------------------|---|
| INTR1 (PRCM_M3_IRQ1) | 16                     | Generated by PRCM when MPU power domain is clock gated. |
| INTR2 (PRCM_M3_IRQ2) | 34                     | Generated by wakeup events                              |

**Table 8-14. M3 Interrupts 1–3 (continued)**

| Interrupt Name | Interrupt Number in M3 | Explanation             |
|----------------|------------------------|-------------------------|
| INTR3 (MBINTO) | 31                     | Mailbox interrupt to M3 |

#### 8.1.4.6.1 Periodic Idling of Cortex A8 MPU

To implement Cortex A8 MPU periodic ON/OFF in the use case, the control flow could be implemented according to the following steps:

1. Cortex A8 MPU executes WFI instruction
2. Any peripheral interrupt in any of the next steps will trigger a wake interrupt to CortexM3 via MPU Subsystem's WKUP signal (INTR2 shown on the diagram). Cortex M3 powers down the MPU(PD\_MPUMPU)
3. On receiving an interrupt Cortex M3 switches ON the MPU power domain by turning on PD\_MPUMPU
4. Cortex M3 goes into idle mode using WFI instruction

#### 8.1.4.6.2 Sleep Sequencing

This section gives the system level guidelines for sleep sequencing. The guidelines can serve as an example for implementing the sleep mode sequencing. The user can opt to implement a sequence with certain steps interchanged between the MPU and Cortex M3 processor.

1. Application saves context of peripherals to memories supporting retention and DDR – this step is only required for DeepSleep0.
2. MPU OCMC\_RAM remains in retention
3. Unused power domains are turned OFF - program clock/power domains PWRSTCTRL, save contexts etc
4. Software populates L3\_OCMC\_RAM for wakeup restoration viz Save EMIF settings, public restoration pointers, etc.
5. Execute WFI from SRAM
6. Any peripheral interrupt will trigger a wake interrupt to CortexM3 via Cortex A8 MPU's WKUP signal (INTR2 shown on the diagram).
7. After MPU power domain is clock gated PRCM will provide an interrupt to CortexM3 (using INTR1 shown in the block diagram)
8. CortexM3 starts execution as follows:
  - CortexM3 configures PLL / HSDIVIDER power down States (bypass modes). During Idle bypass mode (low power mode) the PLL / HSDIVIDER goes into bypass and provides the clock outputs directly from the oscillator clock.
  - CortexM3 disables IP/Peripheral/Interconnects based on MSG1/MSG2/MSG3 sconnect/disconnect/Idle protocol kicks off and completes
  - PRCM turns OFF power domains based on MSG1/MSG2/MSG3 Only applicable for DeepSleep0 & DeepSleep1.
  - CortexM3 lowers the PMIC voltage to OPP50 voltage
  - CortexM3 configures I/O PADs for lowest power as dictated by CortexA8 using MSG2/MSG3
  - CortexM3 goes into WFI
9. Hardware oscillator control circuit disables the oscillator once CortexM3 goes into WFI

#### 8.1.4.6.3 Wakeup Sequencing

This section gives the guidelines for Wakeup sequencing.

1. One of the wakeup event triggers (which was configured during the sleep sequencing) will initiate a wakeup sequence
2. The wake up event will switch on the oscillator (if it was configured to go OFF during sleep)

3. The wake up event will also trigger interrupt to Cortex M3
4. On the wakeup event due to interrupt Cortex M3 execute the following
  - Restore the voltages to normal Operating voltage
  - Enable PLL locking
  - Cortex M3 will switch ON the power domains and/or enable clocks for PD\_PER
  - Cortex M3 will switch ON the power domains and/or enable clocks for PD\_MPU
  - Executes WFI
5. Cortex A8 MPU starts executing from ROM reset vector
6. Restore the application context(only for Deep sleep 0)

### **8.1.5 PRCM Module Overview**

The PRCM is structured using the architectural concepts presented in the 5000x Power Management Framework. This framework provides:

A set of modular, re-usable FSM blocks to be assembled into the full clock and power management mechanism. A register set and associated programming model. Functional sub-block definitions for clock management, power management, system clock source generation, and master clock generation.

The device supports an enhanced power management scheme based on four functional power domains:

#### **Generic Domains**

- WAKEUP
- MPU
- PER
- RTC

The PRCM provides the following functional features:

- Software configurable for direct, automatic, or a combination thereof, functional power domain state transition control
- Device power-up sequence control
- Device sleep / wake-up sequence control
- Centralized reset generation and management
- Centralized clock generation and management

The PRCM modules implement these general functional interfaces:

- OCP configuration ports
- Direct interface to device boundary
- Power switch control signals
- Device control signals
- Clocks control signals
- Resets signals
- A set of power management protocol signals for each module to control and monitor standby, idle and wake-up modes (CM and PRM)
- Emulation signals

#### **8.1.5.1 Interface Descriptions**

This section lists and shortly describes the different interfaces that allow PRCM to communicate with other modules or external devices.

### 8.1.5.1.1 OCP Interfaces

The PRCM has 1 target OCP interfaces, compliant with respect to the OCP/IP2 standard. The OCP port, for the PRCM module is used to control power, reset and wake-up Management.

### 8.1.5.1.2 OCP Slave Interfaces

PRCM implements a 32-bit OCP target interface compliant to the OCP/IP2.0 standard.

### 8.1.5.1.3 Power Control Interface

The Device does has power domain switches over the device, this interface provides PRCM control over power domain switches and receives responses from the power domains which indicate the switch status. It also controls the isolation signals. The control for power domain switches will be latched in PRCM Status Registers

### 8.1.5.1.4 Device Control Interface

This interface provides PRM management of several device-level features which are not specific to any single power domain. This PRM interface controls signals to/from the device for global control:

- Device Type coding
- IOs isolation control

### 8.1.5.1.5 Clocks Interface

This interface gathers all clock inputs and outputs managed by PRCM modules.

### 8.1.5.1.6 Resets Interface

This interface gathers all resets inputs and outputs managed by PRCM module.

### 8.1.5.1.7 Modules Power Management Control Interface

Modules or subsystems in the device are split over 2 categories:

- Initiator: an initiator is a module able to generate traffic on the device interconnects (typically: processors, MMU, EDMA).
- Target: a target is a module that cannot generate traffic on the device interconnects, but that can generate interrupts or DMA request to the system (typically: peripherals). PRCM handles a power management handshake protocol with each module or sub-system. This protocol allows performing proper clock and power transition taking into account each module activity or state.

### 8.1.5.1.8 Initiator Modules Interface

PRCM module handle all initiator modules power management interfaces: MStandby signal MWait signal

### 8.1.5.1.9 Targets Modules Interface

PRCM module handle all target modules power management interfaces: SIdleReq signal SIdleAck signal FCLKEN signal

**Note:** USB Support for SWakeUp

## 8.1.6 Clock Generation and Management

PRCM provides a centralized control for the generation, distribution and gating of most clocks in the device. PRCM gathers external clocks and internally generated clocks for distribution to the other modules in the device. PRCM manages the system clock generation

### 8.1.6.1 Terminology

The PRCM produces 2 types of clock:

**Interface clocks:** these clocks primarily provide clocking for the system interconnect modules and the portions of device's functional modules which interface to the system interconnect modules. In most cases, the interface clock supplies the functional module's system interconnect interface and registers. For some modules, interface clock is also used as functional clock. In this document, interface clocks are represented by blue lines.

**Functional clock:** this clock supplies the functional part of a module or a sub-system. In some cases, a module or a subsystem may require several functional clocks: 1 or several main functional clock(s), 1 or several optional clock(s). A module needs its main clock(s) to be operational. Optional clocks are used for specific features and can be shutdown without stopping the module

#### 8.1.6.2 Clock Structure

To generate high-frequency clocks, the device supports multiple on-chip DPLLs controlled directly by the PRCM module. They are of two types of PLLs, referred to ADPLLs and ADPLLJ throughout this document.

The ADPLLs module is used for the Core, Display, ARM Subsystem and DDR PLLs

The ADPLLJ module is used for the peripheral functional clocks

The device has two reference clocks which are generated by on-chip oscillators or externally. These are for the main clock tree and RTC block, respectively.

In the case of an external oscillator, a clock can directly be connected to XTALIN pin and the oscillator will be put in bypass mode. The 32-Khz crystal oscillator is controlled and configurable by RTC IP. This device also contains an on-chip RC oscillator. This oscillator is not configurable but may be enabled or disabled through the Control Module *RCOSC\_CTRL* register.

The main oscillator on the device (see [Chapter 26, Initialization](#), for possible frequencies) produces the master high frequency clock CLK\_M\_OSC.

#### 8.1.6.3 ADPLLs

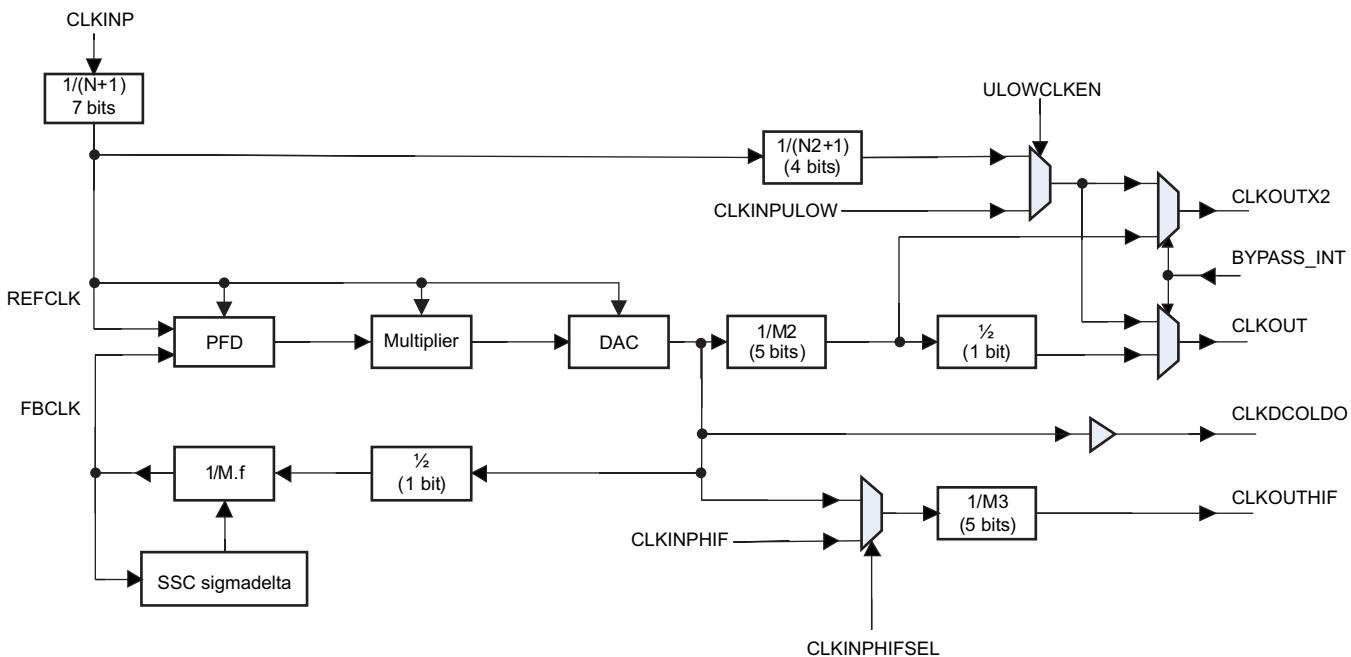
The ADPLLs is a high resolution frequency synthesizer PLL with built in level shifters which allows the generation of PLL locked frequencies up to 2 GHz. ADPLLs has a predivide feature which allows user to divide, for instance, a 24- or 26-MHz reference clock to 1-MHz and then multiply up to 2-GHz maximum.

All PLLs will come-up in bypass mode at reset. SW needs to program all the pll settings appropriately and then wait for PLL to be locked. For more details, see the configuration procedure for each PLL.

The following PLLs are:

- MPU PLL
- Core PLL
- Display PLL
- DDR PLL

Figure 8-7. ADPLLS



The ADPLLS has three input clocks:

- CLKINP: Reference input clock
- CLKINPULOW: Low frequency input clock for bypass mode only.
- CLKINPHIF: High Frequency Input Clock for post-divider M3

The ADPLLS has four output clocks:

- CLKOUTHIF: High Frequency Output Clock from Post divider M3
- CLKOUTX2: Secondary 2x Output
- CLKOUT: Primary output clock
- CLKDCOLDO: Oscillator (DCO) output clock with no bypass

The DPLL has two internal clocks:

- REFCLK (Internal reference clock): This is generated by dividing the input clock CLKINP by the programmed value N+1. The entire loop of the PLL runs on the REFCLK.  
Here,  $\text{REFCLK} = \text{CLKINP}/(\text{N}+1)$ .
- BCLK: Bus clock which is used for programming the various settings using registers

The ADPLLS lock frequency is defined as follows:  $f_{\text{DPLL}} = (\text{M} * \text{CLKINP})/(\text{N}+1)$

### 8.1.6.3.1 Clock Functions

Table 8-15. Output Clocks in Locked Condition

| Pin Name            | Frequency  | Comments                  |
|---------------------|--|---------------------------|
| <b>REGM4XEN='0'</b> |  |                           |
| CLKOUT              | $[\text{M} / (\text{N}+1)] * \text{CLKINP} * [1/\text{M}_2]$     |                           |
| CLKOUTX2            | $2 * [\text{M} / (\text{N}+1)] * \text{CLKINP} * [1/\text{M}_2]$ |                           |
| CLKDCOLDO           | $2 * [\text{M} / (\text{N}+1)] * \text{CLKINP}$                  |                           |
| CLKOUTHIF           | $\text{CLKINPHIF} / \text{M}_3$                                  | $\text{CLKINPHIFSEL}='1'$ |
|                     | $2 * [\text{M} / (\text{N}+1)] * \text{CLKINP} * [1/\text{M}_3]$ | $\text{CLKINPHIFSEL}='0'$ |
| <b>REGM4XEN='1'</b> |  |                           |

**Table 8-15. Output Clocks in Locked Condition (continued)**

| Pin Name  | Frequency                          | Comments         |
|-----------|------------------------------------|------------------|
| CLKOUT    | [4M / (N+1)] * CLKINP * [1/M2]     |                  |
| CLKOUTX2  | 2 * [4M / (N+1)] * CLKINP * [1/M2] |                  |
| CLKDCOLDO | 2 * [4M / (N+1)] * CLKINP          |                  |
| CLKOUTHIF | CLKINPHIF / M3                     | CLKINPHIFSEL='1' |
|           | 2 * [4M / (N+1)] * CLKINP * [1/M3] | CLKINPHIFSEL='0' |

**Table 8-16. Output Clocks Before Lock and During Relock Modes**

| Pin Name  | Frequency       | Comments      |
|-----------|-----------------|---------------|
| CLKOUT    | CLKINP / (N2+1) | ULOWCLKEN='0' |
|           | CLKINPULOW      | ULOWCLKEN='1' |
| CLKOUTX2  | CLKINP / (N2+1) | ULOWCLKEN='0' |
|           | CLKINPULOW      | ULOWCLKEN='1' |
| CLKDCOLDO | Low             |               |
| CLKOUTHIF | CLKINPHIF/M3    | ULOWCLKEN='1' |
|           | Low             | ULOWCLKEN='0' |

**Note:** Since M3 divider is running on the internal LDO domain, in the case when CLKINPHIFSEL='1', CLKOUTHIF could be active only when internal LDO is ON. Hence, whenever LDOPWDN goes low to high to powerdown LDO (happens when TINITZ activated / when entering slow relock bypass mode), output CLKOUTHIF will glitch and stop. To avoid this glitch, it is recommended to gate CLKOUTHIF using control CLKOUTHIFEN before asserting TINITZ / entering any slow relock bypass mode Frequency Range (MHz)

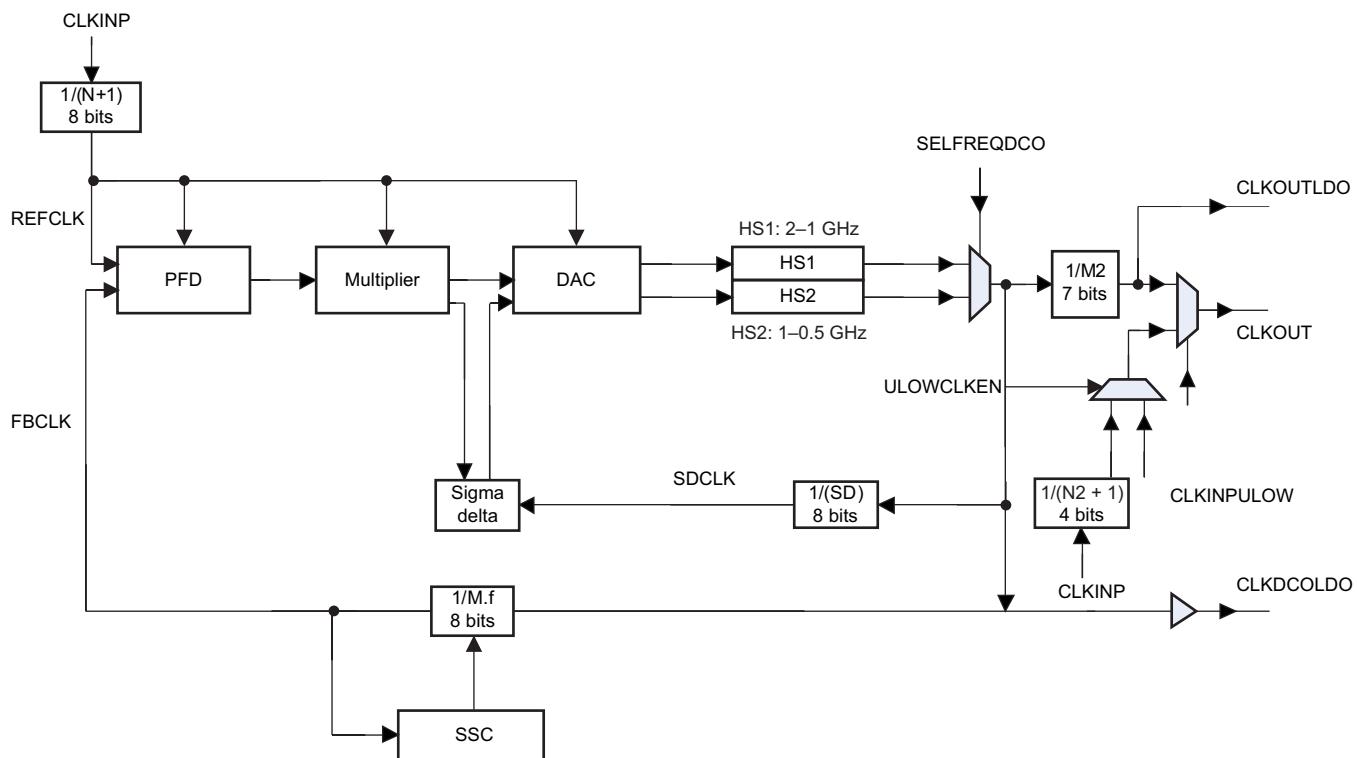
See the device-specific data manual for details on operating performance points (OPPs) supported by your device.

#### 8.1.6.4 ADPLLJ (Low Jitter PLL)

The ADPLLJ is a low jitter PLL with a 2-GHz maximum output. ADPLLJ has a predivide feature which allows user to divide, for instance, a 24-MHz or 26-MHz reference clock to 1 MHz and then multiply up to 2 GHz maximum.

All PLLs will come-up in bypass mode at reset. SW needs to program all the PLL settings appropriately and then wait for PLL to be locked. For more details, see the configuration procedure for each PLL.

Figure 8-8. Basic Structure of the ADPLLJ



The Peripheral PLL belongs to type ADPLLJ:

The DPLL has two input clocks:

- CLKINP: Reference input clock
- CLKINPULOW: Bypass input clock.

The DPLL has two internal clocks:

- REFCLK (Internal reference clock): This is generated by dividing the input clock CLKINP by the programmed value N+1. The entire loop of the PLL runs on the REFCLK.  
Here,  $\text{REFCLK} = \text{CLKINP}/(\text{N}+1)$ .
- CLKDCO (Internal Oscillator clock.): This is the raw clock directly out of the digitally controlled oscillator (DCO) before the post-divider. The PLL output clock is synthesized by an internal oscillator which is phase locked to the refclk. There are two oscillators built within ADPLLJ. The oscillators are user selectable based on the synthesized output clock frequency requirement. In locked condition,  $\text{CLKDCO} = \text{CLKINP} * [\text{M}/(\text{N}+1)]$ .

The ADPLLJ lock frequency is defined as follows:  $f_{\text{DPLL}} = (\text{M} * \text{CLKINP})/(\text{N}+1)$

The DPLL has three external output clocks:

- CLKOUTLDO: Primary output clock in VDDLDOOUT domain. Bypass option not available on this output.

$$\text{CLKOUTLDO} = (\text{M} / (\text{N}+1)) * \text{CLKINP} * (1/\text{M}2)$$

- CLKOUT:

Primary output clock on digital core domain

$$\text{CLKOUT} = (\text{M} / (\text{N}+1)) * \text{CLKINP} * (1/\text{M}2)$$

- CLKDCOLDO:

Oscillator (DCO) output clock before post-division in VDDLDOOUT domain. Bypass option is not available on this output.

$$\text{CLKDCOLDO} = (\text{M} / (\text{N}+1)) * \text{CLKINP}.$$

All clock outputs of the DPLL can be gated. The Control module provides the DPLL with a clock gating control signal to enable or disable the clock, and the DPLL provides the PRCM module with a clock activity status signal to let the PRCM module hardware know when the clock is effectively running or effectively gated. Output clock gating control for various clockouts:  
**CLKOUTEN/CLKOUTLDOEN/CLKDCOLDOEN.**

#### 8.1.6.4.1 Clock Functions

**Table 8-17. Output Clocks in Locked Condition**

| Pin Name  | Frequency                    |
|-----------|------------------------------|
| CLKOUT    | [M /(N+1)] * CLKINP * [1/M2] |
| CLKOUTLDO | [M /(N+1)] * CLKINP * [1/M2] |
| CLKDCOOUT | [M /(N+1)] * CLKINP          |

**Table 8-18. Output Clocks Before Lock and During Relock Modes**

| Pin Name  | Frequency     | Comments      |
|-----------|---------------|---------------|
| CLKOUT    | CLKINP/(N2+1) | ULOWCLKEN='0' |
|           | CLKINPLOW     | ULOWCLKEN='1' |
| CLKDCOLDO | LOW           |               |
| CLKOUTLDO | LOW           |               |

#### 8.1.6.5 M2 and N2 Change On-the-Fly

The dividers M2 and N2 are designed to change on the fly and provide a glitch-free frequency switch from the old to new frequencies. In other words, they can be changed while the PLL is in a locked condition, without having to switch to bypass mode. A status toggle bit will give an indication if the new divisor was accepted. These dividers can also be changed in bypass mode, and the new divisor value will be reflected on output after the PLL relocks. For more details, see the PLL configuration procedures for each PLL.

#### 8.1.6.6 Spread Spectrum Clocking (SSC)

The module supports spread spectrum clocking (SSC) on its output clocks. SSC is used to spread the spectral peaking of the clock to reduce any electromagnetic interference (EMI) that may be caused due to the clock's fundamental or any of its harmonics. When SSC is enabled the clock's spectrum is spread by the amount of frequency spread, and the attenuation is given by the ratio of the frequency spread ( $\Delta f$ ) and the modulation frequency ( $f_m$ ), i.e.,  $\{[10^{\log_{10}(Df/f_m)}] - 10\}$  dB.

SSC is performed by changing the feedback divider (M) in a triangular pattern. Implying, the frequency of the output clock would vary in a triangular pattern. The frequency of this pattern would be modulation frequency ( $f_m$ ). The peak ( $\Delta M$ ) or the amplitude of the triangular pattern as a percent of M would be equal to the percent of the output frequency spread ( $\Delta f$ ); that is,  $\Delta M/M = \Delta f / f_c$ . Next mark with  $F_{in}$  the frequency of the clock signal at the input of the DPLL. Because it is divided to N+1 before entering the phase detector, so the internal reference frequency is  $F_{ref} = F_{in} / (N + 1)$ .

Assume the central frequency  $f_c$  to be equal to the DPLL output frequency  $F_{out}$ , or  $f_c = F_{out} = (F_{in} / (N + 1)) * (M / M2)$ . Since this is in band modulation for the DPLL, the modulation frequency is required to be within the DPLL's loop bandwidth (lowest BW of  $F_{ref} / 70$ ). A higher modulation frequency would result in lesser spreading in the output clock.

SSC can be enabled/disabled using bit CM\_CLKMODE\_DPLL\_xxx.DPLL\_SSC\_EN (where xxx can be any one of the following DPLLS: MPU, DDR, DISP, CORE, PER). An acknowledge signal CM\_CLKMODE\_DPLL\_xxx.DPLL\_SSC\_ACK notifies the exact start and end of SSC. When SSC\_EN is de-asserted, SSC is disabled only after completion of one full cycle of the triangular pattern given by the modulation frequency. This is done in order to maintain the average frequency.

Modulation frequency ( $f_m$ ) can be programmed as a ratio of  $Fref / 4$ ; that is, the value that needs to be programmed  $ModFreqDivider = Fref / (4*f_m)$ . The ModFreqDivider is split into Mantissa and 2^Exponent ( $ModFreqDivider = ModFreqDividerMantissa * 2^{ModFreqDividerExponent}$ ). The mantissa is controlled by 7-bit signal `ModFreqDividerMantissa` through `CM_SSC_MODFREQDIV_DPLL_xxx.MODFREQDEV_MANTISSA` bit field. The exponent is controlled by 3bit signal `ModFreqDividerExponent` through the `CM_SSC_MODFREQDIV_DPLL_xxx.MODFREQDEV_EXPONENT` bit field.

**Note:** Although the same value of `ModFreqDivider` can be obtained by different combinations of mantissa and exponent values, it is recommended to get the target `ModFreqDivider` by programming maximum mantissa and a minimum exponent. To define the Frequency spread ( $\Delta f$ ),  $\Delta M$  must be controlled as explained previously. To define  $\Delta M$ , the step size of  $M$  for each  $Fref$  during the triangular pattern must be programmed; that is,

$$\begin{aligned} \Delta M &= (2^{ModFreqDividerExponent}) * ModFreqDividerMantissa * DeltaMStep \text{ IF} \\ ModFreqDividerExponent \leq 3 &\Delta M = 8 * ModFreqDividerMantissa * DeltaMStep \text{ IF} \\ ModFreqDividerExponent > 3 & \end{aligned}$$

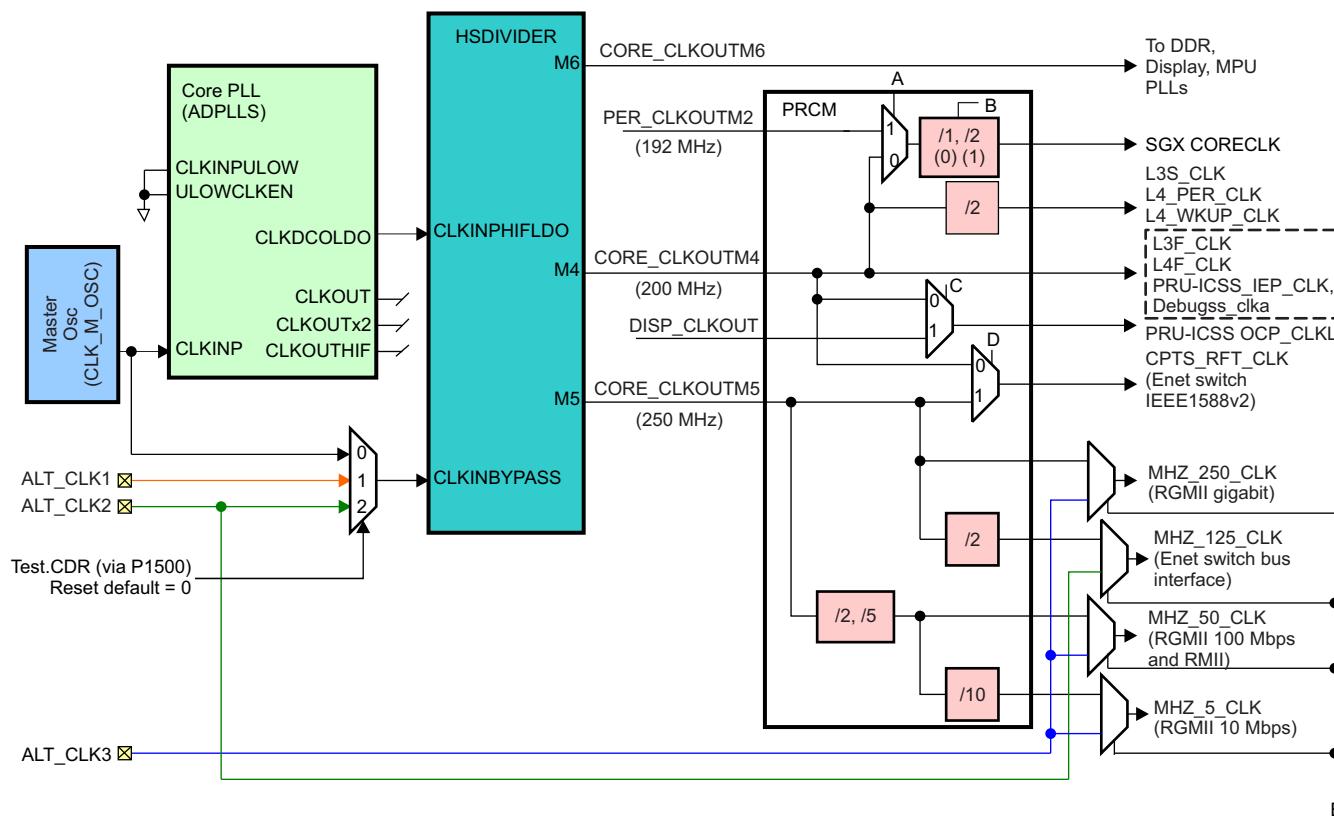
`DeltaMStep` is split into integer part and fractional part. Integer part is controlled by 2-bit signal `DeltaMStepInteger` through the `CM_SSC_DELTAMSTEP_DPLL_xxx.DELTAMSTEP_INTEGER` bit field. Fractional part is controlled by 18-bit signal `DeltaMStepFraction` through the `CM_SSC_DELTAMSTEP_DPLL_xxx.DELTAMSTEP_FRACTION` bit field.

The frequency spread achieved has an overshoot of 20 percent or an inaccuracy of +20 percent. If the `CM_CLKMODE_DPLL.DPLL_SSC_DOWNSPREAD` is set to 1, the frequency spread on lower side is twice the programmed value. The frequency spread on higher side is 0 (except for the overshoot as described previously).

There is restriction of range of  $M$  values. The restriction is  $M - \Delta M \geq 20$ . Also,  $M + \Delta M \leq 2045$ . In case the downspread feature is enabled,  $M - 2 * \Delta M \geq 20$  and  $M \leq 2045$ .

### 8.1.6.7 Core PLL Description

The Core PLL provides the source for a majority of the device infrastructure and peripheral clocks. The Core PLL comprises an ADPLLs with HSDIVIDER and additional dividers and muxes located in the PRCM as shown in [Figure 8-9](#).

**Figure 8-9. Core PLL**


ALT\_CLKs are to be used for internal test purpose and should not be used in functional mode.

**Table 8-19. PLL and Clock Frequencies**

| Mux Select | Register Bit Section 9.2.4.4    |
|------------|---------------------------------|
| A          | PRCM.CLKSEL_GFX_FCLK[1]         |
| B          | PRCM.CLKSEL_GFX_FCLK[0]         |
| C          | PRCM.CLKSEL_PRU-ICSS_OCP_CLK[0] |
| D          | PRCM.CM_CPTS_RFT_CLKSEL[0]      |
| E          | TEST.CDR (via P1500)            |

**Table 8-20** gives the typical PLL and clock frequencies. The HSDIVIDER is used to generate three divided clocks M4, M5 & M6. M4 & M5 are nominally 200 & 250 MHz, respectively.

**Table 8-20. Core PLL Typical Frequencies (MHz)**

| CLOCK  | Source         | Power-On-Reset / HSDIVIDER Bypass |                | OPP100    |            | OPP50 <sup>(1)</sup> |            |
|--|----------------|-----------------------------------|----------------|-----------|------------|----------------------|------------|
|  |                | DIV                               | Freq           | DIV Value | Freq (MHz) | DIV Value            | Freq (MHz) |
| CLKDCOLDO (PLL Lock frequency)   | APLLS          | -                                 | -              | -         | 2000       | -                    | 100        |
| CORE_CLKOUTM4  | HSDIVIDER-M4   | -                                 | Mstr Xtal      | 10        | 200        | 1                    | 100        |
| L3F_CLK, L4F_CLK, PRU-ICSS IEP CLK, DebugSS clka, SGX.MEMCLK, SGX.SYSCLK | CORE_CLKO UTM4 | -                                 | Mstr Xtal      | -         | 200        | -                    | 100        |
| L4_PER, L4_WKUP  | CORE_CLKO UTM4 | 2                                 | Mstr Xtal / 2  | 2         | 100        | 2                    | 50         |
| SGX CORECLK  | CORE_CLKO UTM4 | 1                                 | Mstr Xtal      | 1         | 200        | 1                    | 100        |
|  |                |                                   |                | 2         | 100        | 2                    | 50         |
| CORE_CLKOUTM5  | HSDIVIDER-M5   | -                                 | Mstr Xtal      | 8         | 250        | 1                    | 100        |
| MHZ_250_CLK (Gigabit RGMII)  | CORE_CLKO UTM5 | -                                 | NA             | -         | 250        | -                    | NA         |
| MHZ_125_CLK (Ethernet Switch Bus Clk)                                    | CORE_CLKO UTM5 | 2                                 | Mstr Xtal / 2  | 2         | 125        | 2                    | 50         |
| MHZ_50_CLK (100 mbps RGMII or 10/100 RMII)                               | CORE_CLKO UTM5 | 5                                 | Mstr Xtal / 5  | 5         | 50         | 2                    | 50         |
| MHZ_5_CLK (10 mbps RGMII)  | MHZ_50_CLK     | 10                                | Mstr Xtal / 50 | 10        | 5          | 10                   | 5          |
| CORE_CLKOUTM6  | HSDIVIDER M6   | -                                 | Mstr Xtal      | 4         | 500        | 1                    | 100        |

<sup>(1)</sup> Not all interfaces and peripheral modules are available in OPP50. For more information, see the device specific datasheet.

The ADPLLS module supports two different bypass modes via their internal MNBypass mode and their external Low Power Idle bypass mode. The PLLs are in the MNBypass mode after power-on reset and can be configured by software to enter Low Power Idle bypass mode for power-down.

When the Core PLL is configured in bypass mode, the HSDIVIDER enters bypass mode and the CLKINBYPASS input is driven on the M4, M5, and M6 outputs. CLKINBYPASS defaults to the master oscillator input (typically 24 MHz).

**Table 8-21. Bus Interface Clocks**

|             |  |
|-------------|--|
| L3F_CLK     | SGX530 (MEMCLK & SYSCLK), LCDC, MPU Subsystem, GEMAC Switch (Ethernet), DAP, PRU-ICSS, EMIF, TPTC, TPCC, OCMC RAM, DEBUGSS   |
| L3S_CLK     | USB, TSC, GPMC, MMCHS2, McASP0, McASP1   |
| L4_PER_CLK  | DCAN0, DCAN1<br>DMTIMER2, DMTIMER3, DMTIMER4, DMTIMER5, DMTIMER6, DMTIMER7<br>eCAP/eQEP/ePWM0, eCAP/eQEP/ePWM1, eCAP/eQEP/ePWM2, eFuse<br>ELM, GPIO1, GPIO2, GPIO3<br>I2C1, I2C2, IEEE1500, LCD, Mailbox0<br>McASP0, McASP1<br>MMCHS0, MMCHS1, OCP Watchpoint, SPI0, SPI1, Spinlock<br>UART1, UART2, UART3, UART4, UART5 |
| L4_WKUP_CLK | ADC_TSC, Clock Manager, Control Module<br>DMTIMER0, DMTIMER1_1MS, GPIO0<br>I2C0, M3UMEM, M3DMEM, SmartReflex0, SmartReflex1<br>UART0, WDT1   |

#### 8.1.6.7.1 Core PLL Configuration

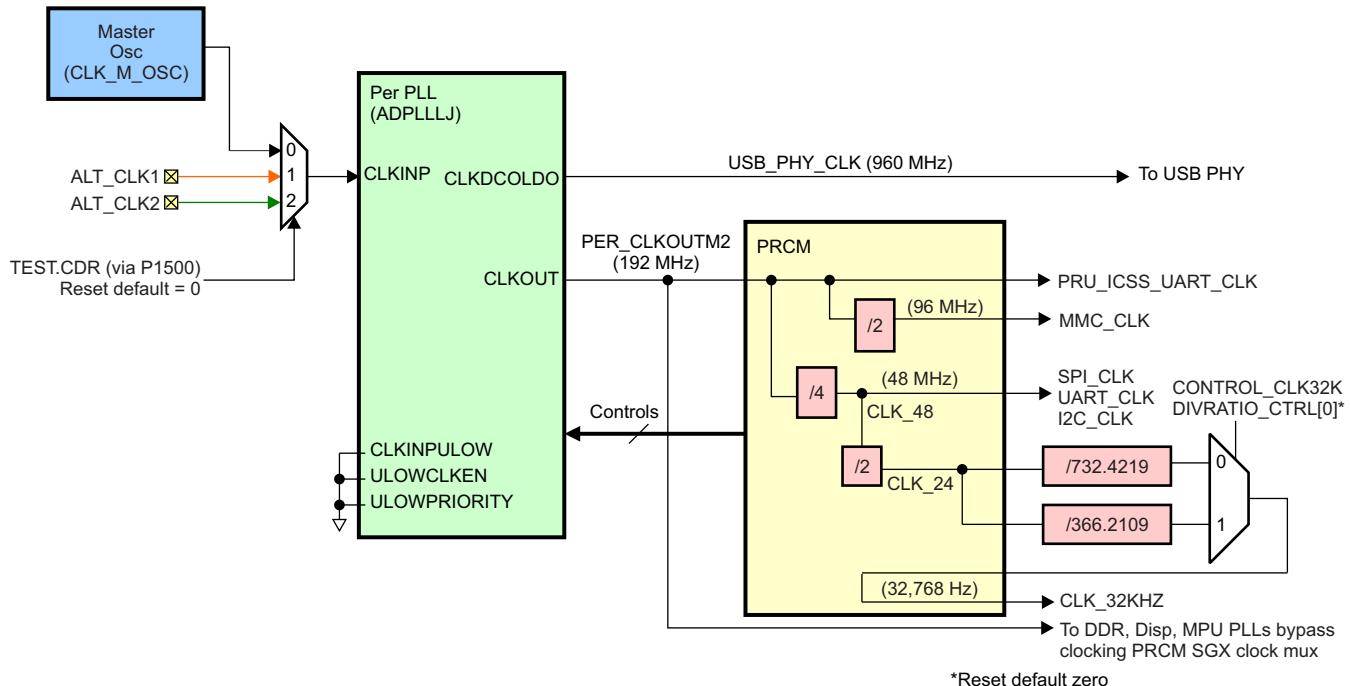
1. Switch PLL to bypass mode by setting CM\_CLKMODE\_DPLL\_CORE.DPLL\_EN to 0x4.
2. Wait for CM\_IDLEST\_DPLL\_CORE.ST\_MN\_BYPASS = 1 to ensure PLL is in bypass (CM\_IDLEST\_DPLL\_CORE.ST\_DPLL\_CLK should also change to 0 to denote the PLL is unlocked).
3. Configure Multiply and Divide values by setting CM\_CLKSEL\_DPLL\_CORE.DPLL\_MULT and DPLL\_DIV to the desired values.
4. Configure M4, M5 and M6 dividers by setting HSDIVIDER\_CLKOUT1\_DIV bits in CM\_DIV\_M4\_DPLL\_CORE, CM\_DIV\_M5\_DPLL\_CORE, and CM\_DIV\_M6\_DPLL\_CORE to the desired values.
5. Switch over to lock mode by setting CM\_CLKMODE\_DPLL\_CORE.DPLL\_EN to 0x7.
6. Wait for CM\_IDLEST\_DPLL\_CORE.ST\_DPLL\_CLK = 1 to ensure PLL is locked (CM\_IDLEST\_DPLL\_CORE.ST\_MN\_BYPASS should also change to 0 to denote the PLL is out of bypass mode).

**Note:** M4, M5, and M6 dividers can also be changed on-the-fly so that there is no need to put the PLL in bypass and back to lock mode. After changing CM\_DIV\_Mx\_DPLL\_CORE.DPLL\_CLKOUT1\_DIV, check CM\_DIV\_Mx\_DPLL\_CORE.DPLL\_HSDIVIDER\_CLKOUT1\_DIVCHACK for a toggle (a change from 0 to 1 or 1 to 0) to see if the change was acknowledged by the PLL.

#### 8.1.6.8 Peripheral PLL Description

The Per PLL provides the source for peripheral functional clocks. The Per PLL comprises an ADPLLJ and additional dividers and muxes located in the PRCM as shown

Figure 8-10. Peripheral PLL Structure



ALT\_CLKs are to be used for internal test purpose and should not be used in functional mode.

The PLL is locked at 960 MHz. The PLL output is divided by the M2 divider to generate a 192-MHz CLKOUT. This clock is gated in the PRCM to form the PRU-ICSS UART clock. There is a /2 divider to create 96 MHz for MMC\_CLK. The clock is also divided within the PRCM by a fixed /4 divider to create a 48-MHz clock for the SPI, UART and I2C modules. The 48-MHz clock is further divided by a fixed /2 divider and a fixed /732.4219 divider to create an accurate 32.768-KHz clock for Timer and debounce use.

Table 8-22. Per PLL Typical Frequencies (MHz)

| Clock                      | Source   | Power-On-Reset / PLL Bypass |                       | OPP100    |            | OPP50     |            |
|----------------------------|--|-----------------------------|-----------------------|-----------|------------|-----------|------------|
|                            |  | DIV Value                   | Freq                  | DIV Value | Freq (MHz) | DIV Value | Freq (MHz) |
| PLL Lock frequency         | PLL  | -                           | -                     | -         | 960        | -         | 96         |
| USB_PHY_CLK                | CLKDCOLDO  | -                           | Held Low              | -         | 960        | -         | N/A        |
| PER_CLKOUTM2               | CLKOUT of ADPLLJ<br>CLKOUT uses PLL's M2 Divider when PLL is locked and PLL's N2 divider when PLL Bypass | N2 is 0 on power-on-reset   | Mstr Xtal/ (N2+1)     | 5         | 192        | 1         | 96         |
| MMC_CLK                    | PER_CLKOUTM2   | 2                           | Mstr Xtal/ ((N2+1)*2) | 2         | 96         | 2         | 48         |
| SPI_CLK, UART_CLK, I2C_CLK | PER_CLKOUTM2   | 4                           | Mstr Xtal/ ((N2+1)*4) | 4         | 48         | 4         | 24         |
| CLK_24                     | CLK_48   | 2                           | CLK_48 /2             | 2         | 24         | 2         | 12         |
| CLK_32KHz                  | CLK_24 (output of CLK_48/2)  | 732.4219                    | CLK_24 / <CLK32_DIV>  | 732.4219  | 0.032768   | 366.2109  | 0.032768   |

The ADPLLJ module supports two different bypass modes via their internal MNBypass mode and their external Low Power Idle bypass mode. The PLLs are in the MNBypass mode after power-on reset and can be configured by software to enter Low Power Idle bypass mode for power-down.

The PER PLL can use the Low Power Idle bypass mode. When the internal bypass mode is selected, the CLKOUT output is driven by CLKINP/(N2+1) where N2 is driven by the PRCM. CLKINP defaults to the master oscillator input (typically 24 MHz)

#### 8.1.6.8.1 Configuring the Peripheral PLL

The following steps detail how to configure the peripheral PLL.

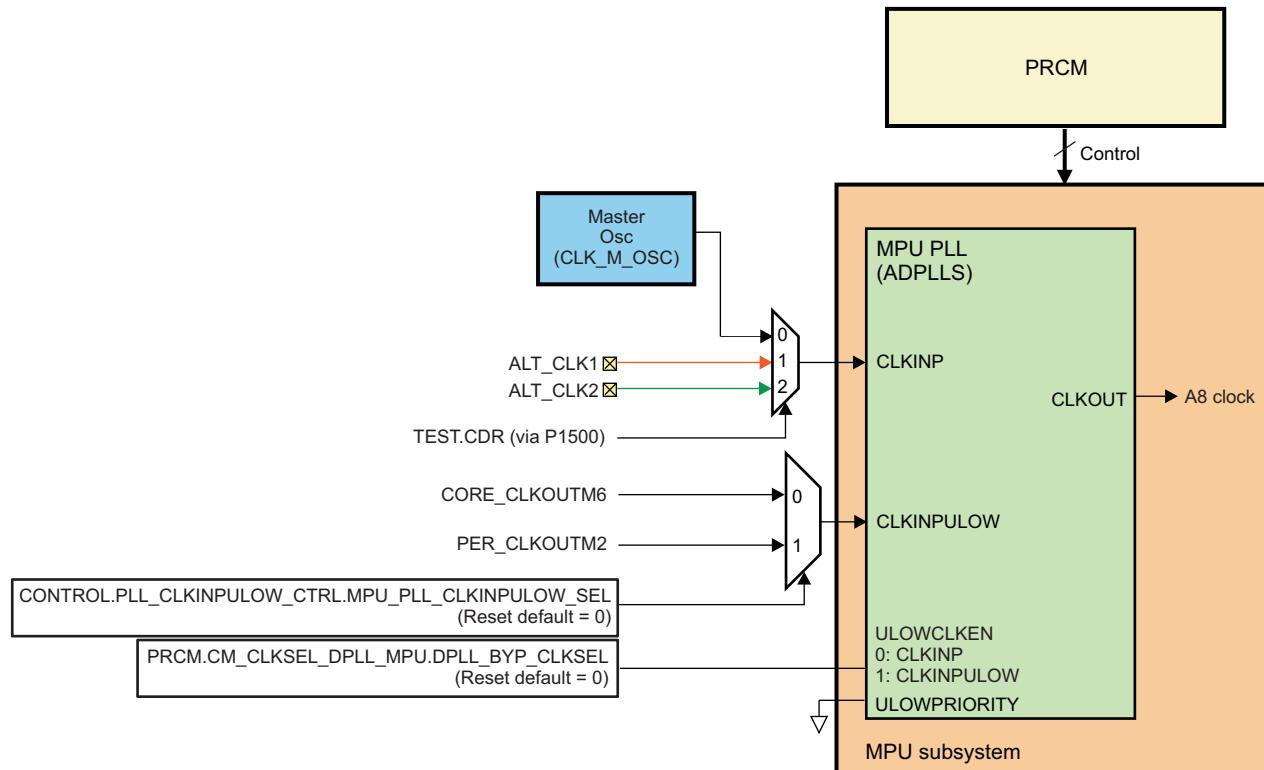
1. Switch PLL to bypass mode by setting CM\_CLKMODE\_DPLL\_PER.DPLL\_EN to 0x4.
2. Wait for CM\_IDLEST\_DPLL\_PER.ST\_MN\_BYPASS = 1 to ensure PLL is in bypass (CM\_IDLEST\_DPLL\_PER.ST\_DPLL\_CLK should also change to 0 to denote the PLL is unlocked).
3. Configure Multiply and Divide values by setting CM\_CLKSEL\_DPLL\_PER.DPLL\_MULT and DPLL\_DIV to the desired values.
4. Configure M2 divider by setting CM\_DIV\_M2\_DPLL\_PER.DPLL\_CLKOUT\_DIV to the desired value.
5. Switch over to lock mode by setting CM\_CLKMODE\_DPLL\_PER.DPLL\_EN to 0x7.
6. Wait for CM\_IDLEST\_DPLL\_PER.ST\_DPLL\_CLK = 1 to ensure PLL is locked (CM\_IDLEST\_DPLL\_PER.ST\_MN\_BYPASS should also change to 0 to denote the PLL is out of bypass mode).

**Note:** M2 divider can also be changed on-the-fly (ie., there is no need to put the PLL in bypass and back to lock mode). After changing CM\_DIV\_M2\_DPLL\_PER.DPLL\_CLKOUT\_DIV, check CM\_DIV\_M2\_DPLL\_PER.DPLL\_CLKOUT\_DIVCHACK for a toggle (a change from 0 to 1 or 1 to 0) to see if the change was acknowledged by the PLL.

#### 8.1.6.9 MPU PLL Description

The Cortex A8 MPU subsystem includes an internal ADPLLS for generating the required Cortex A8 MPU clocks. This PLL is driven by the master oscillator output with control provided by PRCM registers.

Figure 8-11. MPU Subsystem PLL Structure



For example:

For a frequency for MPU, say 600 MHz, the ADPLLs is configured (PLL locked at 1200 MHz and M2 Divider =1) so as to expect CLKOUT = 600 MHz .

The ULOWCLKEN input from a programmable PRCM register selects whether CLKINP or CLKINPULOW is the bypass clock source. This is a glitch free switch. When CLKINP is selected it is sourced through the ADPLLs 1/(N2+1) divider. The PRCM register defaults to 0 on power-up to select the CLKINP source.

The CLKINPULOW input may be sourced from the CORE\_CLKOUTM6 from the Core PLL, or PER\_CLKOUTM2 from the Per PLL. These PLL output clocks can be used as alternate clock sources in low power active use cases for the MPU Subsystem clock when the PLL is in bypass mode.

#### 8.1.6.9.1 Configuring the MPU PLL

The following steps detail how to configure the MPU PLL.

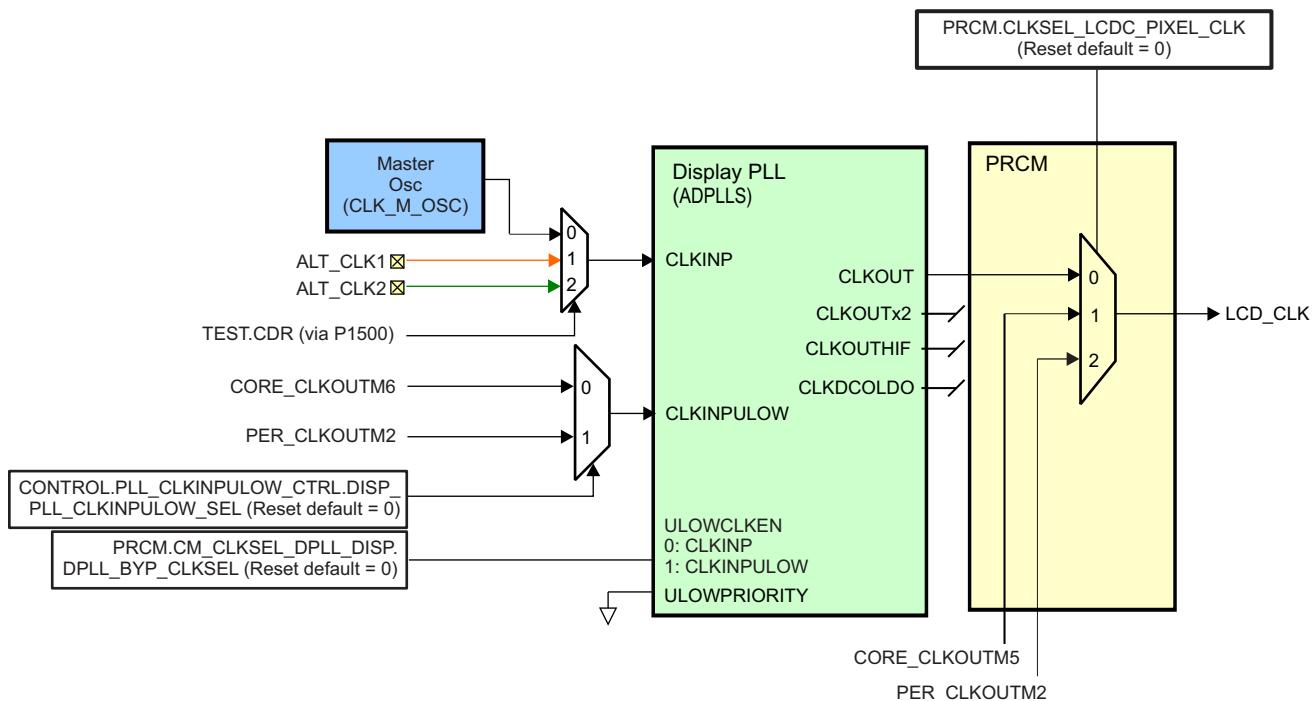
1. Switch PLL to bypass mode by setting CM\_CLKMODE\_DPLL\_MPU.DPLL\_EN to 0x4.
2. Wait for CM\_IDLEST\_DPLL\_MPU.ST\_MN\_BYPASS = 1 to ensure PLL is in bypass (CM\_IDLEST\_DPLL\_MPU.ST\_DPLL\_CLK should also change to 0 to denote the PLL is unlocked).
3. Configure Multiply and Divide values by setting CM\_CLKSEL\_DPLL\_MPU.DPLL\_MULT and DPLL\_DIV to the desired values.
4. Configure M2 divider by setting CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIV to the desired value.
5. Switch over to lock mode by setting CM\_CLKMODE\_DPLL\_MPU.DPLL\_EN to 0x7.
6. Wait for CM\_IDLEST\_DPLL\_MPU.ST\_DPLL\_CLK = 1 to ensure PLL is locked (CM\_IDLEST\_DPLL\_MPU.ST\_MN\_BYPASS should also change to 0 to denote the PLL is out of bypass mode).

**Note:** M2 divider can also be changed on-the-fly (ie., there is no need to put the PLL in bypass and back to lock mode). After changing CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIV, check CM\_DIV\_M2\_DPLL\_MPU.DPLL\_CLKOUT\_DIVCHACK for a toggle (a change from 0 to 1 or 1 to 0) to see if the change was acknowledged by the PLL.

### 8.1.6.10 Display PLL Description

The Display PLL provides the pixel clock required for the LCD display and is independent from the other peripheral and infrastructure clocks. The PLL is clocked from the Master Oscillator. The ADPLLS M2 divider determines the output clock frequency which is clock gated by the PRCM as shown in [Figure 8-12](#).

**Figure 8-12. Display PLL Structure**



For example: say frequency for pixel clock 100 MHz, the ADPLLS is configured (PLL locked at 200 MHz and M2 Divider =1) so as to expect CLKOUT = 100 MHz.

The ULOWCLKEN input from a programmable PRCM register selects whether CLKINP or CLKINPULOW is the bypass clock source. This is a glitch free switch. When CLKINP is selected it is sourced through the ADPLLS 1/(N2+1) divider. The PRCM register defaults to 0 on power-up to select the CLKINP source.

The CLKINPULOW input is sourced from the CORE\_CLKOUTM6 from the Core PLL or PER\_CLKOUTM2 from the Per PLL. This PLL output clock can be used as an alternate clock source in low power active use cases for the pixel clock when the Display PLL is in bypass mode.

#### 8.1.6.10.1 Configuring the Display PLL

The following steps detail how to configure the display PLL.

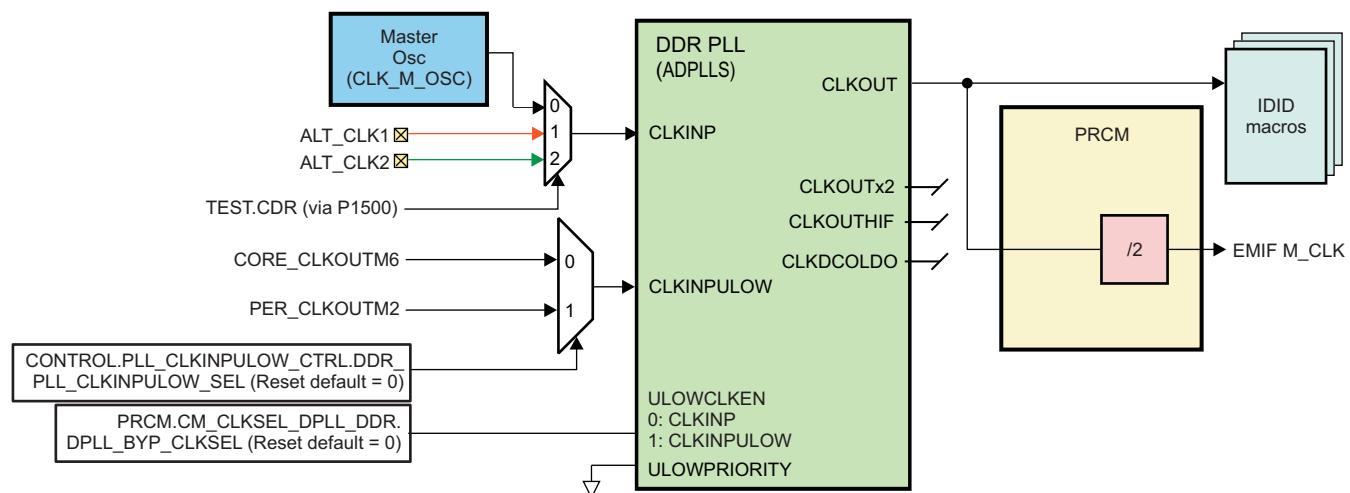
1. Switch PLL to bypass mode by setting CM\_CLKMODE\_DPLL\_DISP.DPLL\_EN to 0x4.
2. Wait for CM\_IDLEST\_DPLL\_DISP.ST\_MN\_BYPASS = 1 to ensure PLL is in bypass (CM\_IDLEST\_DPLL\_DISP.ST\_DPLL\_CLK should also change to 0 to denote the PLL is unlocked).
3. Configure Multiply and Divide values by setting CM\_CLKSEL\_DPLL\_DISP.DPLL\_MULT and DPLL\_DIV to the desired values.
4. Configure M2 divider by setting CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIV to the desired value.
5. Switch over to lock mode by setting CM\_CLKMODE\_DPLL\_DISP.DPLL\_EN to 0x7.
6. Wait for CM\_IDLEST\_DPLL\_DISP.ST\_DPLL\_CLK = 1 to ensure PLL is locked (CM\_IDLEST\_DPLL\_DISP.ST\_MN\_BYPASS should also change to 0 to denote the PLL is out of bypass mode).

**Note:** M2 divider can also be changed on-the-fly (ie., there is no need to put the PLL in bypass and back to lock mode). After changing CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIV, check CM\_DIV\_M2\_DPLL\_DISP.DPLL\_CLKOUT\_DIVCHACK for a toggle (a change from 0 to 1 or 1 to 0) to see if the change was acknowledged by the PLL.

### 8.1.6.11 DDR PLL Description

The DDR PLL provides the clocks required by the DDR macros and the EMIF and is independent from the other peripheral and infrastructure clocks. The PLL is clocked from the Master Oscillator. The ADPLLS M2 divider determines the output clock frequency which is connected directly to the DDR Macros. The clock is also routed through the PRCM where a fixed /2 divider is used to create the M\_CLK used by the EMIF as shown in Figure 8-13.

Figure 8-13. DDR PLL Structure



For OPP information, see the device-specific data manual.

Example frequency for DDR clock, say 266 MHz, the ADPLLS is configured (PLL locked at 532 MHz and M2 Divider =1) so as to expect CLKOUT = 266 MHz.

The ULOWCLKEN input from a programmable PRCM register selects whether CLKINP or CLKINPULOW is the bypass clock source. This is a glitch free switch. When CLKINP is selected it is sourced through the ADPLLS 1/(N2+1) divider. The PRCM register defaults to 0 on power-up to select the CLKINP source.

The CLKINPULOW input may be sourced from the CORE\_CLKOUTM6 from the Core PLL, or PER\_CLKOUTM2 from the Per PLL. These PLL output clocks can be used as alternate clock sources in low power active use cases for the DDR clocks when PLL is in bypass mode

#### 8.1.6.11.1 Configuring the DDR PLL

The following steps detail how to configure the DDR PLL.

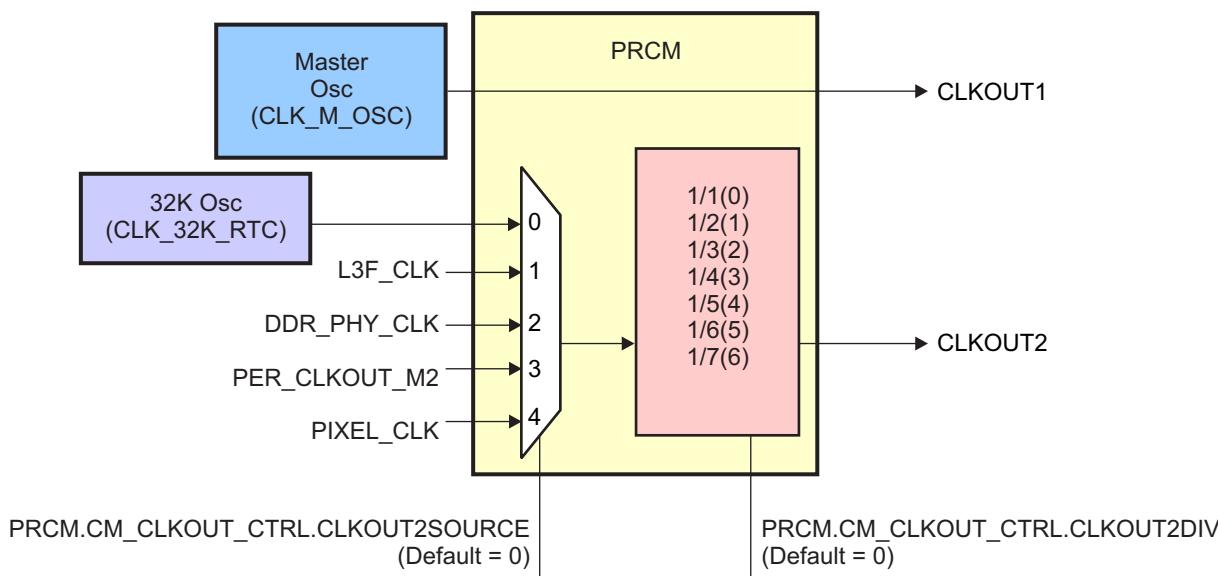
1. Switch PLL to bypass mode by setting CM\_CLKMODE\_DPLL\_DDR.DPLL\_EN to 0x4.
2. Wait for CM\_IDLEST\_DPLL\_DDR.ST\_MN\_BYPASS = 1 to ensure PLL is in bypass (CM\_IDLEST\_DPLL\_DDR.ST\_DPLL\_CLK should also change to 0 to denote the PLL is unlocked).
3. Configure Multiply and Divide values by setting CM\_CLKSEL\_DPLL\_DDR.DPLL\_MULT and DPLL\_DIV to the desired values.
4. Configure M2 divider by setting CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIV to the desired value.
5. Switch over to lock mode by setting CM\_CLKMODE\_DPLL\_DDR.DPLL\_EN to 0x7.
6. Wait for CM\_IDLEST\_DPLL\_DDR.ST\_DPLL\_CLK = 1 to ensure PLL is locked (CM\_IDLEST\_DPLL\_DDR.ST\_MN\_BYPASS should also change to 0 to denote the PLL is out of bypass mode).

**Note:** M2 divider can also be changed on-the-fly (i.e., there is no need to put the PLL in bypass and back to lock mode). After changing CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIV, check CM\_DIV\_M2\_DPLL\_DDR.DPLL\_CLKOUT\_DIVCHACK for a toggle (a change from 0 to 1 or 1 to 0) to see if the change was acknowledged by the PLL.

#### 8.1.6.12 CLKOUT Signals

The CLKOUT1 and CLKOUT2 signals go device pads and can be used as source clocks for FPGAs or other system devices. CLKOUT1 created from the master oscillator. CLKOUT2 can be sourced from the 32-KHz crystal oscillator or any of the PLL (except MPU PLL) outputs. The selected output can be further modified by a programmable divider to create the desired output frequency.

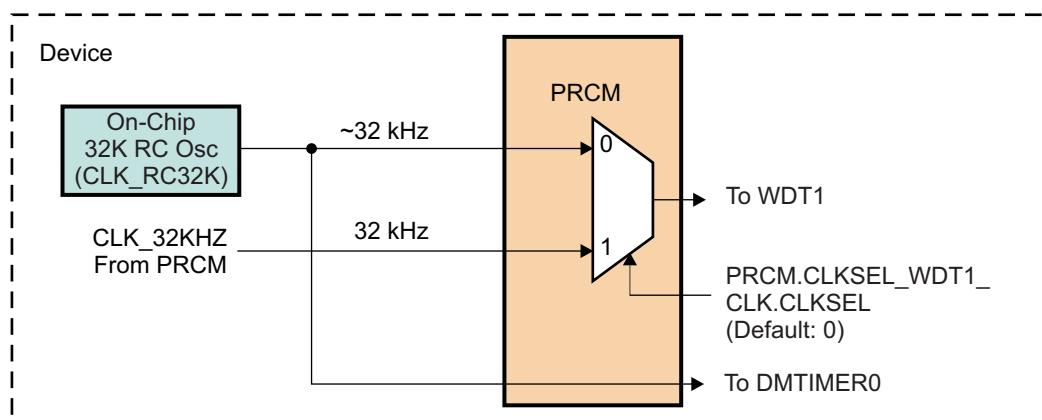
**Figure 8-14. CLKOUT Signals**



#### 8.1.6.13 Timer Clock Structure

The CLK\_32KHZ clock is an accurate 32.768-MHz clock derived from the PER PLL and can also be selected for the WDT1. The DMTIMER0 can only be clocked from the internal RC oscillator (CLK\_RC32K). The clock options are shown in Figure 8-15.

**Figure 8-15. Watchdog Timer Clock Selection**

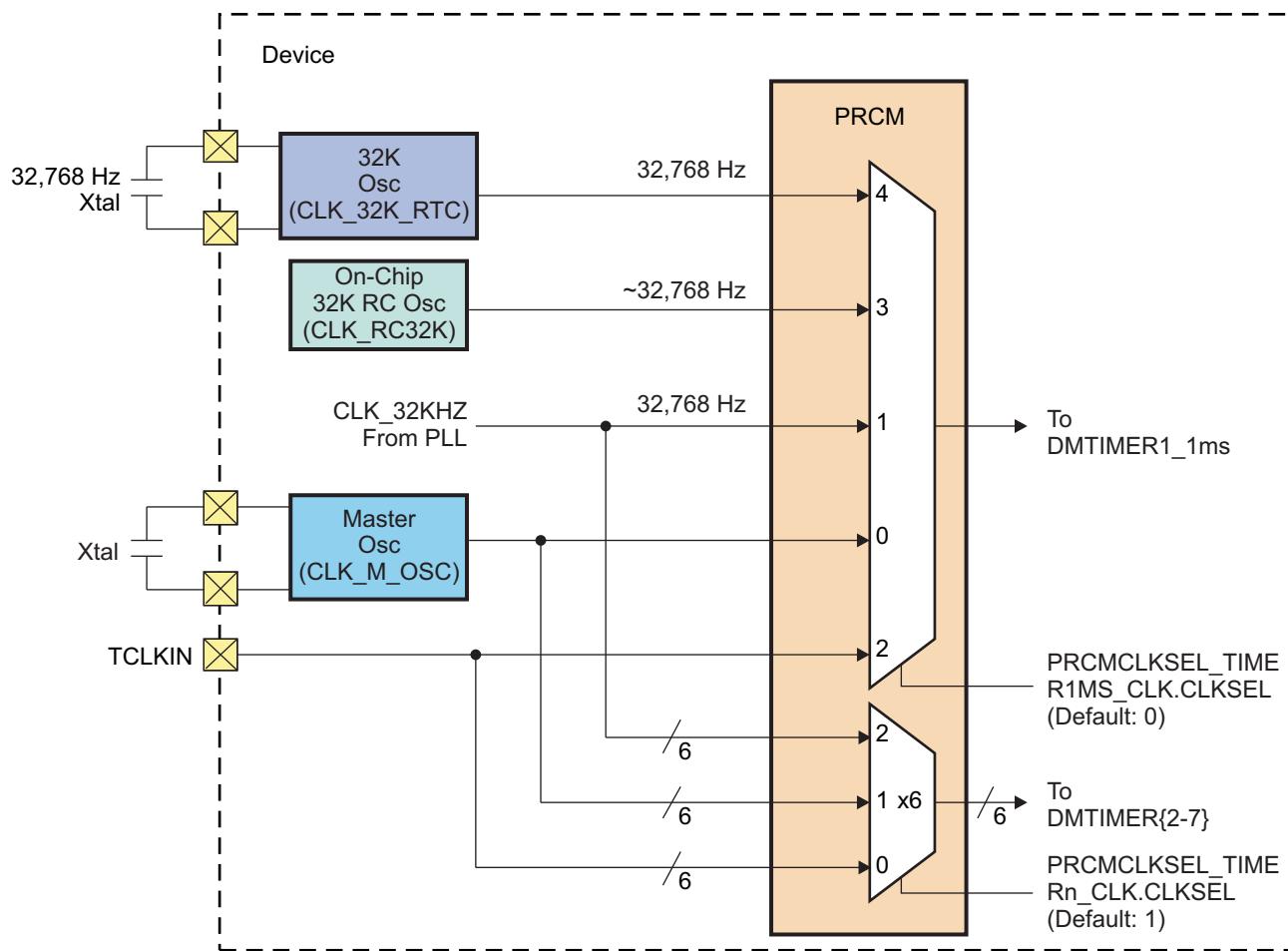


All mux selections are in PRCM unless explicitly shown otherwise in the diagrams.

The clock selections for the other device Timer modules are shown in [Figure 8-16](#). CLK\_32KHZ, the master oscillator, and the external pin (TCLKIN) are optional clocks available for timers which may be selected based on end use application.

DMTIMER1 is implemented using the DMTimer\_1ms module which is capable of generating an accurate 1ms tick using a 32.768 KHz clock. During low power modes, the Master Oscillator is disabled. CLK\_32KHZ also would not be available in this scenario since it is sourced from the Master Osc based PER PLL. Hence, in low power modes DMTIMER1 in the WKUP domain can use the 32K RC oscillator for generating the OS (operating system) 1ms tick generation and timer based wakeup. Since most applications expect an accurate 1ms OS tick which the inaccurate 32K RC (16-60 KHz) oscillator cannot provide, a separate 32768 Hz oscillator (32K Osc) is provided as another option.

**Figure 8-16. Timer Clock Selection**

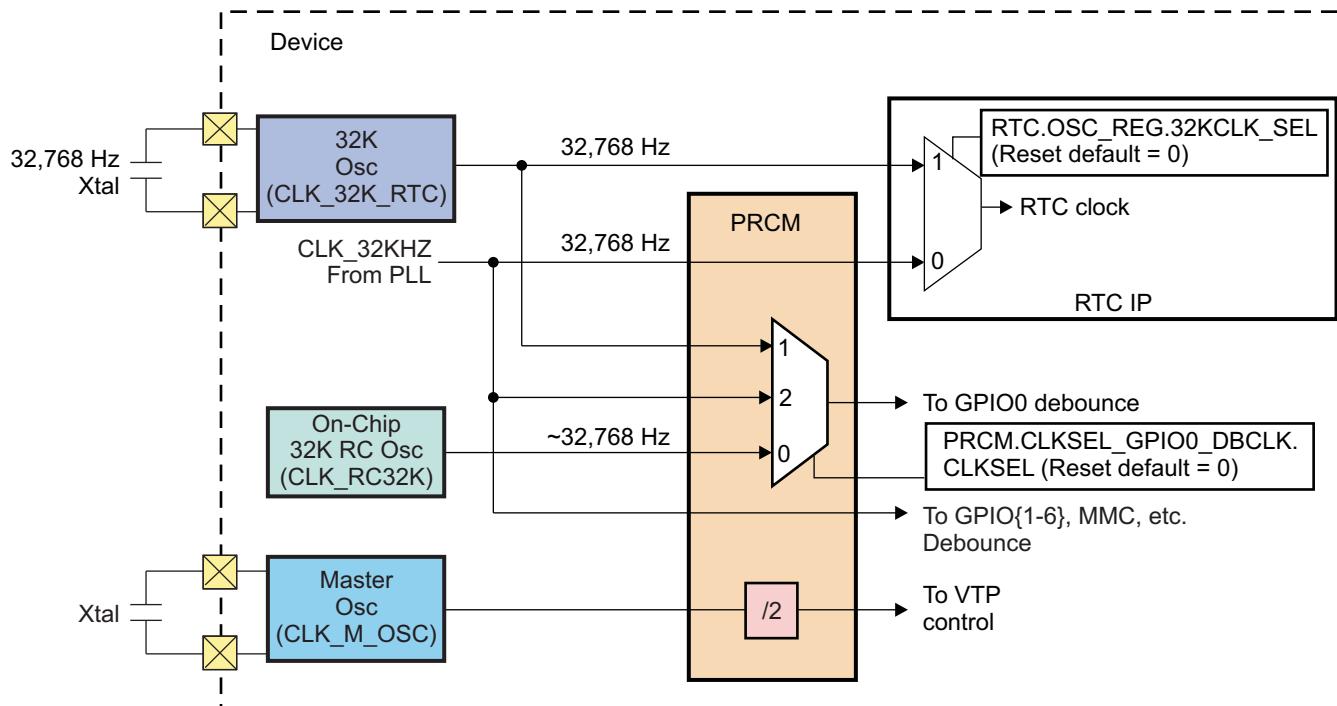


All mux selections are in PRCM unless explicitly shown otherwise in the diagrams.

The RTC, Debounce and VTP clock options are shown in [Figure 8-17](#). In low power modes, the debounce for GPIO0 in WKUP domain can use the accurate 32768 Hz crystal oscillator or the inaccurate (16 KHz to 60 KHz) 32K RC oscillator when the Master Osc is powered down.

The 32K Osc requires an external 32768-Hz crystal.

All mux selections are in PRCM unless explicitly shown otherwise in the diagrams.

**Figure 8-17. RTC, VTP, and Debounce Clock Selection**


All mux selections are in PRCM unless explicitly shown otherwise in the diagrams.

### 8.1.7 Reset Management

#### 8.1.7.1 Overview

The PRCM manages the resets to all power domains inside device and generation of a single reset output signal through device pin, PWRONRSTn, for external use. The PRCM has no knowledge of or control over resets generated locally within a module, e.g., via the OCP configuration register bit IPName\_SYSConfig.SoftReset.

All PRCM reset outputs are asynchronously asserted. These outputs are active-low except for the PLL resets. Deassertion is synchronous to the clock which runs a counter used to stall, or delay, reset de-assertion upon source deactivation. This clock will be CLK\_M\_OSC used by all the reset managers. All modules receiving a PRCM generated reset are expected to treat the reset as asynchronous and implement local re-synchronization upon de-activation as needed.

One or more Reset Managers are required per power domain. Independent management of multiple reset domains is required to meet the reset sequencing requirements of all modules in the power domain.

#### 8.1.7.2 Reset Concepts and Definitions

The PRCM collects many sources of reset. Here below is a list of qualifiers of the source of reset:

- Cold reset: it affects all the logic in a given entity
- Warm reset: it is a partial reset which doesn't affect all the logic in a given entity
- Global reset: it affects the entire device
- Local reset: it affects part of the device (1 power domain for example)
- S/W reset: it is initiated by software
- H/W reset: it is hardware driven

Each reset source is specified as being a cold or warm type. Cold types are synonymous with power-on-reset (POR) types. Such sources are applied globally within each receiving entity (i.e., sub-system, module, macro-cell) upon assertion. Cold reset events include: device power-up, power-domain power-up, and eFuse programming failures.

Warm reset types are not necessarily applied globally within each receiving entity. A module may use a warm reset to reset a subset of its logic. This is often done to speed-up reset recovery time, i.e., the time to transition to a safe operating state, compared to the time required upon receipt of a cold reset. Warm reset events include: software initiated per power-domain, watch-dog time-out, externally triggered, and emulation initiated.

Reset sources, warm or cold types, intended for device-wide effect are classified as global sources. Reset sources intended for regional effect are classified as local sources.

Each Reset Manager provides two reset outputs. One is a cold reset generated from the group of global and local cold reset sources it receives. The other is a warm+cold reset generated from the combined groups of, global and local, cold and warm reset sources it receives.

The Reset Manager asserts one, or both, of its reset outputs asynchronously upon reset source assertion. Reset deassertion is extended beyond the time the source gets de-asserted. The reset manager will then extend the active period of the reset outputs beyond the release of the reset source, according to the PRCM's internal constraints and device's constraints. Some reset durations can be software-configured. Most (but not all) reset sources are logged by PRCM's reset status registers. The same reset output can generally be activated by several reset sources and the same reset source can generally activate several reset outputs. All the reset signals output of the PRCM are active low. Several conventions are used in this document for signal and port names. They include:

- "\_RST" in a signal or port name is used to denote reset signal.
- "\_PWRON\_RST" in a signal or port name is used to denote a cold reset source

### 8.1.7.3 Global Power On Reset (Cold Reset)

There are several cold reset sources. See [Table 8-23](#) for a summary of the different reset sources.

#### 8.1.7.3.1 Power On Reset (PORz)

The source of power on reset is PORz signal on the device. Everything on device is reset with assertion of power on reset. This reset is non-blockable. PORz can be driven by external power management devices or power supervisor circuitry. During power-up, when power supplies to the device are ramping up, PORz needs to be driven Low. When the ramp-up is complete and supplies reach their steady-state values, PORz need to be driven High. During normal operation when any of the device power supplies are turned OFF, PORz must be driven Low.

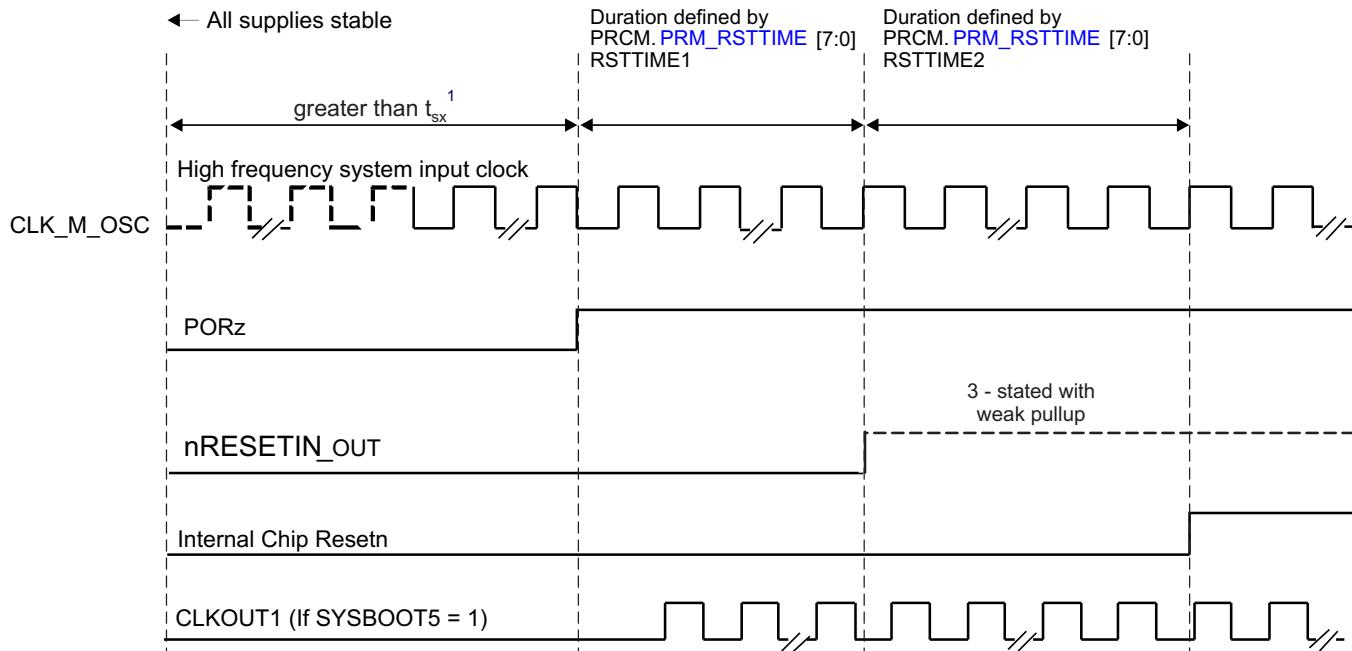
#### 8.1.7.3.2 PORz Sequence

- PORz pin at chip boundary gets asserted (goes low). **Note:** The state of nRESETIN\_OUT during PORz assertion should be a don't care, it should not affect PORz (only implication is if they are both asserted and nRESETIN\_OUT is deasserted after PORz you will get re-latching of boot config pins and may see warm nRESETIN\_OUT flag set in PRCM versus POR).
- All IOs will go to tri-state immediately.
- When power comes-up, PORz value will propagate to the PRCM.
- PRCM will fan out reset to the complete chip and all logic which uses async reset will get reset. nRESETIN\_OUT will go low to indicate reset-in-progress.
- External clocks will start toggling and PRCM will propagate these clocks to the chip keeping PLLs in bypass mode.
- All logic using sync reset will get reset.
- When power and clocks to the chip are stable, PORz must be de-asserted.
- Boot configuration pins are latched upon de-assertion of PORz pin
- IO cell controls from IPs for all the IOs with a few exceptions (see datasheet for details) are driven by GPIO module. GPIO puts all IOs in input mode.

- FuseFarm reset will be de-asserted to start eFuse scanning.
- Once eFuse scanning is complete, reset to the host processor and to all other peripherals (peripherals without local processor) will be de-asserted.
- nRESETIN\_OUT will be de-asserted after time defined by PRM\_RSTTIME.RSTTIME1.
- Once host processors finish booting then all remaining peripherals will see reset de-assertion.

Note that all modules with local CPUs will have local reset asserted by default at PORz and reset de-assertion would require host processor to write to respective registers in PRCM.

**Figure 8-18. PORz**



(1) For information on  $t_{sx}$ , see *AM335x ARM Cortex-A8 Microprocessors (MPUs)* (literature number [SPRS717](#)).

#### 8.1.7.3.3 Bad Device Reset

This reset is asserted whenever the DEVICE\_TYPE encodes an unsupported device type, such as the code for a "bad" device.

#### 8.1.7.3.4 Global Cold Software Reset (GLOBAL\_COLD\_SW\_RST)

The source for GLOBAL\_COLD\_SW\_RST is generated internally by the PRM. It is activated upon setting the PRM\_RSTCTRL.RST\_GLOBAL\_COLD\_SW bit in the PRM memory map. This bit is self-clearing, i.e., it is automatically cleared by the hardware.

#### 8.1.7.4 Global Warm Reset

##### 8.1.7.4.1 External Warm Reset

nRESETIN\_OUT is a bidirectional warm reset signal. As an input, it is typically used by an external source as a device reset. Refer to Table 8-24 for a summary of the differences between a warm reset and cold reset. Some of these differences are:

- The warm reset can be blocked to the EMAC switch and its reference clock source PLL using the RESET\_ISO register in the Control Module.
- The warm reset assumes that clocks and power to the chip are stable from assertion through deassertion, whereas during the cold reset, the power supplies can become stable during assertion
- Some PRCM and Control module registers are warm reset insensitive and maintain their value

throughout a warm reset

- SYSBOOT pins are not latched with a warm reset. The device will boot with the SYSBOOT values from the previous cold reset.
- Most debug subsystem logic is not affected by warm reset. This allows you to maintain any debug sessions throughout a warm reset event.
- PLLs are not affected by warm reset

As an output, nRESETIN\_OUT can be used to reset external devices. nRESET\_OUT will drive low during a cold reset or an internally generated warm reset. After completion of a cold or warm reset, nRESETIN\_OUT will continue to drive low for a period defined by PRM\_RSTTIME.RSTTIME1. RSTTIME1 is a timer that counts down to zero at a rate equal to the high frequency system input clock CLK\_M\_OSC. This allows external devices to be held in reset for some time after the AM335x comes out of reset.

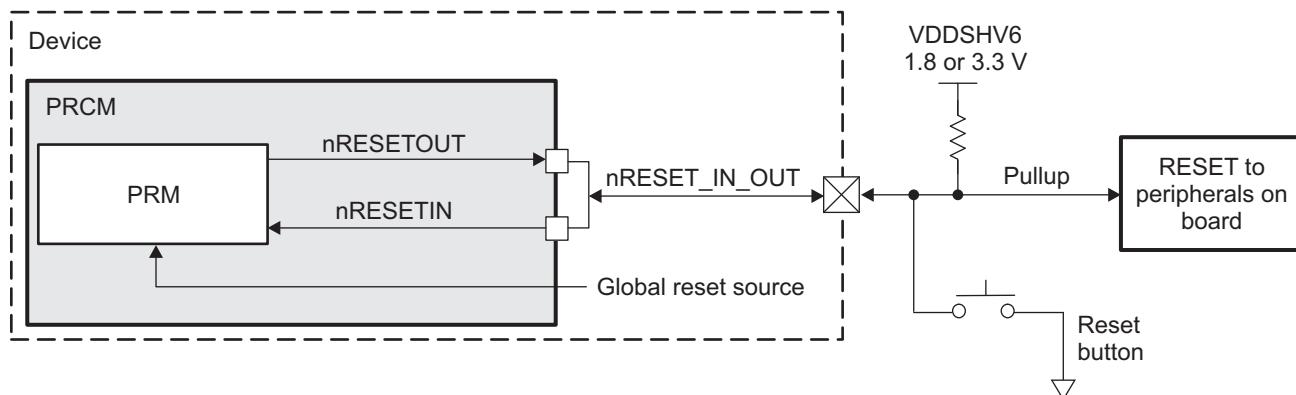
Caution must be used when implementing the nRESETIN\_OUT as an bi-directional reset signal. Because of the short maximum time allowed using RSTTIME1, it does not supply an adequate debounce time for an external push button circuit. The processor could potentially start running while external components are still in reset. It is recommended that this signal be used as input only (do not connect to other devices as a reset) to implement a push button reset circuit to the AM335x, or an output only to be able to reset other devices after an AM335x reset completes.

#### 8.1.7.4.1.1 Warm Reset Input/Reset Output (nRESETIN\_OUT)

Any global reset source (internal or external) causes nRESETIN\_OUT to be driven and maintained at the boundary of the device for at least the amount of time configured in the PRCM.PRM\_RSTTIME.RSTTIME1 bit field. This ensures that the device and its related peripherals are reset together. The nRESETIN\_OUT output buffer is configured as an open-drain; consequently, an external pull-up resistor is required.

After the de-assertion, the bi-directional pin nRESETIN\_OUT is tri-stated to allow for assertion from off chip source (externally).

**Figure 8-19. External System Reset**



Note: It is recommended to implement warm reset as an input only (for example, push button) or an output only (to reset external peripherals), not both.

The device will have one pin nRESETIN\_OUT which reflects chip reset status. This output will always assert asynchronously when any chip watchdog timer reset occurs if any of the following reset events occurs:

- POR (only internal stretched portion of reset event after bootstrap is latched)
- External Warm reset (nRESETIN\_OUT pin, only internal stretched portion of reset event after bootstrap is latched)
- Emulation reset (Cold or warm from ICEPICK)
- Reset requestor
- SW cold/warm reset

This output will remain asserted as long as PRCM keeps reset to the host processor asserted.

**Note:** TRST does not cause RSTOUTn assertion

#### 8.1.7.4.1.2 Warm Reset Sequence

1. nRESETIN\_OUT pin at chip boundary gets asserted (goes low). NOTE: For Warm Reset sequence to work as described, it is expected that PORz pin is always inactive, otherwise you will get PORz functionality as described in previous section.
2. All IOs (except test and emulation) will go to tri-state immediately.
3. Chip clocks are not affected as both PLL and dividers are intact.
4. nRESETIN\_OUT gets de-asserted after 30 cycles
5. PRCM de-asserts reset to the host processor and all other peripherals without local CPUs.
6. Note that all IPs with local CPUs will have local reset asserted by default at Warm Reset and reset de-assertion would require host processor to write to respective registers in PRCM.

Figure 8-20 shows the nRESETIN\_OUT waveform when using nRESETIN\_OUT as warm reset source. For the duration when external warm reset switch is closed, both the device and chip will be driving zero.

**Figure 8-20. nRESETIN\_OUT Waveform as Warm Reset Source**

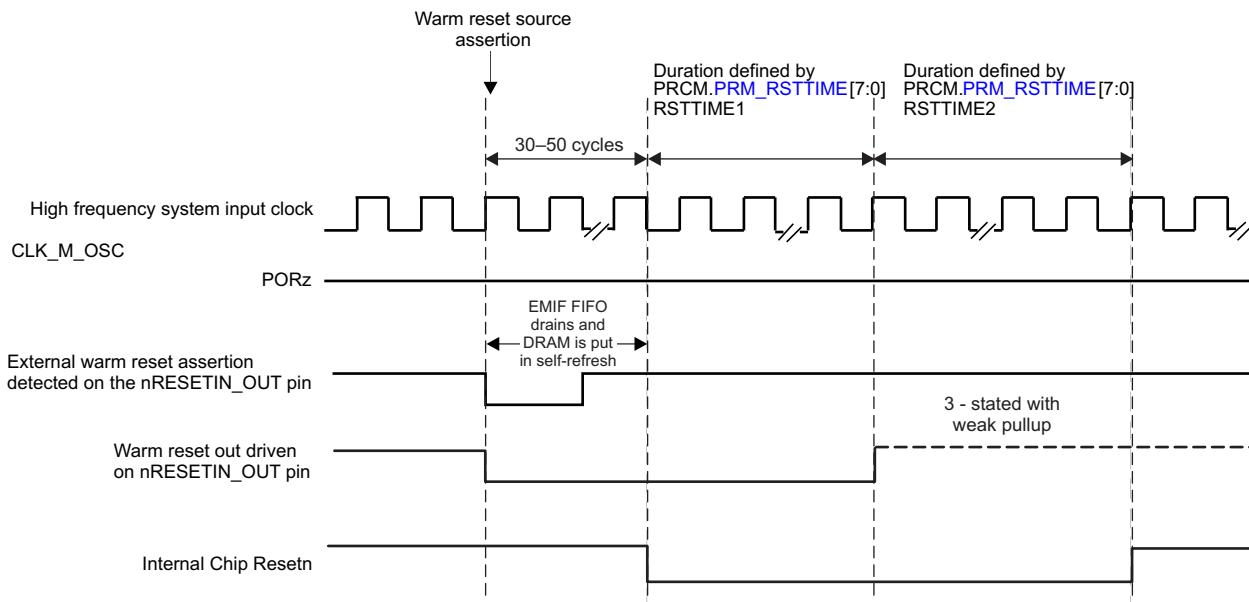
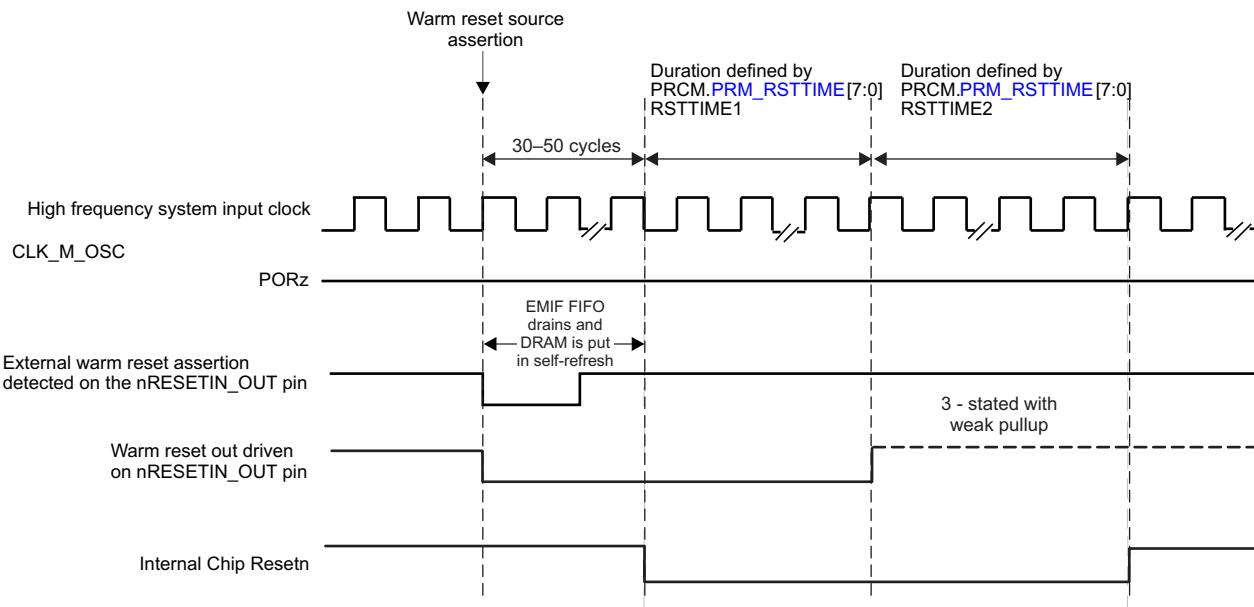


Figure 8-21 shows the nRESETIN\_OUT waveform when any one of the warm reset sources captured except using nRESETIN\_OUT itself as warm reset source.

**Figure 8-21. nRESETIN\_OUT Waveform Not as Warm Reset Source**


#### 8.1.7.4.2 Watchdog Timer

There is one watchdog timer on the device. The reset is not blockable.

#### 8.1.7.4.3 Global Warm Software Reset (GLOBAL\_SW\_WARM\_RST)

#### 8.1.7.4.4 Test Reset (TRSTz)

This reset is triggered from TRSTz pin on JTAG interface. This is a non-blockable reset and it resets test and emulation logic.

**NOTE:** A PORz reset assertion should cause entire device to reset including all test and emulation logic regardless of the state of TRSTz. Therefore, PORz assertion will achieve full reset of the device even if TRSTz pin is pulled permanently high and no special toggling of TRSTz pin is required during power ramp to achieve full POR reset to the device. Further, it is acceptable for TRSTz input to be pulled permanently low during normal functional usage of the device in the end-system to ensure that all test and emulation logic is kept in reset.

### 8.1.7.5 Reset Characteristics

The following table shows characteristic of each reset source.

**Table 8-23. Reset Sources**

| Characteristic                              | Cold Reset Sources |                  |                  | Warm Reset sources |                  |                  |       |
|---|--------------------|------------------|------------------|--------------------|------------------|------------------|-------|
|   | Pin PORz           | SW Cold Reset    | Bad Device       | Pin Warm Reset     | Watchdog Timer   | SW Warm Reset    | TRSTz |
| Boot pins latched                           | Y                  | N                | N                | N                  | N                | N                | N     |
| Resets Standard Efuses                      | Y                  | N                | N                | N                  | N                | N                | N     |
| Resets Customer Efuses                      | Y                  | Y                | Y                | Y                  | Y                | Y                | N     |
| DRAM contents preserved                     | N                  | N                | N                | Y <sup>(1)</sup>   | Y <sup>(1)</sup> | Y <sup>(1)</sup> | Y     |
| Resets PLLs <sup>(2)</sup>                  | Y                  | Y                | Y                | N                  | N                | N                | N     |
| Resets Clock Dividers <sup>(2)</sup>        | Y                  | Y                | Y                | N                  | N                | N                | N     |
| PLLs enter bypass mode <sup>(2)</sup>       | Y                  | Y                | Y                | Y                  | Y                | Y                | N     |
| Reset source blockable by emulation         | N                  | N                | N                | Y                  | Y                | Y                | Y     |
| Resets test and emulation logic             | Y                  | Y                | Y                | N                  | N                | N                | N     |
| Resets GMAC switch and related chip logic   | Y                  | N <sup>(3)</sup> | Y                | N <sup>(3)</sup>   | N <sup>(3)</sup> | N <sup>(3)</sup> |       |
| Resets Chip Functional Logic <sup>(4)</sup> | Y                  | Y                | Y                | Y                  | Y                | Y                | N     |
| Puts IOs in Tri-state                       | Y                  | Y                | Y                | Y                  | Y                | Y                | N     |
| Resets Pinmux Registers                     | Y                  | Y                | Y <sup>(5)</sup> | Y <sup>(5)</sup>   | Y <sup>(5)</sup> | Y <sup>(5)</sup> | N     |
| Reset out Assertion (nRESETIN_OUT Pin)      | Y                  | Y                | Y <sup>(5)</sup> | Y <sup>(5)</sup>   | Y <sup>(5)</sup> | Y <sup>(5)</sup> | N     |
| Resets RTC                                  | N                  | N                | N                | N                  | N                | N                | N     |

<sup>(1)</sup> The ROM software does not utilize this feature of DRAM content preservation. Hence, the AM335x re-boots like a cold boot for warm reset as well.

<sup>(2)</sup> CORE PLL is an exception when EMAC switch reset isolation is enabled

<sup>(3)</sup> Only true if GMAC switch reset isolation is enabled in control registers, otherwise will be reset.

<sup>(4)</sup> There are exception details in control module & PRCM registers which are captured in the register specifications in [Chapter 8](#) and [Chapter 9](#). This includes some pinmux registers which are warm reset in-sensitive.

<sup>(5)</sup> Some special IOs/Muxing registers like test, emulation, GEMAC Switch (When under reset isolation mode), etc related will not be affected under warm reset conditions.

**Note:** The Bandgap macros for the LDOs will be reset on PORz only.

#### 8.1.7.6 EMAC Switch Reset Isolation

The device will support reset isolation for the Ethernet Switch IP. This allows the device to undergo a warm reset without disrupting the switch or traffic being routed through the switch during the reset condition.

If configured by registers in control module that EMAC reset isolation is active, then behavior is as follows:

Any "Warm" reset source(except the software warm reset) will be blocked to the EMAC switch logic in the IP (the IP has two reset inputs to support such isolation) and to PLL (and its control bits) which is sourcing the EMAC switch clocks as required by the IP (50- or 125-MHz reference clocks). Also, the EMAC switch related IO pins must retain their pin muxing and not glitch (continuously controlled by the EMAC switch IP) by blocking reset to the controlling register bits.

If configured by registers in control module that EMAC reset isolation is NOT active (default state), then the warm reset sources are allowed to propagate as normal including to the EMAC Switch IP (both reset inputs to the IP).

All cold or POR resets will always propagate to the EMAC switch IP as normal (as otherwise defined in this document).

#### 8.1.7.7 Reset Priority

If more than one of these reset sources are asserted simultaneously then the following priority order should be used:

1. POR
2. TRSTz
3. External warm reset
4. Emulation
5. Reset requestors
6. Software resets

#### 8.1.7.8 Trace Functionality Across Reset

Other than POR, TRSTz and SW cold reset, none of the other resets will affect trace functionality. This requires that Debug\_SS has required reset isolation for the trace logic. Also the los and muxing control (if any) for trace los should not get affected by any other reset. Since none of the PLLs getting reset with other resets, clocks will be any way stable.

#### 8.1.7.9 RTC PORz

AM335x supports RTC only mode by supplying dedicated power to RTC module. The RTC module has a dedicated PORz signal (RTC\_PORz) to reset RTC logic and circuitry during powerup. RTC\_PORz is expected to be driven low when the RTC power supply is ramping up. After the power supply reaches its stable value, the RTC\_PORz can be de-asserted. The RTC module is not affected by the device PORz. Similarly RTC\_PORz does not affect the device reset.

If the RTC-only mode is not required, then PORz and RTC\_PORz needs to be shorted. For power-up sequencing with respect to RTC\_PORz, see the device specific datasheet.

### 8.1.8 Power-Up/Down Sequence

Each power domain has dedicated warm and cold reset.

Warm reset gets asserted each time there is any warm reset source requesting reset. Warm reset is also asserted when power domain moves from ON to OFF state.

Cold reset for the domain is asserted in response to cold reset sources. When domain moves from OFF to ON state then also cold reset gets asserted as this is similar to power-up condition for that domain.

### 8.1.9 IO State

All IOs except for JTAG i/f and Reset output (and any special cases mentioned in pinlist) should have their output drivers tri-state and internal pulls enabled during assertion of all reset sources. JTAG i/f IO is affected only by TRSTz.

**Note:** The PRUs and Cortex M3 processor are held under reset after global warm reset by assertion of software source of reset. Other domains are held under reset after global warm reset until the MPU software enables their respective interface clock.

### 8.1.10 Voltage and Power Domains

The following table shows how the device core logic is partitioned into two core logic voltage domains and four power domains. The table lists which voltage and power domain a functional module belongs.

**Table 8-24. Core Logic Voltage and Power Domains**

| Logic Voltage Domain Name | Module           |
|---------------------------|------------------|
| CORE                      | All Core Modules |
| RTC                       | RTC              |

### 8.1.10.1 Voltage Domains

The core logic is divided into two voltage domains: VDD\_CORE and VDD\_RTC.

### 8.1.10.2 Power Domains

In order to reduce power due to leakage, the core logic supply voltage to 4 of these power domains can be turned OFF with internal power switches. The internal power switches are controlled through memory mapped registers in Control Module.

In a use case whereby all the modules within a power domain are not used that power domain can be placed in the OFF state.

Error: Reference source not found shows the allowable combination power domain ON/OFF states and which power domains are switched via internal power switches. At power-on-reset, all domains except always-on will be in the power domain OFF state.

**Table 8-25. Power Domain State Table**

| MODE                       | POWER DOMAIN |            |            |            |            |            |
|----------------------------|--------------|------------|------------|------------|------------|------------|
|                            | WAKEUP       | MPU        | GFX        | PER        | RTC        | EFUSE      |
| No Voltage Supply          | N/A          | N/A        | N/A        | N/A        | N/A        | N/A        |
| Power On Reset             | ON           | OFF        | OFF        | OFF        | OFF        | OFF        |
| ALL OTHER FUNCTIONAL MODES | ON           | DON'T CARE |
| Internal Power Switch      | NO           | YES        | YES        | YES        | YES        | YES        |

### 8.1.11 Device Modules and Power Management Attributes List

**Table 8-26. Power Domain of Various Modules**

| Power Supply | Power Domain   | Modules OR Supply Destinations (sinks)  |
|--------------|--|---|
|              | PD_WKUP<br>This is non-switchable and cannot be shut OFF | Wake-CortexM3SS<br>L4_WKUP peripherals:<br>PRCM<br>Control Module<br>GPIO0<br>DMTIMER0_dmc<br>DMTIMER1<br>UART0<br>I2C0<br>TSC<br>WDT1<br>SmartReflex_c2_0<br>SmartReflex_c2_1<br>L4_WKUP |

**Table 8-26. Power Domain of Various Modules (continued)**

| Power Supply | Power Domain | Modules OR Supply Destinations (sinks)   |
|--------------|--------------|--|
| VDD_CORE     | PD_PER       | Toplevel: Pinmux gates/logic, oscillator<br>wake logic<br>DDR PHY (DIDs)<br>WKUP_DFTSS<br>Debugs<br>Digital section (VDD) of CORE PLL, PER<br>PLL, Display PLL and DDR PLL<br>Emulation sections of MPU<br>VDD of all IOs including crystal oscillators<br>RC Oscillator<br>Infrastructure:<br>L3<br>L4_PER, L4_Fast, L4_FW<br>EMIF4<br>EDMA<br>GPMC<br>OCMC controller<br>L3 / L4_PER / L4_Fast / L4_FW<br>Peripherals<br>PRU-ICSS<br>LCD controller<br>Ethernet Switch<br>USB Controller<br>GPMC<br>MMC0..2<br>DMTIMER2..7<br>Uart1..5<br>SPI0, 1<br>I2C1, 2<br>DCAN0, 1<br>McASP0, 1<br>ePWM0..2<br>eCAP0..2<br>eQEP0,1<br>GPIO1..3<br>ELM<br>Mailbox0, Spinlock<br>OCP_WP<br>Others:<br>USB2PHYCORE (VDD/digital section)<br>USB2PHYCM (VDD/digital section) |
|              | PD_GFX       | SGX530   |
| VDD_MPU      | PD_MPU       | CPU, L1, L2 of MPU   |
|              | PD_WKUP      | Interrupt controller of MPU Subsystem<br>Digital Portion (VDD) of MPU PLL  |
| VDD_RTC      | PD_RTC       | 1. RTC<br>VDD for 32 768 Hz Crystal Osc<br>VDD for IO for the alarm pin  |

### 8.1.11.1 Power Domain Power Down Sequence

The following sequence of steps happen during the power down of a power domain

All IPs (belonging to a power domain) with STANDBY interface will assert STANDBY. STANDBY assertion should get triggered by IP based on its activity on OCP initiator port. IP should assert STANDBY whenever initiator port is IDLE. Some of the IPs may not have this feature and they will require SW write to standby-mode register to get STANDBY assertion from IP.

1. SW will request all modules in given power domain to go to disable state by programming module control register inside PRCM.
2. PRCM will start and wait for completion of power management handshake with IPs (IdleReq/IdleAck).
3. PRCM will gate-off all the clocks to the power domain.
4. SW will request all clock domains in given power domain to go to “force sleep” mode by programming functional clock domain register in PRCM. Note that PRCM has already gated-off clocks and this register programming may look redundant.
5. SW will request PRCM to take this power domain to OFF state by programming PWRSTCTRL register. Note that this step can be skipped if PWRSTCTRL is permanently programmed to OFF state. When this is done, functional clock domain register decides when power domain will be taken to OFF state. Only reason not to have OFF state in PWRSTCTRL is to take power domain to just clock gate state without power gating.
6. PSCON specific to this power domain will assert isolation enable for the domain.
7. PRCM will assert warm and cold reset to the power domain.
8. PSCON will assert control signals to switch-off power using on-die switches.
9. On-die switches will send acknowledge back to PSCON.

### 8.1.11.2 Power Domain Power-Up Sequence

The following sequence of steps occurs during power-up of a power domain. This sequence is not relevant to always-on domain as this domain will never go to OFF state as long as the device is powered. This sequence will be repeated each time a domain is taken to ON state from OFF (including first time power-up). Note that some of the details are intentionally taken out here to simplify things.

There can be multiple reasons to start power-up sequence for a domain. For example it can be due to an interrupt from one of the IPs which is powered-up.

1. SW will request required clock domains inside this power domain to go to force wake-up state by programming functional clock domain register.
2. PRCM will enable clocks to the required clock domains.
3. PSCON specific to this power domain will assert control signal to un-gate the power.
4. Once power is un-gated, on die switches will send acknowledge back to PSCON.
5. PRCM will de-assert cold and warm reset to the power domain.
6. PRCM will turn-off isolation cells.
7. SW will request PRCM to enable required module in the power domain by programming module control register.
8. PRCM will initiate and wait for completion of PM protocol to enable the modules (IdleReq/IdleAck).

## 8.1.12 Clock Module Registers

### 8.1.12.1 CM\_PER Registers

Table 8-27 lists the memory-mapped registers for the CM\_PER. All register offset addresses not listed in Table 8-27 should be considered as reserved locations and the register contents should not be modified.

**Table 8-27. CM\_PER REGISTERS**

| Offset | Acronym                 | Register Name | Section                             |
|--------|-------------------------|---------------|-------------------------------------|
| 0h     | CM_PER_L4LS_CLKSTCTRL   |               | <a href="#">Section 8.1.12.1.1</a>  |
| 4h     | CM_PER_L3S_CLKSTCTRL    |               | <a href="#">Section 8.1.12.1.2</a>  |
| 8h     | CM_PER_L4FW_CLKSTCTRL   |               | <a href="#">Section 8.1.12.1.3</a>  |
| Ch     | CM_PER_L3_CLKSTCTRL     |               | <a href="#">Section 8.1.12.1.4</a>  |
| 14h    | CM_PER_CPGMAC0_CLKCTRL  |               | <a href="#">Section 8.1.12.1.5</a>  |
| 18h    | CM_PER_LCDC_CLKCTRL     |               | <a href="#">Section 8.1.12.1.6</a>  |
| 1Ch    | CM_PER_USB0_CLKCTRL     |               | <a href="#">Section 8.1.12.1.7</a>  |
| 24h    | CM_PER_TPTC0_CLKCTRL    |               | <a href="#">Section 8.1.12.1.8</a>  |
| 28h    | CM_PER_EMIF_CLKCTRL     |               | <a href="#">Section 8.1.12.1.9</a>  |
| 2Ch    | CM_PER_OCMCRAM_CLKCTRL  |               | <a href="#">Section 8.1.12.1.10</a> |
| 30h    | CM_PER_GPMC_CLKCTRL     |               | <a href="#">Section 8.1.12.1.11</a> |
| 34h    | CM_PER_MCASP0_CLKCTRL   |               | <a href="#">Section 8.1.12.1.12</a> |
| 38h    | CM_PER_UART5_CLKCTRL    |               | <a href="#">Section 8.1.12.1.13</a> |
| 3Ch    | CM_PER_MMC0_CLKCTRL     |               | <a href="#">Section 8.1.12.1.14</a> |
| 40h    | CM_PER_ELM_CLKCTRL      |               | <a href="#">Section 8.1.12.1.15</a> |
| 44h    | CM_PER_I2C2_CLKCTRL     |               | <a href="#">Section 8.1.12.1.16</a> |
| 48h    | CM_PER_I2C1_CLKCTRL     |               | <a href="#">Section 8.1.12.1.17</a> |
| 4Ch    | CM_PER_SPI0_CLKCTRL     |               | <a href="#">Section 8.1.12.1.18</a> |
| 50h    | CM_PER_SPI1_CLKCTRL     |               | <a href="#">Section 8.1.12.1.19</a> |
| 60h    | CM_PER_L4LS_CLKCTRL     |               | <a href="#">Section 8.1.12.1.20</a> |
| 64h    | CM_PER_L4FW_CLKCTRL     |               | <a href="#">Section 8.1.12.1.21</a> |
| 68h    | CM_PER_MCASP1_CLKCTRL   |               | <a href="#">Section 8.1.12.1.22</a> |
| 6Ch    | CM_PER_UART1_CLKCTRL    |               | <a href="#">Section 8.1.12.1.23</a> |
| 70h    | CM_PER_UART2_CLKCTRL    |               | <a href="#">Section 8.1.12.1.24</a> |
| 74h    | CM_PER_UART3_CLKCTRL    |               | <a href="#">Section 8.1.12.1.25</a> |
| 78h    | CM_PER_UART4_CLKCTRL    |               | <a href="#">Section 8.1.12.1.26</a> |
| 7Ch    | CM_PER_TIMER7_CLKCTRL   |               | <a href="#">Section 8.1.12.1.27</a> |
| 80h    | CM_PER_TIMER2_CLKCTRL   |               | <a href="#">Section 8.1.12.1.28</a> |
| 84h    | CM_PER_TIMER3_CLKCTRL   |               | <a href="#">Section 8.1.12.1.29</a> |
| 88h    | CM_PER_TIMER4_CLKCTRL   |               | <a href="#">Section 8.1.12.1.30</a> |
| ACh    | CM_PER_GPIO1_CLKCTRL    |               | <a href="#">Section 8.1.12.1.31</a> |
| B0h    | CM_PER_GPIO2_CLKCTRL    |               | <a href="#">Section 8.1.12.1.32</a> |
| B4h    | CM_PER_GPIO3_CLKCTRL    |               | <a href="#">Section 8.1.12.1.33</a> |
| BCh    | CM_PER_TPCC_CLKCTRL     |               | <a href="#">Section 8.1.12.1.34</a> |
| C0h    | CM_PER_DCAN0_CLKCTRL    |               | <a href="#">Section 8.1.12.1.35</a> |
| C4h    | CM_PER_DCAN1_CLKCTRL    |               | <a href="#">Section 8.1.12.1.36</a> |
| CCh    | CM_PER_EPWMSS1_CLKCTRL  |               | <a href="#">Section 8.1.12.1.37</a> |
| D0h    | CM_PER_EMIF_FW_CLKCTRL  |               | <a href="#">Section 8.1.12.1.38</a> |
| D4h    | CM_PER_EPWMSS0_CLKCTRL  |               | <a href="#">Section 8.1.12.1.39</a> |
| D8h    | CM_PER_EPWMSS2_CLKCTRL  |               | <a href="#">Section 8.1.12.1.40</a> |
| DCh    | CM_PER_L3_INSTR_CLKCTRL |               | <a href="#">Section 8.1.12.1.41</a> |
| E0h    | CM_PER_L3_CLKCTRL       |               | <a href="#">Section 8.1.12.1.42</a> |

**Table 8-27. CM\_PER REGISTERS (continued)**

| <b>Offset</b>   | <b>Acronym</b>             | <b>Register Name</b> | <b>Section</b>      |
|-----------------|----------------------------|----------------------|---------------------|
| E4h             | CM_PER_IIEEE5000_CLKCTRL   |                      | Section 8.1.12.1.43 |
| E8h             | CM_PER_PRU_ICSS_CLKCTRL    |                      | Section 8.1.12.1.44 |
| EC <sub>h</sub> | CM_PER_TIMER5_CLKCTRL      |                      | Section 8.1.12.1.45 |
| F0h             | CM_PER_TIMER6_CLKCTRL      |                      | Section 8.1.12.1.46 |
| F4h             | CM_PER_MMC1_CLKCTRL        |                      | Section 8.1.12.1.47 |
| F8h             | CM_PER_MMC2_CLKCTRL        |                      | Section 8.1.12.1.48 |
| FCh             | CM_PER_TPTC1_CLKCTRL       |                      | Section 8.1.12.1.49 |
| 100h            | CM_PER_TPTC2_CLKCTRL       |                      | Section 8.1.12.1.50 |
| 10Ch            | CM_PER_SPINLOCK_CLKCTRL    |                      | Section 8.1.12.1.51 |
| 110h            | CM_PER_MAILBOX0_CLKCTRL    |                      | Section 8.1.12.1.52 |
| 11Ch            | CM_PER_L4HS_CLKSTCTRL      |                      | Section 8.1.12.1.53 |
| 120h            | CM_PER_L4HS_CLKCTRL        |                      | Section 8.1.12.1.54 |
| 12Ch            | CM_PER_OCPWP_L3_CLKSTCTRL  |                      | Section 8.1.12.1.55 |
| 130h            | CM_PER_OCPWP_CLKCTRL       |                      | Section 8.1.12.1.56 |
| 140h            | CM_PER_PRU_ICSS_CLKSTCTRL  | L                    | Section 8.1.12.1.57 |
| 144h            | CM_PER_CPSW_CLKSTCTRL      |                      | Section 8.1.12.1.58 |
| 148h            | CM_PER_LCDC_CLKSTCTRL      |                      | Section 8.1.12.1.59 |
| 14Ch            | CM_PER_CLKDIV32K_CLKCTRL   |                      | Section 8.1.12.1.60 |
| 150h            | CM_PER_CLK_24MHZ_CLKSTCTRL | RL                   | Section 8.1.12.1.61 |

### 8.1.12.1.1 CM\_PER\_L4LS\_CLKSTCTRL Register (offset = 0h) [reset = C0102h]

CM\_PER\_L4LS\_CLKSTCTRL is shown in [Figure 8-22](#) and described in [Table 8-28](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-PER and ON-INPER states. It also hold one status bit per clock input of the domain.

**Figure 8-22. CM\_PER\_L4LS\_CLKSTCTRL Register**

| 31                          | 30                          | 29                          | 28                          | 27                          | 26                     | 25                       | 24                          |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|------------------------|--------------------------|-----------------------------|
|                             | Reserved                    |                             | CLKACTIVITY_TIME<br>R6_GCLK | CLKACTIVITY_TIME<br>R5_GCLK | Reserved               | CLKACTIVITY_SPI_G<br>CLK | CLKACTIVITY_I2C_F<br>CLK    |
|                             | R-0h                        |                             | R-0h                        | R-0h                        | R-0h                   | R-0h                     | R-0h                        |
| 23                          | 22                          | 21                          | 20                          | 19                          | 18                     | 17                       | 16                          |
| Reserved                    | Reserved                    | CLKACTIVITY_GPIO_3_GDBCLK   | CLKACTIVITY_GPIO_2_GDBCLK   | CLKACTIVITY_GPIO_1_GDBCLK   | Reserved               | CLKACTIVITY_LCDC_GCLK    | CLKACTIVITY_TIME<br>R4_GCLK |
| R-0h                        | R-0h                        | R-0h                        | R-0h                        | R-1h                        | R-1h                   | R-0h                     | R-0h                        |
| 15                          | 14                          | 13                          | 12                          | 11                          | 10                     | 9                        | 8                           |
| CLKACTIVITY_TIME<br>R3_GCLK | CLKACTIVITY_TIME<br>R2_GCLK | CLKACTIVITY_TIME<br>R7_GCLK | Reserved                    | CLKACTIVITY_CAN_CLK         | CLKACTIVITY_UART_GFCLK | Reserved                 | CLKACTIVITY_L4LS_GCLK       |
| R-0h                        | R-0h                        | R-0h                        | R-0h                        | R-0h                        | R-0h                   | R-0h                     | R-1h                        |
| 7                           | 6                           | 5                           | 4                           | 3                           | 2                      | 1                        | 0                           |
|                             | Reserved                    |                             |                             |                             |                        | CLKTRCTRL                |                             |
|                             |                             |                             |                             |                             |                        |                          | R/W-2h                      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-28. CM\_PER\_L4LS\_CLKSTCTRL Register Field Descriptions**

| Bit   | Field                     | Type | Reset | Description   |
|-------|---------------------------|------|-------|---|
| 31-29 | Reserved                  | R    | 0h    |   |
| 28    | CLKACTIVITY_TIMER6_GCLK   | R    | 0h    | This field indicates the state of the TIMER6 CLKTIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active |
| 27    | CLKACTIVITY_TIMER5_GCLK   | R    | 0h    | This field indicates the state of the TIMER5 CLKTIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active |
| 26    | Reserved                  | R    | 0h    | Reserved.   |
| 25    | CLKACTIVITY_SPI_GCLK      | R    | 0h    | This field indicates the state of the SPI_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active        |
| 24    | CLKACTIVITY_I2C_FCLK      | R    | 0h    | This field indicates the state of the I2C_FCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active        |
| 23    | Reserved                  | R    | 0h    |   |
| 22    | Reserved                  | R    | 0h    | Reserved.   |
| 21    | CLKACTIVITY_GPIO_3_GDBCLK | R    | 0h    | This field indicates the state of the GPIO3_GDBCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active    |
| 20    | CLKACTIVITY_GPIO_2_GDBCLK | R    | 0h    | This field indicates the state of the GPIO2_GDBCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active    |

**Table 8-28. CM\_PER\_L4LS\_CLKSTCTRL Register Field Descriptions (continued)**

| Bit | Field                     | Type | Reset | Description   |
|-----|---------------------------|------|-------|---|
| 19  | CLKACTIVITY_GPIO_1_GDBCLK | R    | 1h    | This field indicates the state of the GPIO1_GDBCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 18  | Reserved                  | R    | 1h    | Reserved.   |
| 17  | CLKACTIVITY_LCDC_GC_LK    | R    | 0h    | This field indicates the state of the LCD clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 16  | CLKACTIVITY_TIMER4_GCLK   | R    | 0h    | This field indicates the state of the TIMER4 CLKTIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 15  | CLKACTIVITY_TIMER3_GCLK   | R    | 0h    | This field indicates the state of the TIMER3 CLKIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 14  | CLKACTIVITY_TIMER2_GCLK   | R    | 0h    | This field indicates the state of the TIMER2 CLKIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 13  | CLKACTIVITY_TIMER7_GCLK   | R    | 0h    | This field indicates the state of the TIMER7 CLKIMER clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 12  | Reserved                  | R    | 0h    |   |
| 11  | CLKACTIVITY_CAN_CLK       | R    | 0h    | This field indicates the state of the CAN_CLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 10  | CLKACTIVITY_UART_GF_CLK   | R    | 0h    | This field indicates the state of the UART_GFCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 9   | Reserved                  | R    | 0h    |   |
| 8   | CLKACTIVITY_L4LS_GC_LK    | R    | 1h    | This field indicates the state of the L4LS_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 7-2 | Reserved                  | R    | 0h    |   |
| 1-0 | CLKTRCTRL                 | R/W  | 2h    | Controls the clock state transition of the L4 SLOW clock domain in PER power domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.2 CM\_PER\_L3S\_CLKSTCTRL Register (offset = 4h) [reset = Ah]

CM\_PER\_L3S\_CLKSTCTRL is shown in [Figure 8-23](#) and described in [Table 8-29](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-23. CM\_PER\_L3S\_CLKSTCTRL Register**

|          |    |    |          |                      |          |           |          |
|----------|----|----|----------|----------------------|----------|-----------|----------|
| 31       | 30 | 29 | 28       | 27                   | 26       | 25        | 24       |
| Reserved |    |    |          |                      |          | Reserved  |          |
| R-0h     |    |    |          |                      |          | R-0h      |          |
| 23       | 22 | 21 | 20       | 19                   | 18       | 17        | 16       |
| Reserved |    |    |          |                      |          | R-0h      |          |
| 15       | 14 | 13 | 12       | 11                   | 10       | 9         | 8        |
| Reserved |    |    |          |                      |          | Reserved  | Reserved |
| R-0h     |    |    |          |                      |          | R-0h      | R-0h     |
| 7        | 6  | 5  | 4        | 3                    | 2        | 1         | 0        |
| Reserved |    |    | Reserved | CLKACTIVITY_L3S_GCLK | Reserved | CLKTRCTRL |          |
| R-0h     |    |    | R-0h     | R-1h                 | R-0h     | R/W-2h    |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-29. CM\_PER\_L3S\_CLKSTCTRL Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description   |
|-------|----------------------|------|-------|---|
| 31-26 | Reserved             | R    | 0h    |   |
| 25-11 | Reserved             | R    | 0h    |   |
| 10    | Reserved             | R    | 0h    |   |
| 9     | Reserved             | R    | 0h    |   |
| 8     | Reserved             | R    | 0h    |   |
| 7-5   | Reserved             | R    | 0h    |   |
| 4     | Reserved             | R    | 0h    |   |
| 3     | CLKACTIVITY_L3S_GCLK | R    | 1h    | This field indicates the state of the L3S_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 2     | Reserved             | R    | 0h    |   |
| 1-0   | CLKTRCTRL            | R/W  | 2h    | Controls the clock state transition of the L3 Slow clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.3 CM\_PER\_L4FW\_CLKSTCTRL Register (offset = 8h) [reset = 102h]

CM\_PER\_L4FW\_CLKSTCTRL is shown in [Figure 8-24](#) and described in [Table 8-30](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-24. CM\_PER\_L4FW\_CLKSTCTRL Register**

|          |    |    |    |    |    |           |                       |
|----------|----|----|----|----|----|-----------|-----------------------|
| 31       | 30 | 29 | 28 | 27 | 26 | 25        | 24                    |
| Reserved |    |    |    |    |    | Reserved  |                       |
| R-0h     |    |    |    |    |    | R-0h      |                       |
| 23       | 22 | 21 | 20 | 19 | 18 | 17        | 16                    |
| Reserved |    |    |    |    |    | R-0h      |                       |
| 15       | 14 | 13 | 12 | 11 | 10 | 9         | 8                     |
| Reserved |    |    |    |    |    | Reserved  | CLKACTIVITY_L4FW_GCLK |
| R-0h     |    |    |    |    |    | R-0h      | R-0h                  |
| 7        | 6  | 5  | 4  | 3  | 2  | 1         | 0                     |
| Reserved |    |    |    |    |    | CLKTRCTRL |                       |
| R-0h     |    |    |    |    |    | R/W-2h    |                       |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-30. CM\_PER\_L4FW\_CLKSTCTRL Register Field Descriptions**

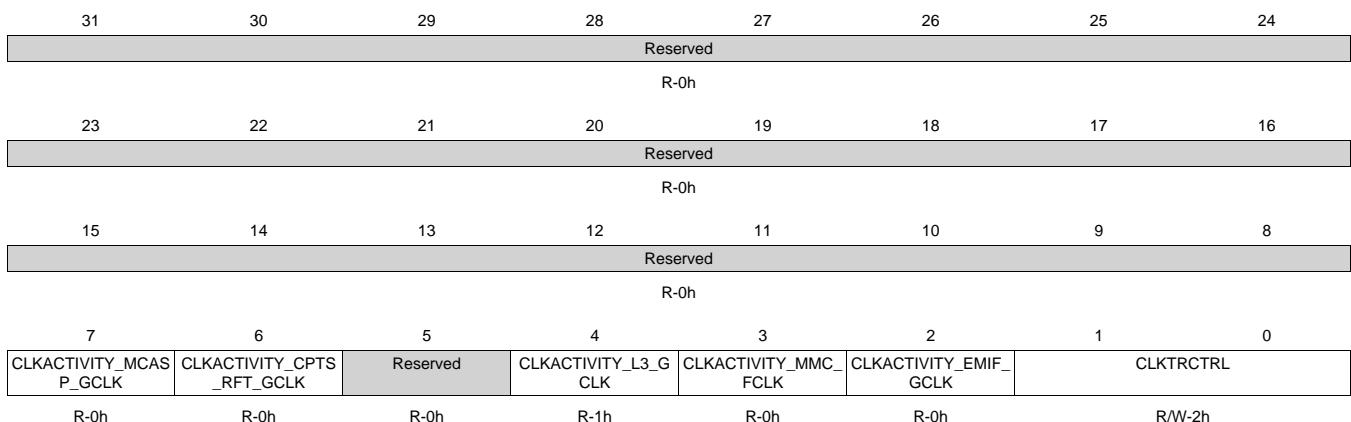
| Bit   | Field                  | Type | Reset | Description  |
|-------|------------------------|------|-------|--|
| 31-26 | Reserved               | R    | 0h    |  |
| 25-11 | Reserved               | R    | 0h    |  |
| 10    | Reserved               | R    | 0h    |  |
| 9     | Reserved               | R    | 0h    |  |
| 8     | CLKACTIVITY_L4FW_GC_LK | R    | 1h    | This field indicates the state of the L4FW clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 7-2   | Reserved               | R    | 0h    |  |
| 1-0   | CLKTRCTRL              | R/W  | 2h    | Controls the clock state transition of the L4 Firewall clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated.<br>Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved |

#### 8.1.12.1.4 CM\_PER\_L3\_CLKSTCTRL Register (offset = Ch) [reset = 12h]

CM\_PER\_L3\_CLKSTCTRL is shown in [Figure 8-25](#) and described in [Table 8-31](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-25. CM\_PER\_L3\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-31. CM\_PER\_L3\_CLKSTCTRL Register Field Descriptions**

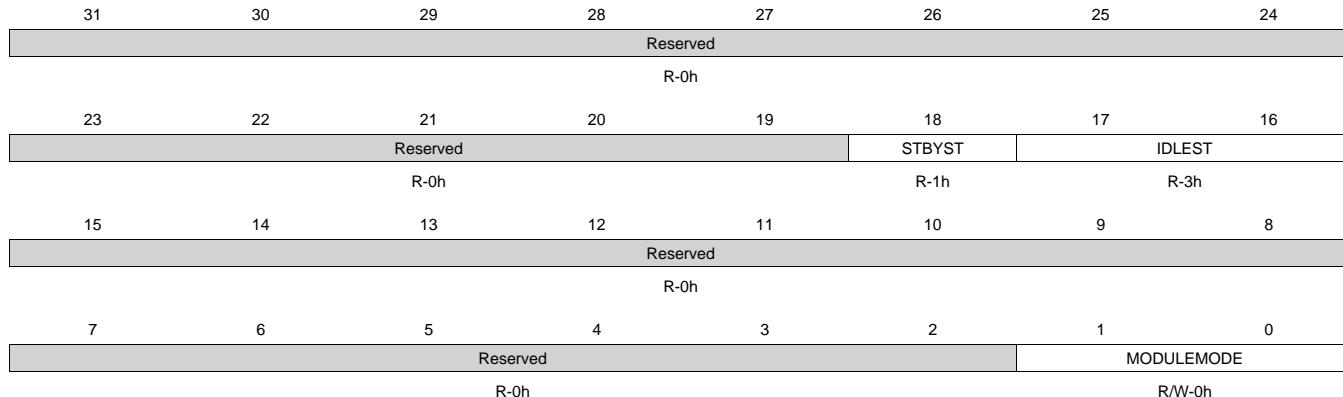
| Bit  | Field                    | Type | Reset | Description  |
|------|--------------------------|------|-------|--|
| 31-8 | Reserved                 | R    | 0h    |  |
| 7    | CLKACTIVITY_MCASP_GCLK   | R    | 0h    | This field indicates the state of the MCASP_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 6    | CLKACTIVITY_CPTS_RF_GCLK | R    | 0h    | This field indicates the state of the CLKACTIVITY_CPTS_RFT_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 5    | Reserved                 | R    | 0h    |  |
| 4    | CLKACTIVITY_L3_GCLK      | R    | 1h    | This field indicates the state of the L3_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 3    | CLKACTIVITY_MMC_FCLK     | R    | 0h    | This field indicates the state of the MMC_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 2    | CLKACTIVITY_EMIF_GCLK    | R    | 0h    | This field indicates the state of the EMIF_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 1-0  | CLKTRCTRL                | R/W  | 2h    | Controls the clock state transition of the L3 clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.5 CM\_PER\_CPGMAC0\_CLKCTRL Register (offset = 14h) [reset = 70000h]

CM\_PER\_CPGMAC0\_CLKCTRL is shown in [Figure 8-26](#) and described in [Table 8-32](#).

This register manages the CPSW clocks.

**Figure 8-26. CM\_PER\_CPGMAC0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-32. CM\_PER\_CPGMAC0\_CLKCTRL Register Field Descriptions**

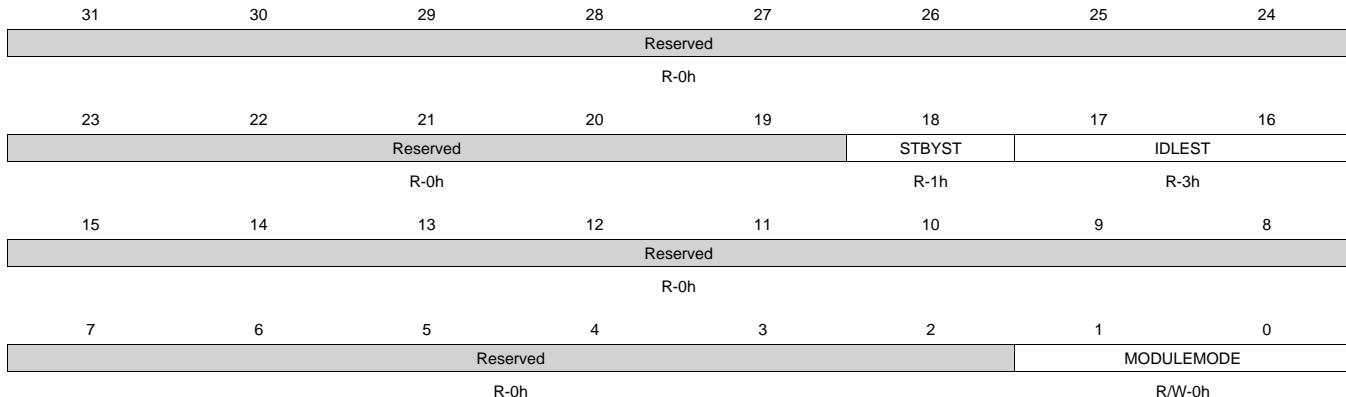
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-19 | Reserved   | R    | 0h    |  |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>This bit is warm reset insensitive when CPSW RESET_ISO is enabled<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>This bit is warm reset insensitive when CPSW RESET_ISO is enabled<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>This bit is warm reset insensitive when CPSW RESET_ISO is enabled<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.6 CM\_PER\_LCDC\_CLKCTRL Register (offset = 18h) [reset = 70000h]

CM\_PER\_LCDC\_CLKCTRL is shown in [Figure 8-27](#) and described in [Table 8-33](#).

This register manages the LCD clocks.

**Figure 8-27. CM\_PER\_LCDC\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-33. CM\_PER\_LCDC\_CLKCTRL Register Field Descriptions**

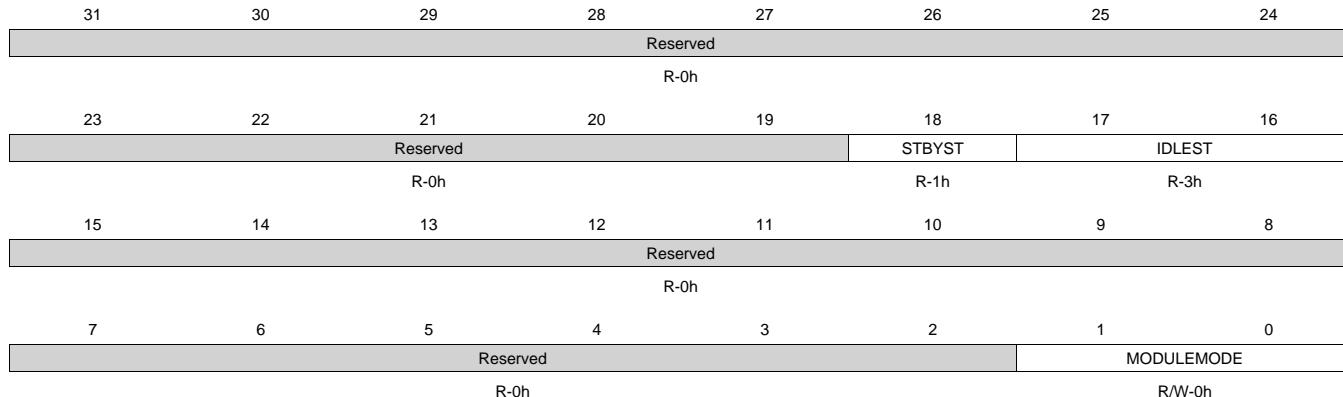
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.7 CM\_PER\_USB0\_CLKCTRL Register (offset = 1Ch) [reset = 70000h]

CM\_PER\_USB0\_CLKCTRL is shown in [Figure 8-28](#) and described in [Table 8-34](#).

This register manages the USB clocks.

**Figure 8-28. CM\_PER\_USB0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-34. CM\_PER\_USB0\_CLKCTRL Register Field Descriptions**

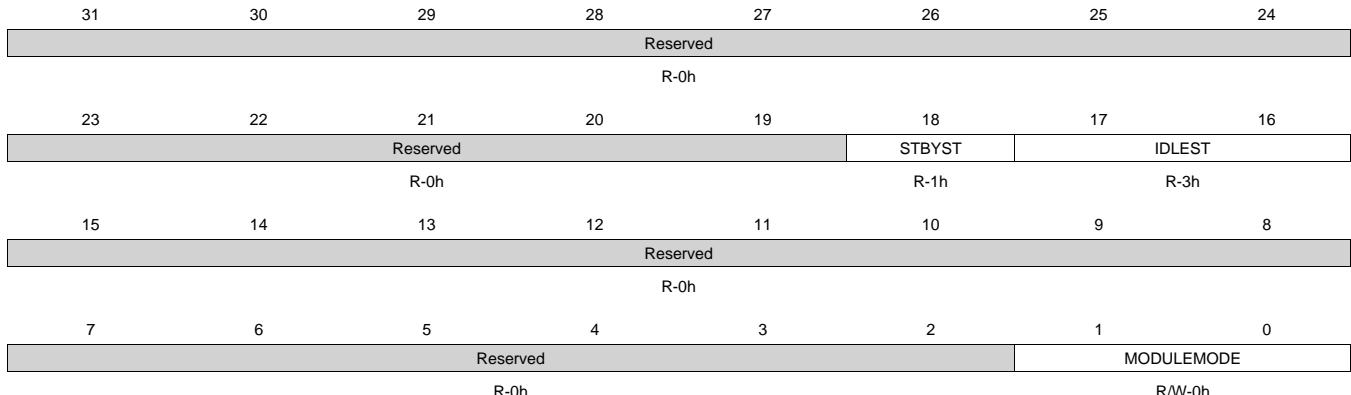
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.8 CM\_PER\_TPTC0\_CLKCTRL Register (offset = 24h) [reset = 70000h]

CM\_PER\_TPTC0\_CLKCTRL is shown in [Figure 8-29](#) and described in [Table 8-35](#).

This register manages the TPTC clocks.

**Figure 8-29. CM\_PER\_TPTC0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-35. CM\_PER\_TPTC0\_CLKCTRL Register Field Descriptions**

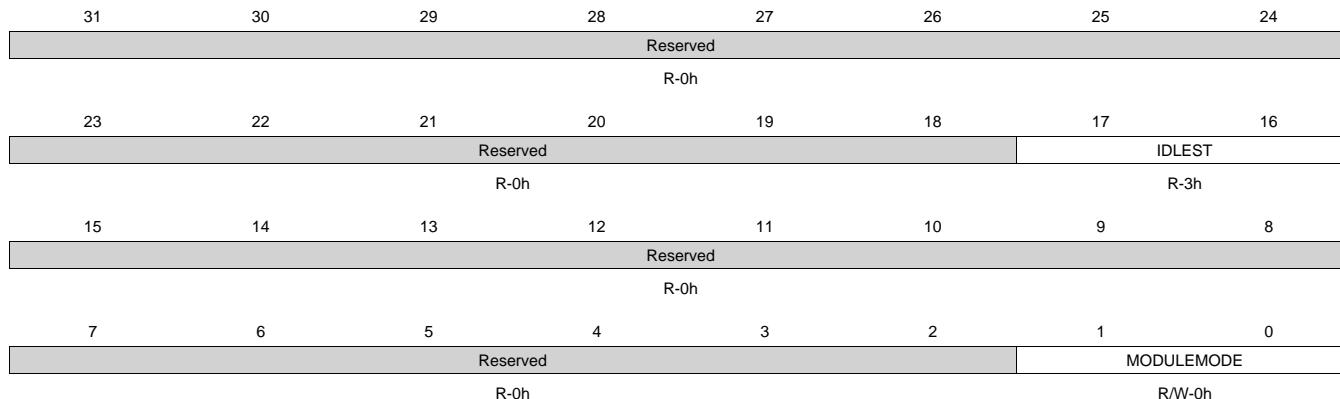
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.9 CM\_PER\_EMIF\_CLKCTRL Register (offset = 28h) [reset = 30000h]

CM\_PER\_EMIF\_CLKCTRL is shown in [Figure 8-30](#) and described in [Table 8-36](#).

This register manages the EMIF clocks.

**Figure 8-30. CM\_PER\_EMIF\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-36. CM\_PER\_EMIF\_CLKCTRL Register Field Descriptions**

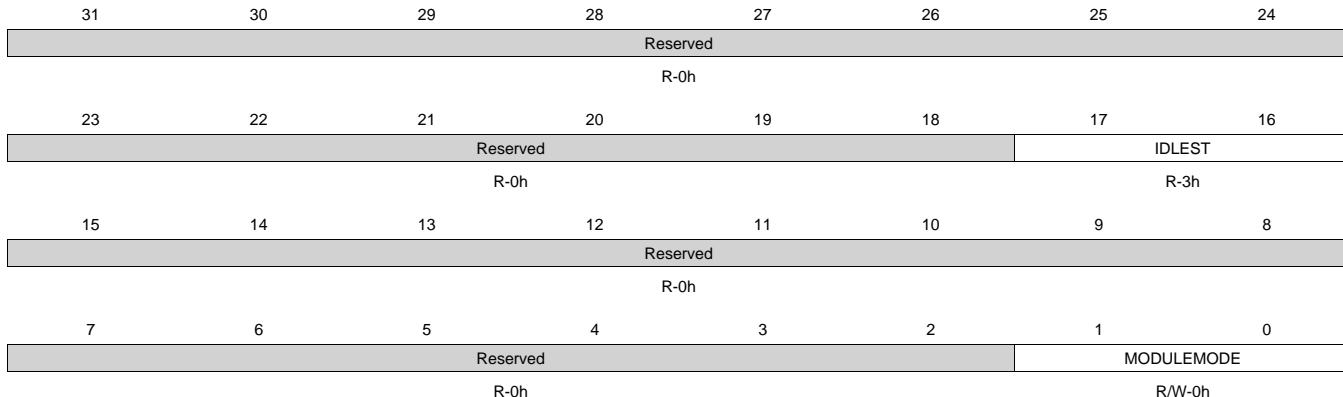
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.10 CM\_PER\_OCMCRAM\_CLKCTRL Register (offset = 2Ch) [reset = 30000h]

CM\_PER\_OCMCRAM\_CLKCTRL is shown in [Figure 8-31](#) and described in [Table 8-37](#).

This register manages the OCMC clocks.

**Figure 8-31. CM\_PER\_OCMCRAM\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-37. CM\_PER\_OCMCRAM\_CLKCTRL Register Field Descriptions**

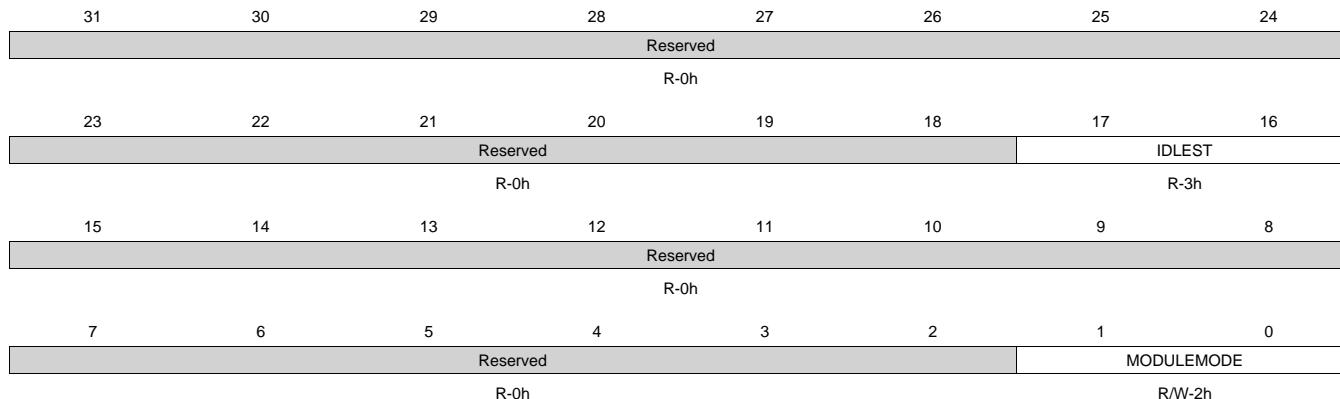
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.11 CM\_PER\_GPMC\_CLKCTRL Register (offset = 30h) [reset = 30002h]

CM\_PER\_GPMC\_CLKCTRL is shown in [Figure 8-32](#) and described in [Table 8-38](#).

This register manages the GPMC clocks.

**Figure 8-32. CM\_PER\_GPMC\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-38. CM\_PER\_GPMC\_CLKCTRL Register Field Descriptions**

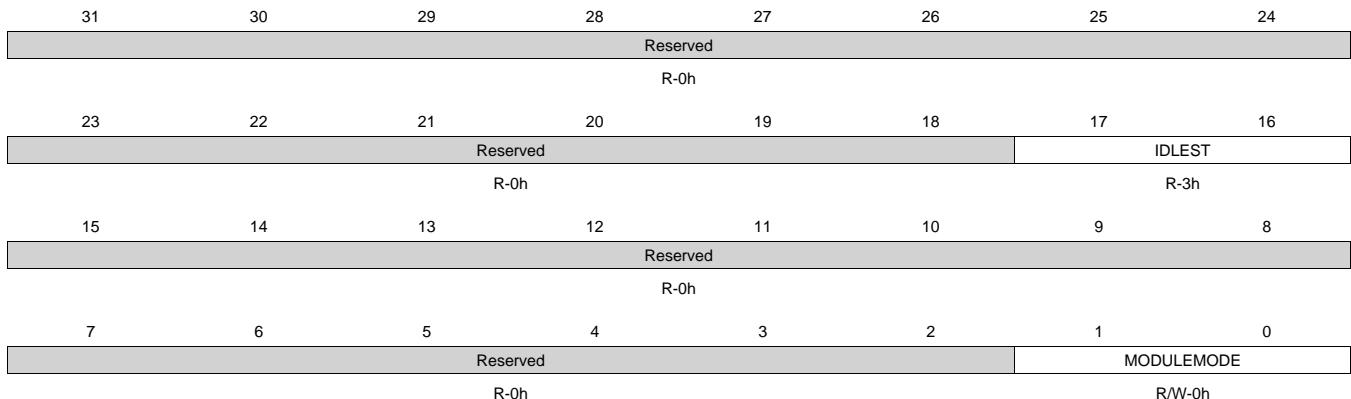
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.12 CM\_PER\_MCASP0\_CLKCTRL Register (offset = 34h) [reset = 30000h]

CM\_PER\_MCASP0\_CLKCTRL is shown in [Figure 8-33](#) and described in [Table 8-39](#).

This register manages the MCASP0 clocks.

**Figure 8-33. CM\_PER\_MCASP0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-39. CM\_PER\_MCASP0\_CLKCTRL Register Field Descriptions**

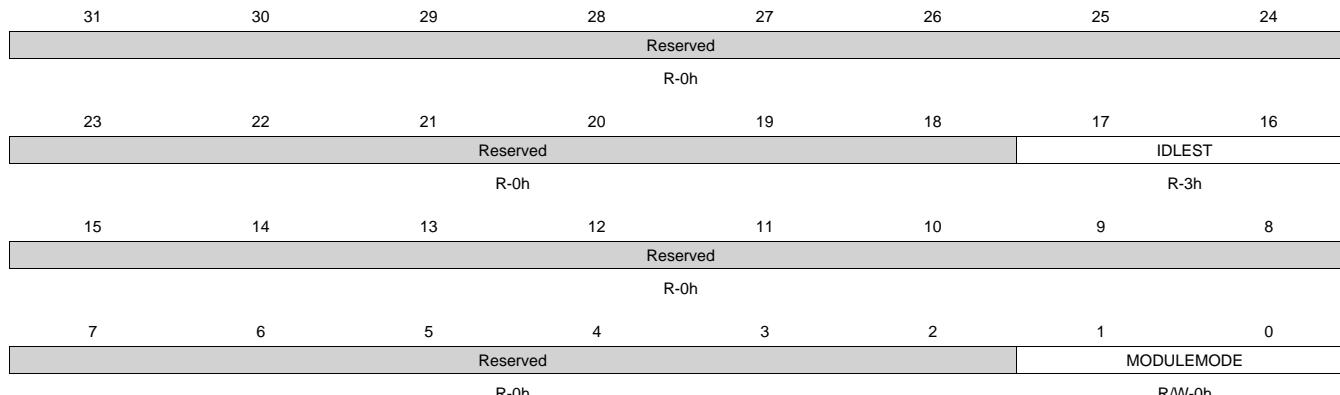
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.13 CM\_PER\_UART5\_CLKCTRL Register (offset = 38h) [reset = 30000h]

CM\_PER\_UART5\_CLKCTRL is shown in [Figure 8-34](#) and described in [Table 8-40](#).

This register manages the UART5 clocks.

**Figure 8-34. CM\_PER\_UART5\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-40. CM\_PER\_UART5\_CLKCTRL Register Field Descriptions**

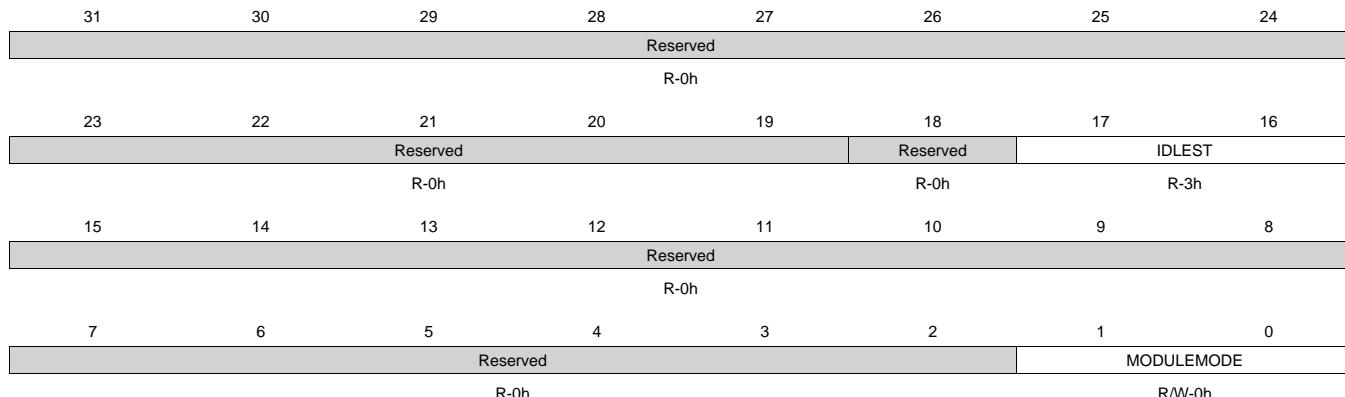
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.14 CM\_PER\_MMC0\_CLKCTRL Register (offset = 3Ch) [reset = 30000h]

CM\_PER\_MMC0\_CLKCTRL is shown in [Figure 8-35](#) and described in [Table 8-41](#).

This register manages the MMC0 clocks.

**Figure 8-35. CM\_PER\_MMC0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-41. CM\_PER\_MMC0\_CLKCTRL Register Field Descriptions**

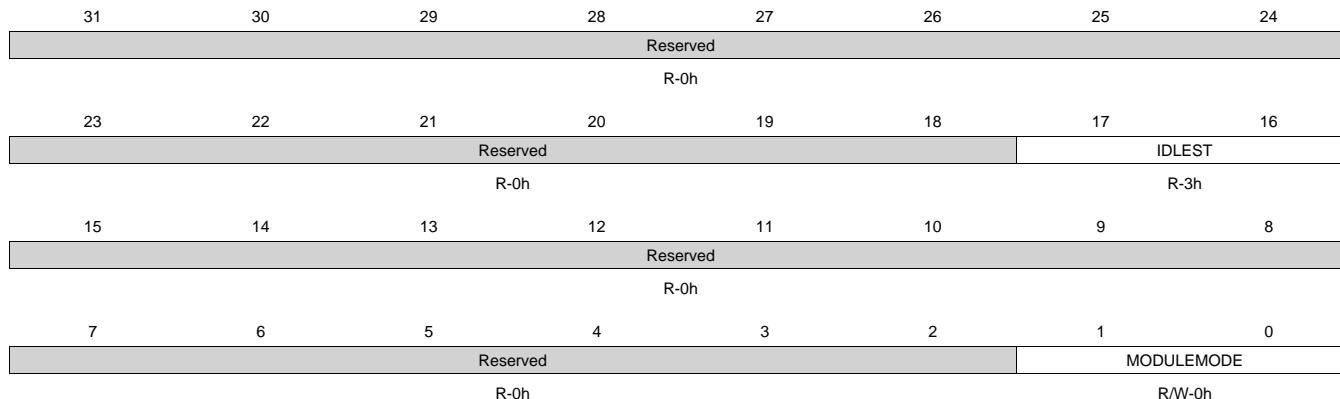
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.15 CM\_PER\_ELM\_CLKCTRL Register (offset = 40h) [reset = 30000h]

CM\_PER\_ELM\_CLKCTRL is shown in [Figure 8-36](#) and described in [Table 8-42](#).

This register manages the ELM clocks.

**Figure 8-36. CM\_PER\_ELM\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-42. CM\_PER\_ELM\_CLKCTRL Register Field Descriptions**

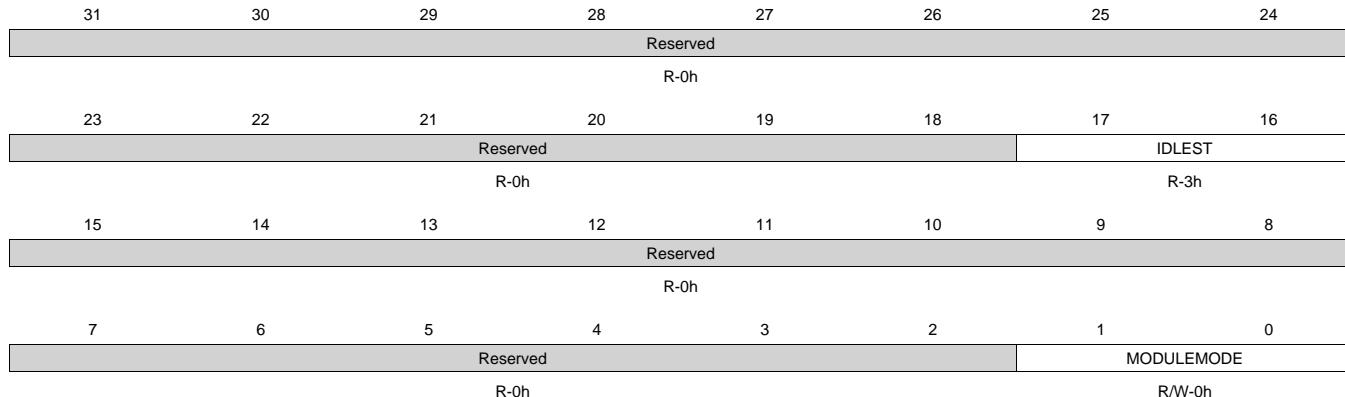
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.16 CM\_PER\_I2C2\_CLKCTRL Register (offset = 44h) [reset = 30000h]

CM\_PER\_I2C2\_CLKCTRL is shown in [Figure 8-37](#) and described in [Table 8-43](#).

This register manages the I2C2 clocks.

**Figure 8-37. CM\_PER\_I2C2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-43. CM\_PER\_I2C2\_CLKCTRL Register Field Descriptions**

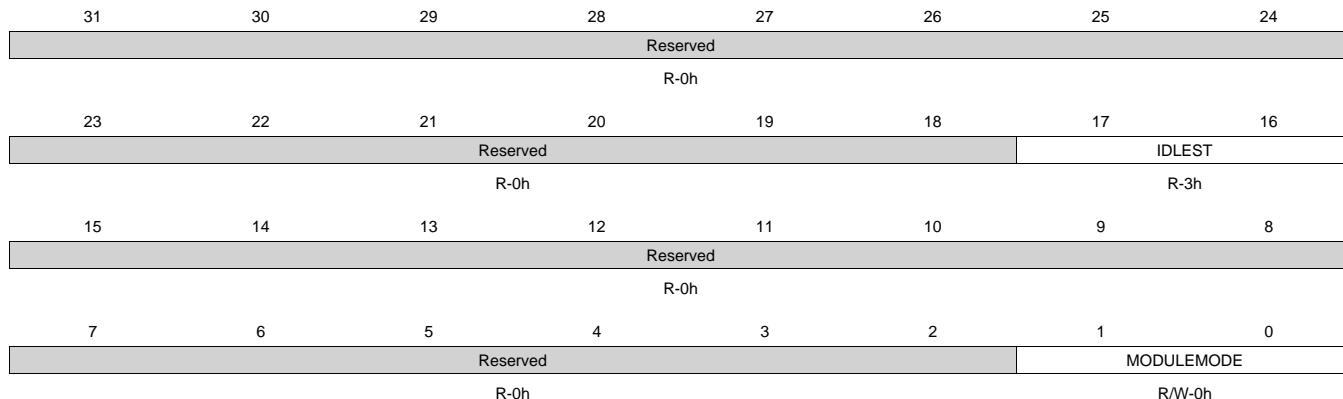
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.17 CM\_PER\_I2C1\_CLKCTRL Register (offset = 48h) [reset = 30000h]

CM\_PER\_I2C1\_CLKCTRL is shown in [Figure 8-38](#) and described in [Table 8-44](#).

This register manages the I2C1 clocks.

**Figure 8-38. CM\_PER\_I2C1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-44. CM\_PER\_I2C1\_CLKCTRL Register Field Descriptions**

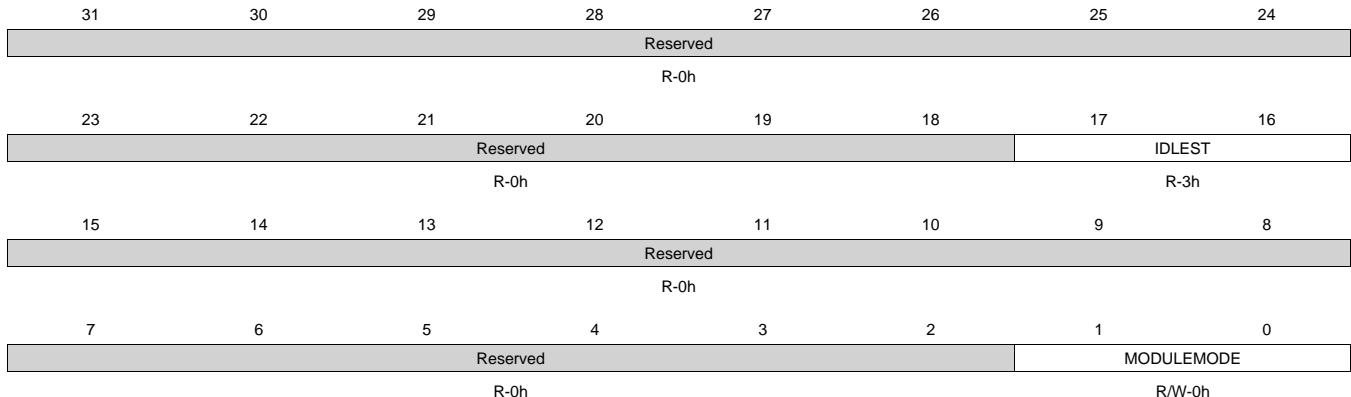
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.18 CM\_PER\_SPI0\_CLKCTRL Register (offset = 4Ch) [reset = 30000h]

CM\_PER\_SPI0\_CLKCTRL is shown in [Figure 8-39](#) and described in [Table 8-45](#).

This register manages the SPI0 clocks.

**Figure 8-39. CM\_PER\_SPI0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-45. CM\_PER\_SPI0\_CLKCTRL Register Field Descriptions**

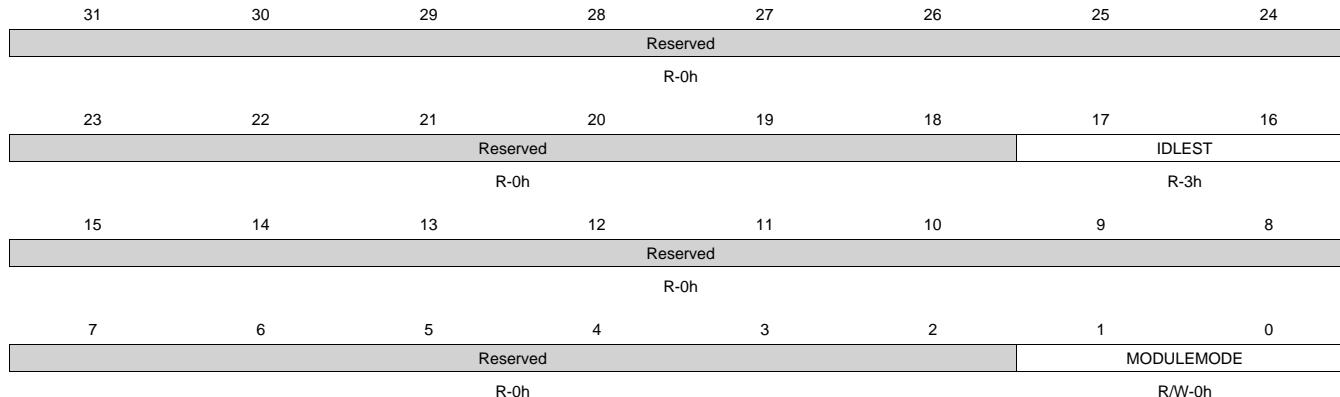
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.19 CM\_PER\_SPI1\_CLKCTRL Register (offset = 50h) [reset = 30000h]

CM\_PER\_SPI1\_CLKCTRL is shown in [Figure 8-40](#) and described in [Table 8-46](#).

This register manages the SPI1 clocks.

**Figure 8-40. CM\_PER\_SPI1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-46. CM\_PER\_SPI1\_CLKCTRL Register Field Descriptions**

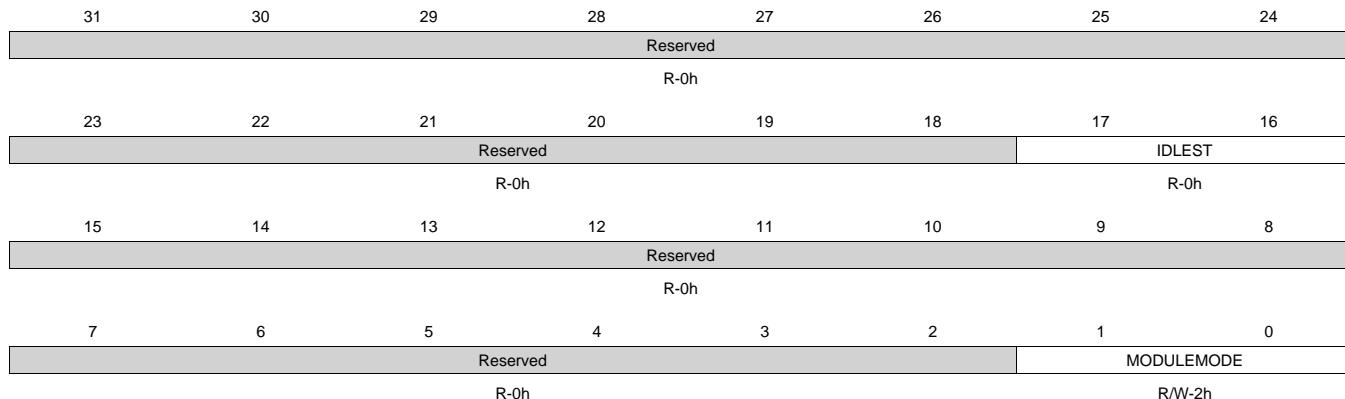
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.20 CM\_PER\_L4LS\_CLKCTRL Register (offset = 60h) [reset = 2h]

CM\_PER\_L4LS\_CLKCTRL is shown in [Figure 8-41](#) and described in [Table 8-47](#).

This register manages the L4LS clocks.

**Figure 8-41. CM\_PER\_L4LS\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-47. CM\_PER\_L4LS\_CLKCTRL Register Field Descriptions**

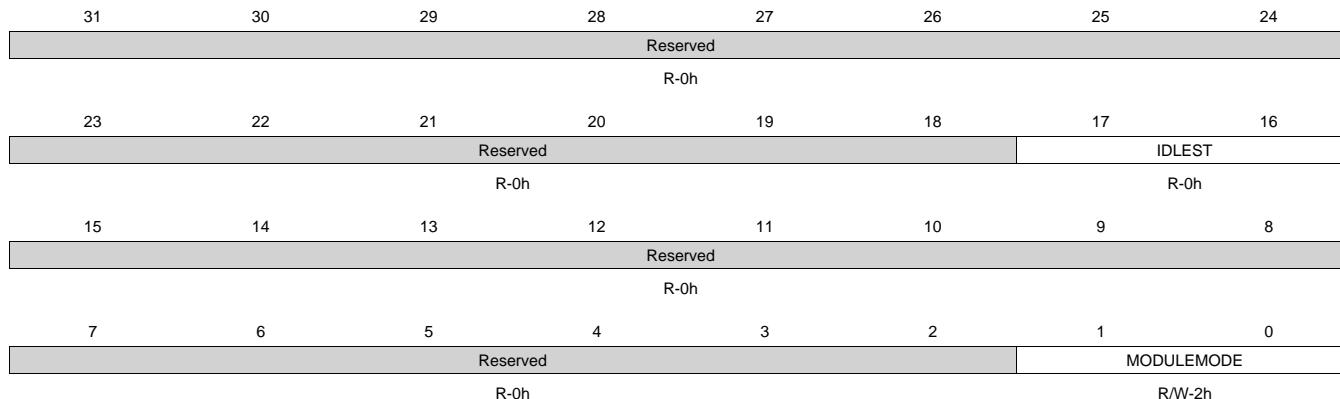
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 0h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 2h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.21 CM\_PER\_L4FW\_CLKCTRL Register (offset = 64h) [reset = 2h]

CM\_PER\_L4FW\_CLKCTRL is shown in [Figure 8-42](#) and described in [Table 8-48](#).

This register manages the L4FW clocks.

**Figure 8-42. CM\_PER\_L4FW\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-48. CM\_PER\_L4FW\_CLKCTRL Register Field Descriptions**

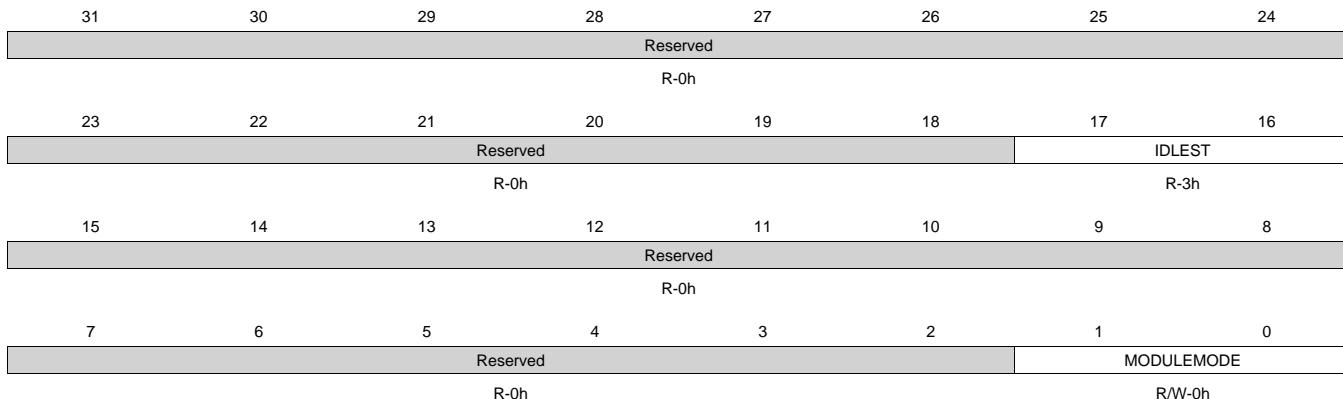
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.22 CM\_PER\_MCASP1\_CLKCTRL Register (offset = 68h) [reset = 30000h]

CM\_PER\_MCASP1\_CLKCTRL is shown in [Figure 8-43](#) and described in [Table 8-49](#).

This register manages the MCASP1 clocks.

**Figure 8-43. CM\_PER\_MCASP1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-49. CM\_PER\_MCASP1\_CLKCTRL Register Field Descriptions**

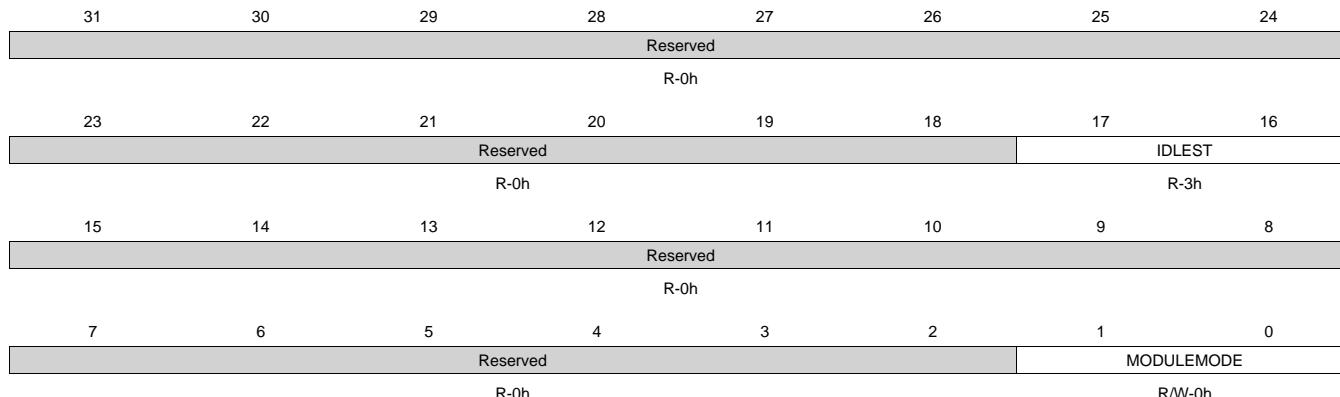
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.23 CM\_PER\_UART1\_CLKCTRL Register (offset = 6Ch) [reset = 30000h]

CM\_PER\_UART1\_CLKCTRL is shown in [Figure 8-44](#) and described in [Table 8-50](#).

This register manages the UART1 clocks.

**Figure 8-44. CM\_PER\_UART1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-50. CM\_PER\_UART1\_CLKCTRL Register Field Descriptions**

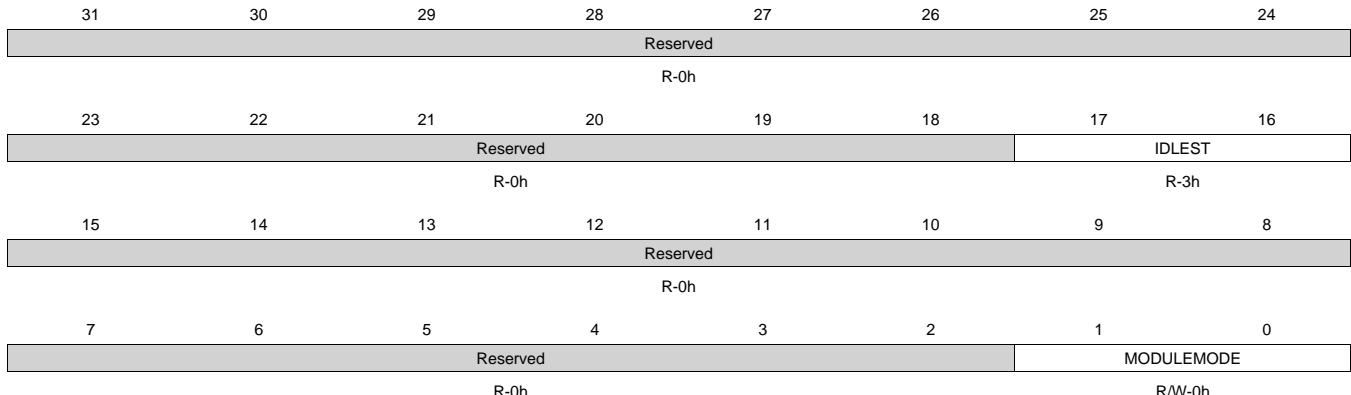
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.24 CM\_PER\_UART2\_CLKCTRL Register (offset = 70h) [reset = 30000h]

CM\_PER\_UART2\_CLKCTRL is shown in [Figure 8-45](#) and described in [Table 8-51](#).

This register manages the UART2 clocks.

**Figure 8-45. CM\_PER\_UART2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-51. CM\_PER\_UART2\_CLKCTRL Register Field Descriptions**

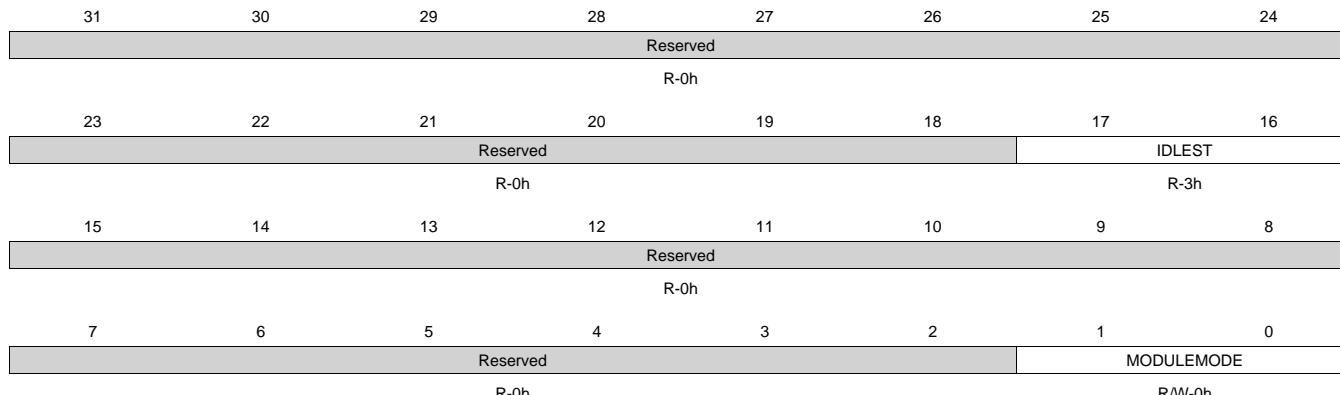
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.25 CM\_PER\_UART3\_CLKCTRL Register (offset = 74h) [reset = 30000h]

CM\_PER\_UART3\_CLKCTRL is shown in [Figure 8-46](#) and described in [Table 8-52](#).

This register manages the UART3 clocks.

**Figure 8-46. CM\_PER\_UART3\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-52. CM\_PER\_UART3\_CLKCTRL Register Field Descriptions**

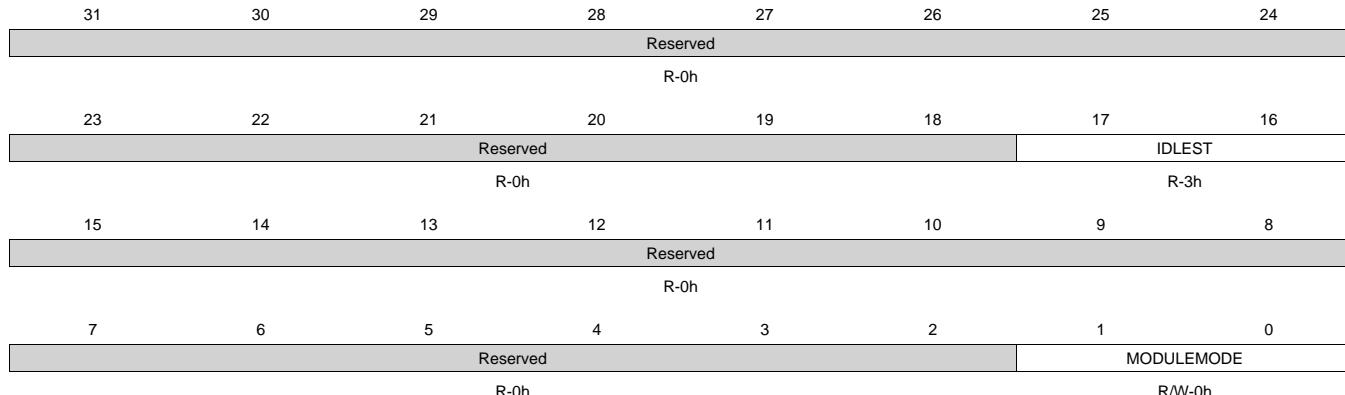
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.26 CM\_PER\_UART4\_CLKCTRL Register (offset = 78h) [reset = 30000h]

CM\_PER\_UART4\_CLKCTRL is shown in [Figure 8-47](#) and described in [Table 8-53](#).

This register manages the UART4 clocks.

**Figure 8-47. CM\_PER\_UART4\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-53. CM\_PER\_UART4\_CLKCTRL Register Field Descriptions**

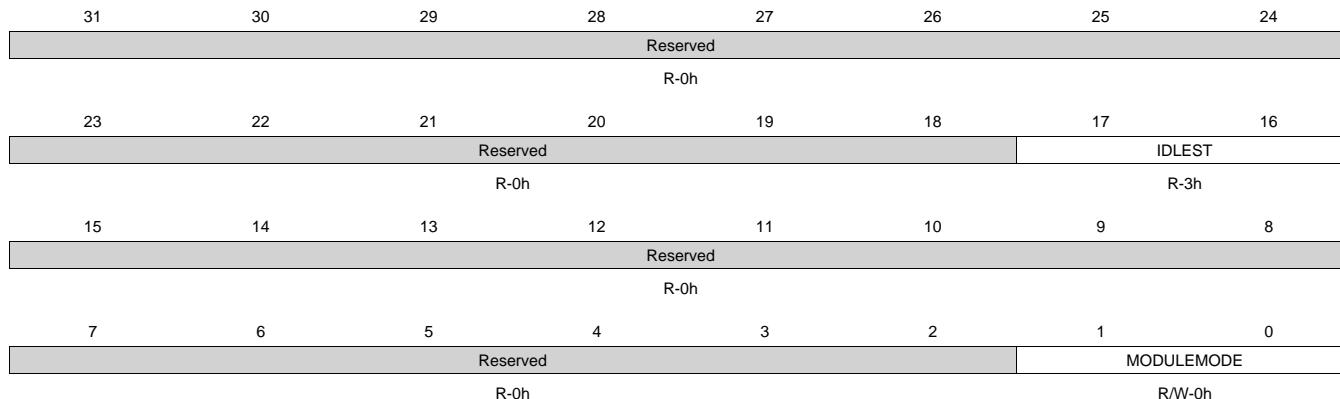
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.27 CM\_PER\_TIMER7\_CLKCTRL Register (offset = 7Ch) [reset = 30000h]

CM\_PER\_TIMER7\_CLKCTRL is shown in [Figure 8-48](#) and described in [Table 8-54](#).

This register manages the TIMER7 clocks.

**Figure 8-48. CM\_PER\_TIMER7\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-54. CM\_PER\_TIMER7\_CLKCTRL Register Field Descriptions**

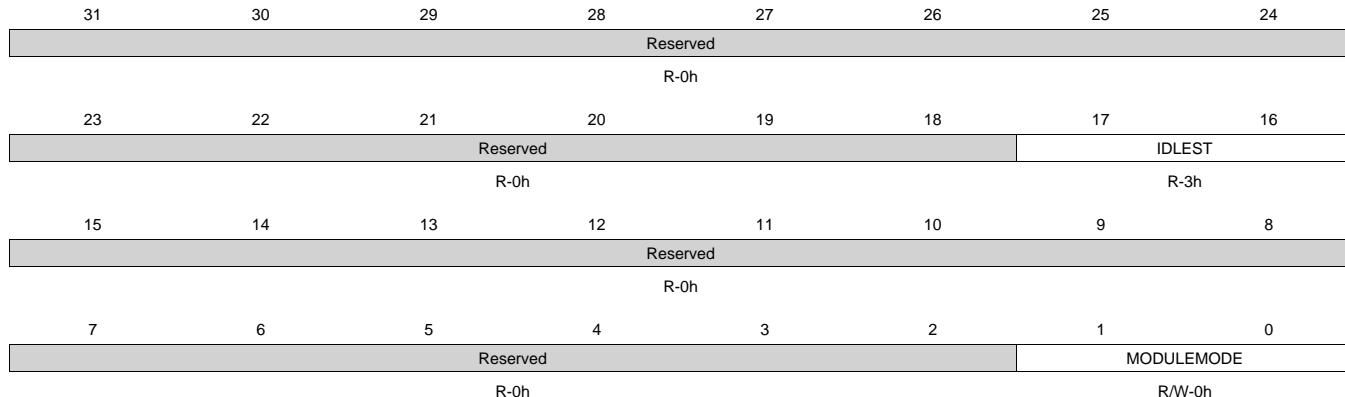
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.28 CM\_PER\_TIMER2\_CLKCTRL Register (offset = 80h) [reset = 30000h]

CM\_PER\_TIMER2\_CLKCTRL is shown in [Figure 8-49](#) and described in [Table 8-55](#).

This register manages the TIMER2 clocks.

**Figure 8-49. CM\_PER\_TIMER2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-55. CM\_PER\_TIMER2\_CLKCTRL Register Field Descriptions**

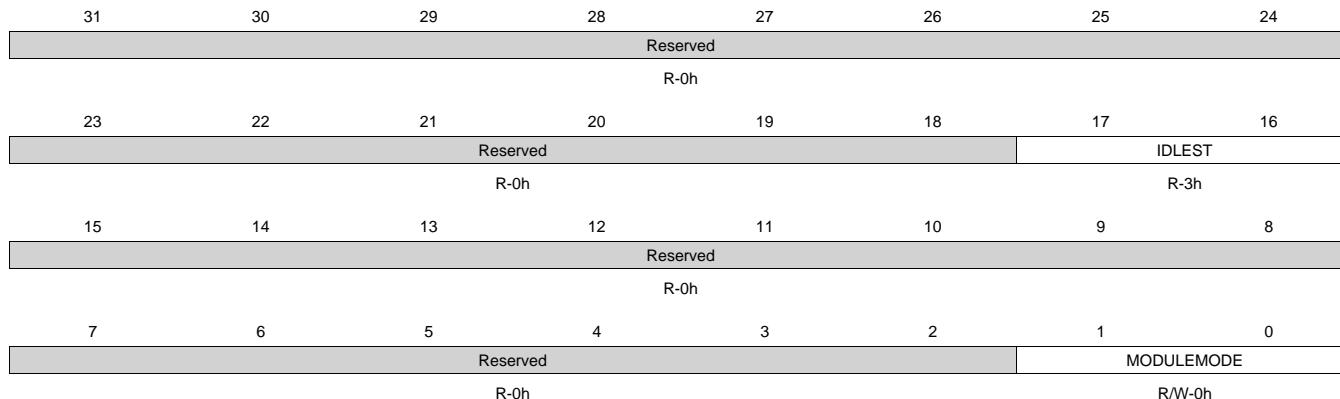
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.29 CM\_PER\_TIMER3\_CLKCTRL Register (offset = 84h) [reset = 30000h]

CM\_PER\_TIMER3\_CLKCTRL is shown in [Figure 8-50](#) and described in [Table 8-56](#).

This register manages the TIMER3 clocks.

**Figure 8-50. CM\_PER\_TIMER3\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-56. CM\_PER\_TIMER3\_CLKCTRL Register Field Descriptions**

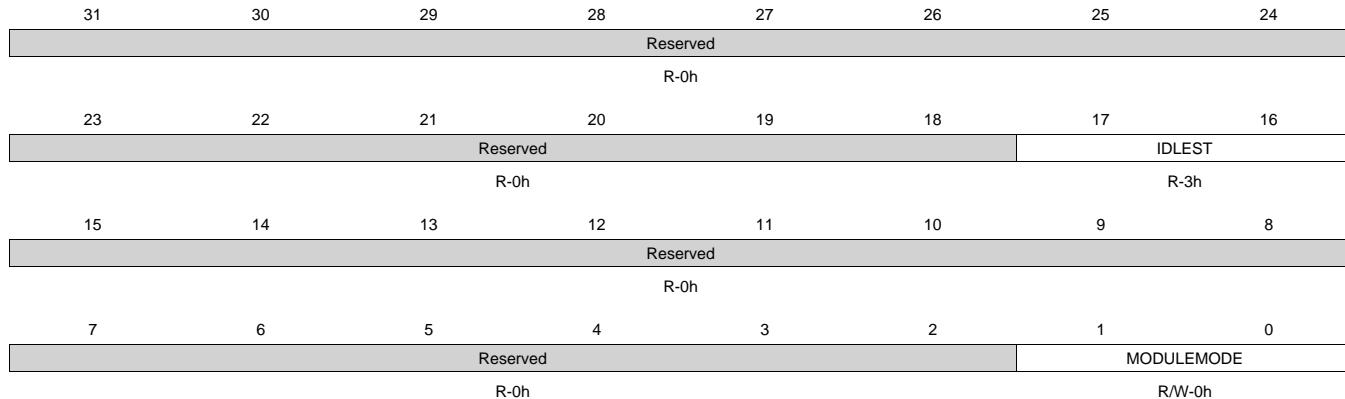
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.30 CM\_PER\_TIMER4\_CLKCTRL Register (offset = 88h) [reset = 30000h]

CM\_PER\_TIMER4\_CLKCTRL is shown in [Figure 8-51](#) and described in [Table 8-57](#).

This register manages the TIMER4 clocks.

**Figure 8-51. CM\_PER\_TIMER4\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-57. CM\_PER\_TIMER4\_CLKCTRL Register Field Descriptions**

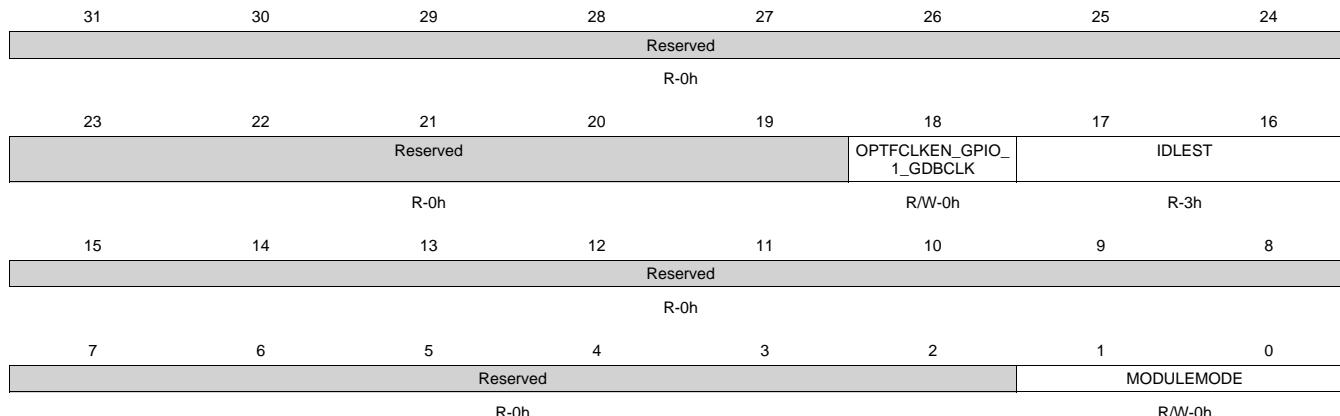
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.31 CM\_PER\_GPIO1\_CLKCTRL Register (offset = ACh) [reset = 30000h]

CM\_PER\_GPIO1\_CLKCTRL is shown in [Figure 8-52](#) and described in [Table 8-58](#).

This register manages the GPIO1 clocks.

**Figure 8-52. CM\_PER\_GPIO1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-58. CM\_PER\_GPIO1\_CLKCTRL Register Field Descriptions**

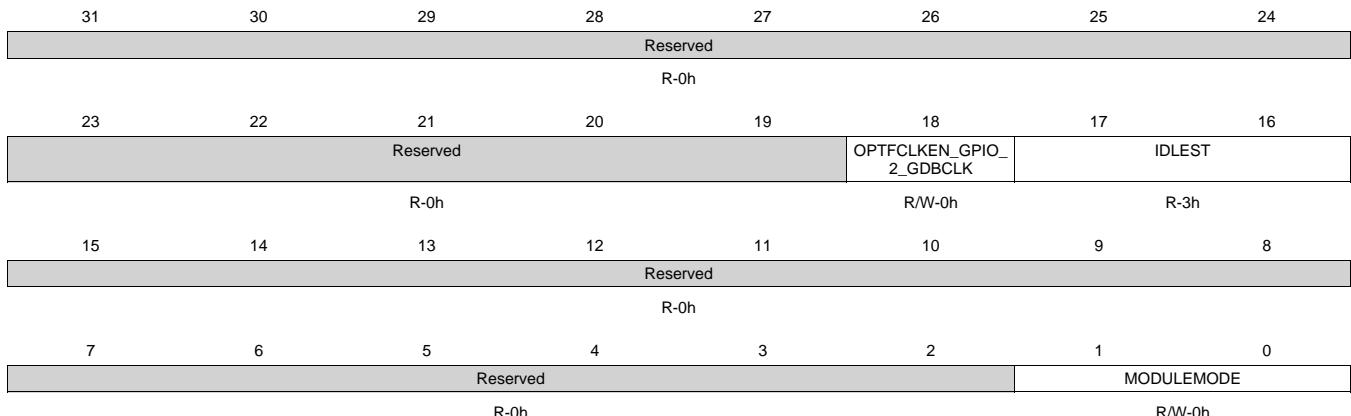
| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-19 | Reserved                | R    | 0h    |   |
| 18    | OPTFCLKEN_GPIO_1_GDBCLK | R/W  | 0h    | Optional functional clock control.<br>0x0 = FCLK_DIS : Optional functional clock is disabled<br>0x1 = FCLK_EN : Optional functional clock is enabled  |
| 17-16 | IDLEST                  | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved                | R    | 0h    |   |
| 1-0   | MODULEMODE              | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.32 CM\_PER\_GPIO2\_CLKCTRL Register (offset = B0h) [reset = 30000h]

CM\_PER\_GPIO2\_CLKCTRL is shown in [Figure 8-53](#) and described in [Table 8-59](#).

This register manages the GPIO2 clocks.

**Figure 8-53. CM\_PER\_GPIO2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-59. CM\_PER\_GPIO2\_CLKCTRL Register Field Descriptions**

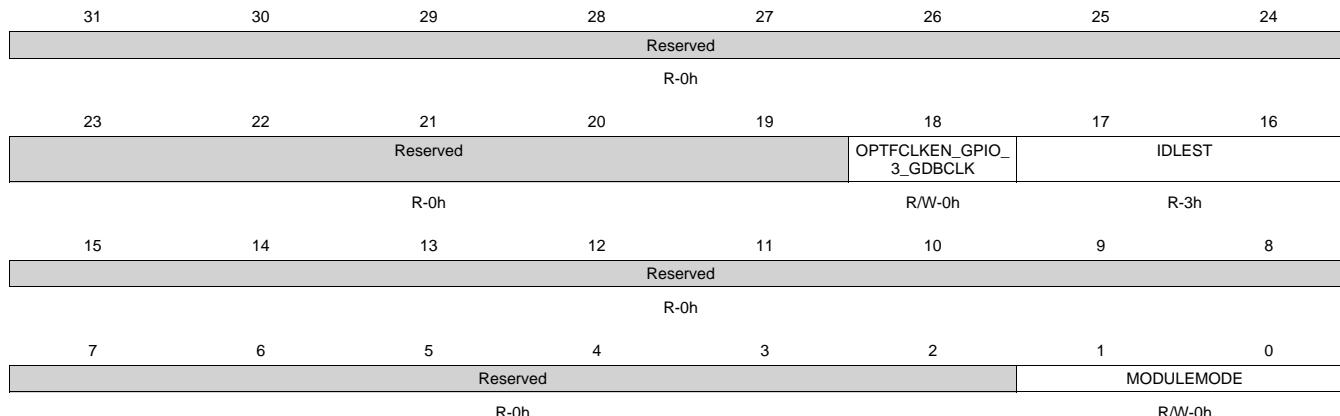
| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-19 | Reserved                | R    | 0h    |   |
| 18    | OPTFCLKEN_GPIO_2_GDBCLK | R/W  | 0h    | Optional functional clock control.<br>0x0 = FCLK_DIS : Optional functional clock is disabled<br>0x1 = FCLK_EN : Optional functional clock is enabled  |
| 17-16 | IDLEST                  | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved                | R    | 0h    |   |
| 1-0   | MODULEMODE              | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.33 CM\_PER\_GPIO3\_CLKCTRL Register (offset = B4h) [reset = 30000h]

CM\_PER\_GPIO3\_CLKCTRL is shown in [Figure 8-54](#) and described in [Table 8-60](#).

This register manages the GPIO3 clocks.

**Figure 8-54. CM\_PER\_GPIO3\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-60. CM\_PER\_GPIO3\_CLKCTRL Register Field Descriptions**

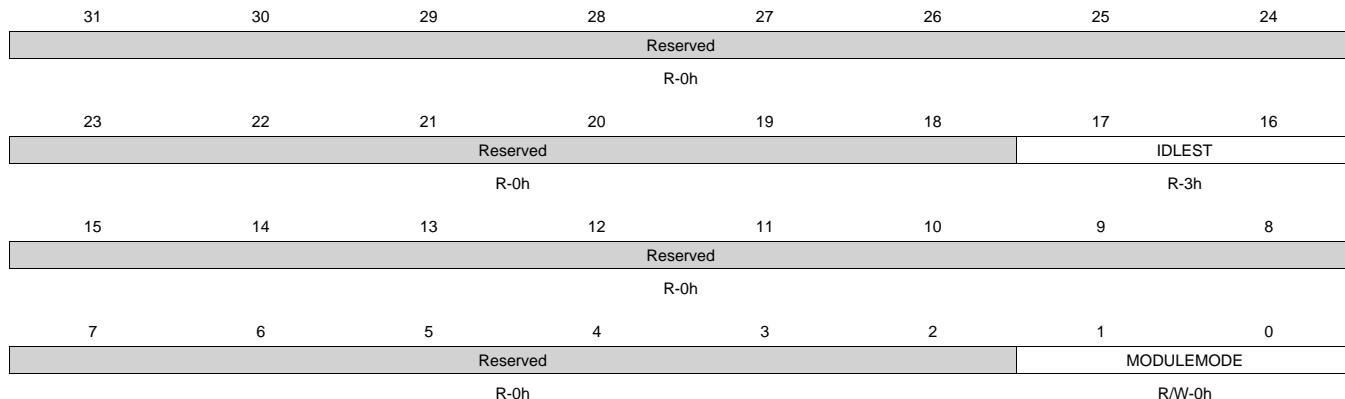
| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-19 | Reserved                | R    | 0h    |   |
| 18    | OPTFCLKEN_GPIO_3_GDBCLK | R/W  | 0h    | Optional functional clock control.<br>0x0 = FCLK_DIS : Optional functional clock is disabled<br>0x1 = FCLK_EN : Optional functional clock is enabled  |
| 17-16 | IDLEST                  | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved                | R    | 0h    |   |
| 1-0   | MODULEMODE              | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.34 CM\_PER\_TPCC\_CLKCTRL Register (offset = BCh) [reset = 30000h]

CM\_PER\_TPCC\_CLKCTRL is shown in [Figure 8-55](#) and described in [Table 8-61](#).

This register manages the TPCC clocks.

**Figure 8-55. CM\_PER\_TPCC\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-61. CM\_PER\_TPCC\_CLKCTRL Register Field Descriptions**

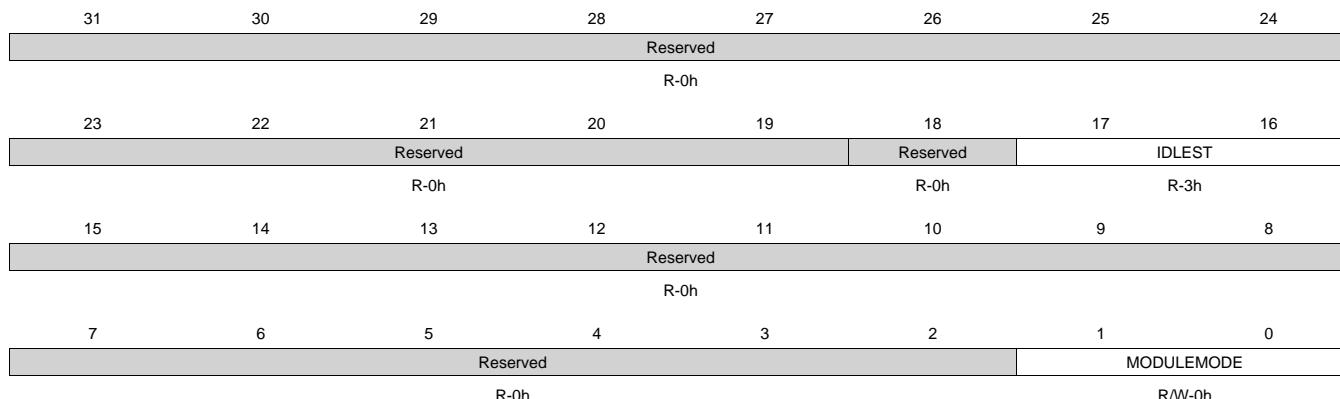
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.35 CM\_PER\_DCANO\_CLKCTRL Register (offset = C0h) [reset = 30000h]

CM\_PER\_DCANO\_CLKCTRL is shown in [Figure 8-56](#) and described in [Table 8-62](#).

This register manages the DCAN0 clocks.

**Figure 8-56. CM\_PER\_DCANO\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-62. CM\_PER\_DCANO\_CLKCTRL Register Field Descriptions**

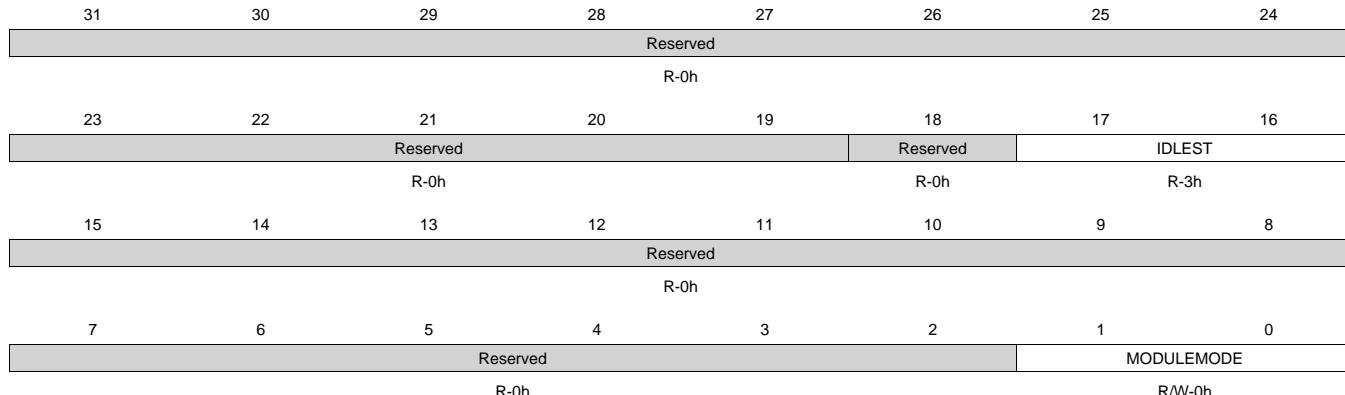
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.36 CM\_PER\_DCAN1\_CLKCTRL Register (offset = C4h) [reset = 30000h]

CM\_PER\_DCAN1\_CLKCTRL is shown in [Figure 8-57](#) and described in [Table 8-63](#).

This register manages the DCAN1 clocks.

**Figure 8-57. CM\_PER\_DCAN1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-63. CM\_PER\_DCAN1\_CLKCTRL Register Field Descriptions**

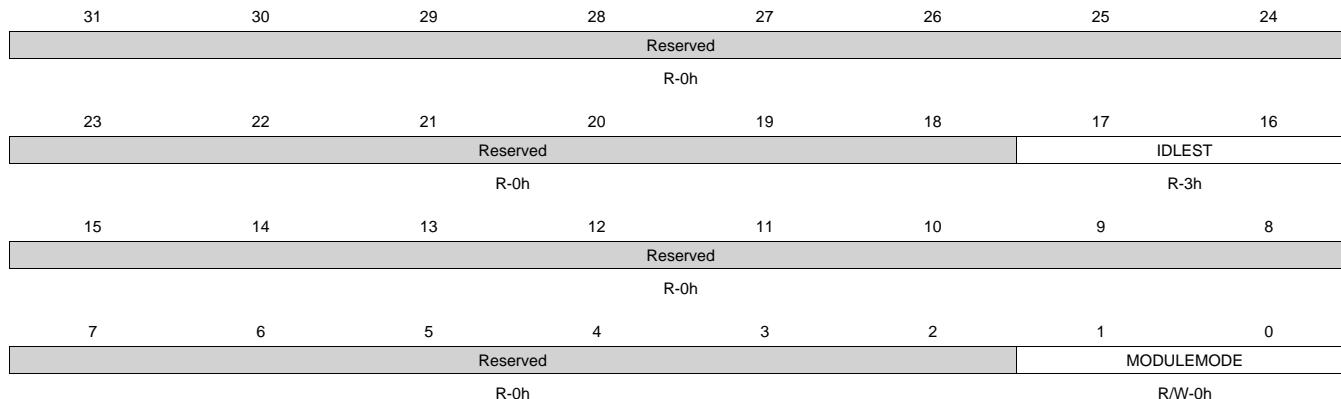
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.37 CM\_PER\_EPWMSS1\_CLKCTRL Register (offset = CCh) [reset = 30000h]

CM\_PER\_EPWMSS1\_CLKCTRL is shown in Figure 8-58 and described in Table 8-64.

This register manages the PWMSS1 clocks.

**Figure 8-58. CM\_PER\_EPWMSS1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-64. CM\_PER\_EPWMSS1\_CLKCTRL Register Field Descriptions**

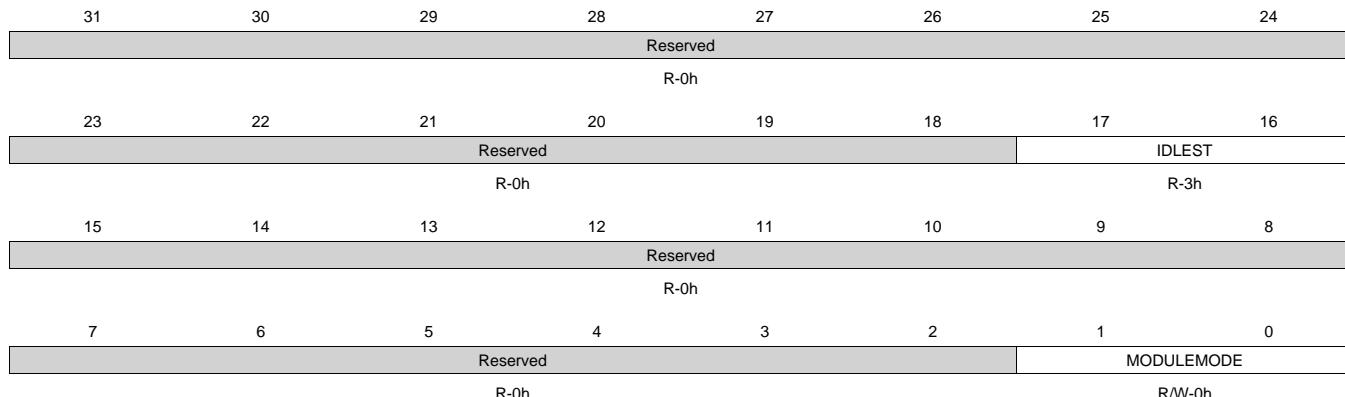
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.38 CM\_PER\_EMIF\_FW\_CLKCTRL Register (offset = D0h) [reset = 30000h]

CM\_PER\_EMIF\_FW\_CLKCTRL is shown in [Figure 8-59](#) and described in [Table 8-65](#).

This register manages the EMIF Firewall clocks.

**Figure 8-59. CM\_PER\_EMIF\_FW\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-65. CM\_PER\_EMIF\_FW\_CLKCTRL Register Field Descriptions**

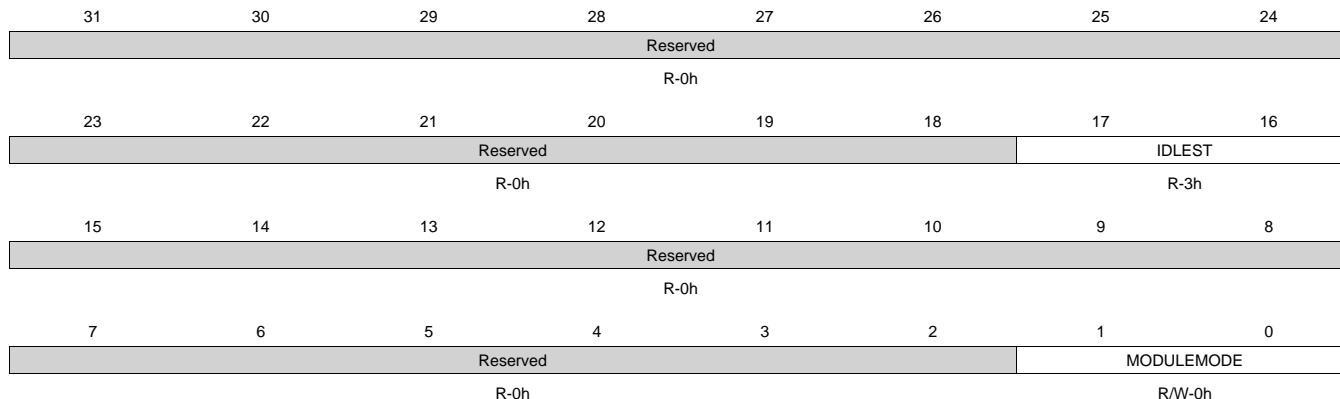
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.39 CM\_PER\_EPWMSS0\_CLKCTRL Register (offset = D4h) [reset = 30000h]

CM\_PER\_EPWMSS0\_CLKCTRL is shown in Figure 8-60 and described in Table 8-66.

This register manages the PWMSS0 clocks.

**Figure 8-60. CM\_PER\_EPWMSS0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-66. CM\_PER\_EPWMSS0\_CLKCTRL Register Field Descriptions**

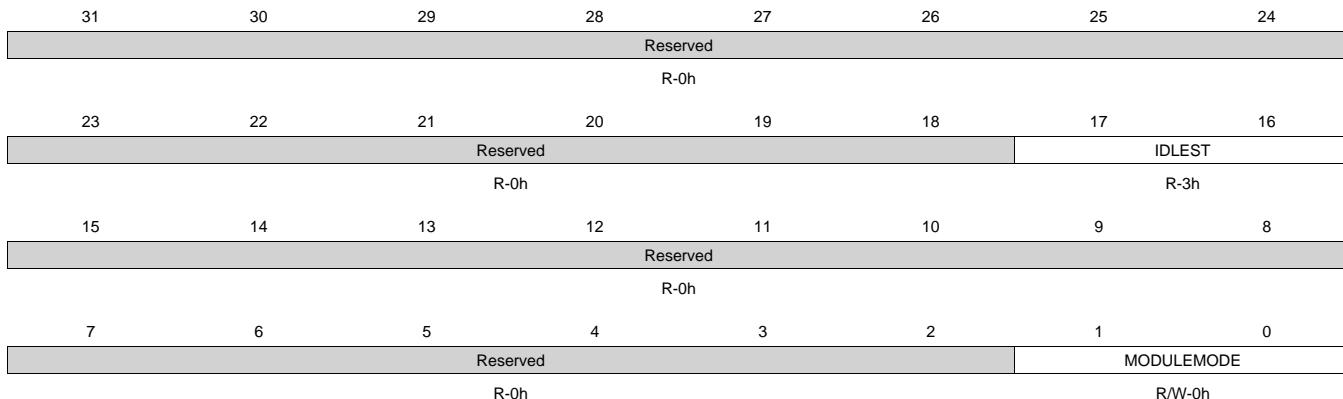
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### 8.1.12.1.40 CM\_PER\_EPWMSS2\_CLKCTRL Register (offset = D8h) [reset = 30000h]

CM\_PER\_EPWMSS2\_CLKCTRL is shown in [Figure 8-61](#) and described in [Table 8-67](#).

This register manages the PWMSS2 clocks.

**Figure 8-61. CM\_PER\_EPWMSS2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-67. CM\_PER\_EPWMSS2\_CLKCTRL Register Field Descriptions**

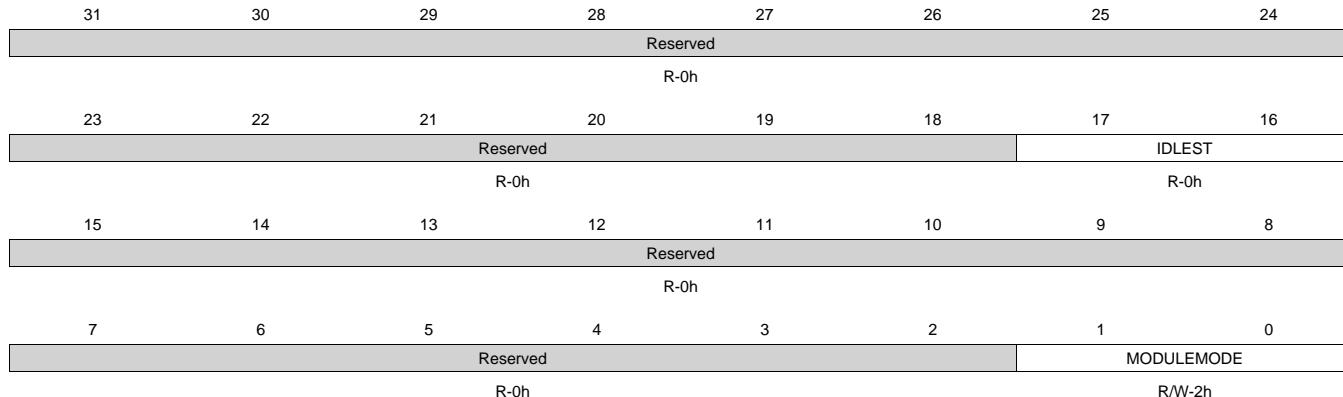
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

#### **8.1.12.1.41 CM\_PER\_L3\_INSTR\_CLKCTRL Register (offset = DCh) [reset = 2h]**

CM\_PER\_L3\_INSTR\_CLKCTRL is shown in [Figure 8-62](#) and described in [Table 8-68](#).

This register manages the L3 INSTR clocks.

**Figure 8-62. CM\_PER\_L3\_INSTR\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-68. CM\_PER\_L3\_INSTR\_CLKCTRL Register Field Descriptions**

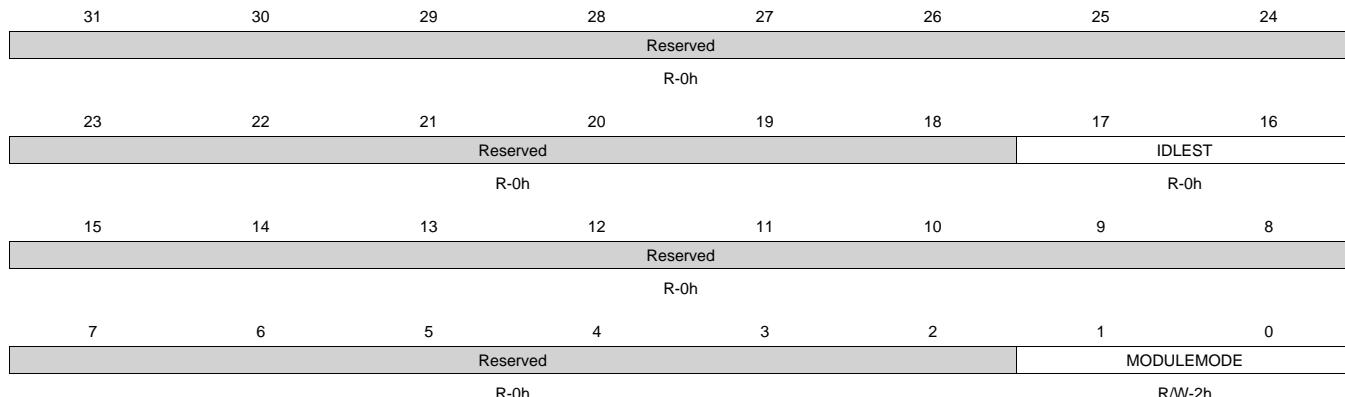
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### 8.1.12.1.42 CM\_PER\_L3\_CLKCTRL Register (offset = E0h) [reset = 2h]

CM\_PER\_L3\_CLKCTRL is shown in [Figure 8-63](#) and described in [Table 8-69](#).

This register manages the L3 Interconnect clocks.

**Figure 8-63. CM\_PER\_L3\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-69. CM\_PER\_L3\_CLKCTRL Register Field Descriptions**

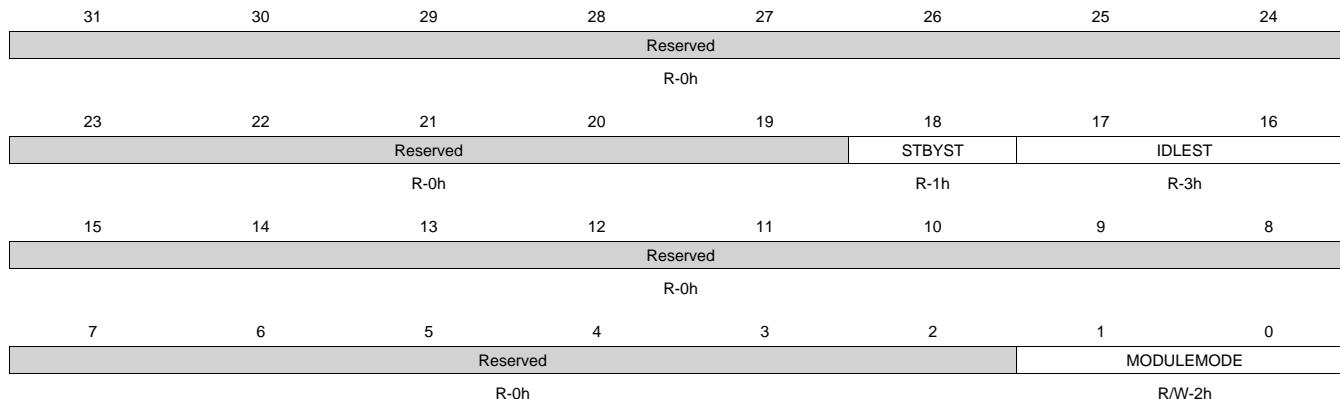
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### **8.1.12.1.43 CM\_PER\_IEEE5000\_CLKCTRL Register (offset = E4h) [reset = 70002h]**

CM\_PER\_IEEE5000\_CLKCTRL is shown in [Figure 8-64](#) and described in [Table 8-70](#).

This register manages the IEEE1500 clocks.

**Figure 8-64. CM\_PER\_IEEE5000\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-70. CM\_PER\_IEEE5000\_CLKCTRL Register Field Descriptions**

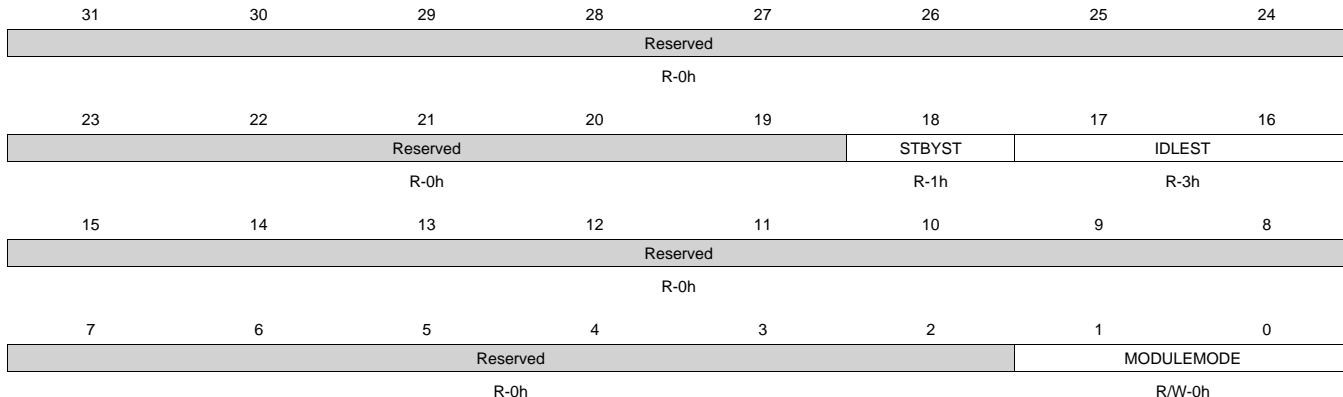
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### 8.1.12.1.44 CM\_PER\_PRU\_ICSS\_CLKCTRL Register (offset = E8h) [reset = 70000h]

CM\_PER\_PRU\_ICSS\_CLKCTRL is shown in [Figure 8-65](#) and described in [Table 8-71](#).

This register manages the PRU-ICSS clocks.

**Figure 8-65. CM\_PER\_PRU\_ICSS\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-71. CM\_PER\_PRU\_ICSS\_CLKCTRL Register Field Descriptions**

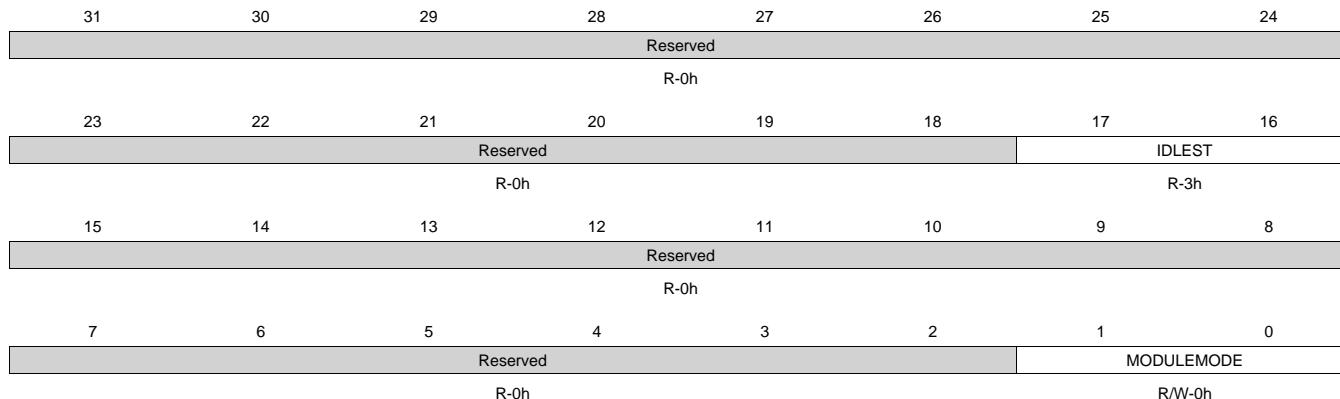
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.45 CM\_PER\_TIMER5\_CLKCTRL Register (offset = ECh) [reset = 30000h]

CM\_PER\_TIMER5\_CLKCTRL is shown in [Figure 8-66](#) and described in [Table 8-72](#).

This register manages the TIMER5 clocks.

**Figure 8-66. CM\_PER\_TIMER5\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-72. CM\_PER\_TIMER5\_CLKCTRL Register Field Descriptions**

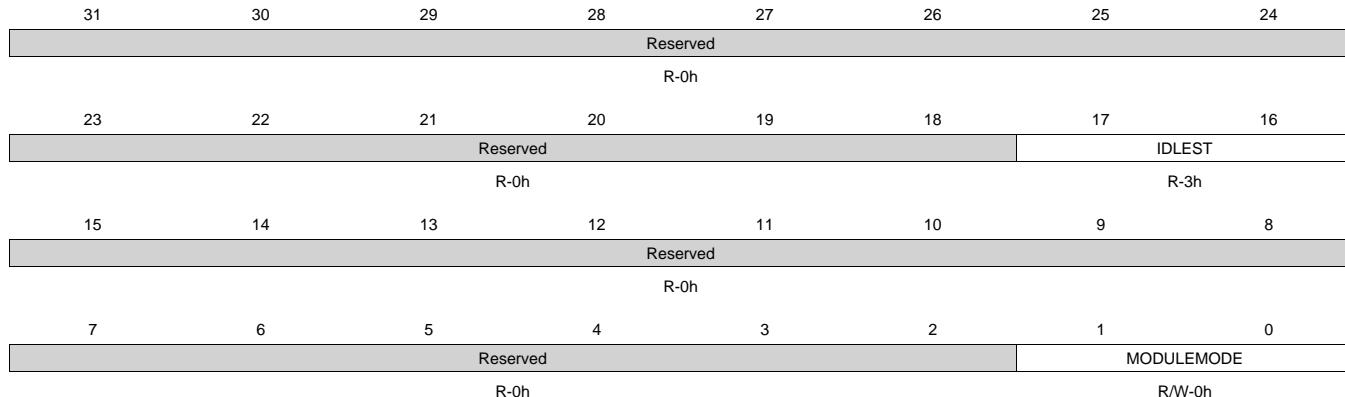
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### 8.1.12.1.46 CM\_PER\_TIMER6\_CLKCTRL Register (offset = F0h) [reset = 30000h]

CM\_PER\_TIMER6\_CLKCTRL is shown in [Figure 8-67](#) and described in [Table 8-73](#).

This register manages the TIMER6 clocks.

**Figure 8-67. CM\_PER\_TIMER6\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-73. CM\_PER\_TIMER6\_CLKCTRL Register Field Descriptions**

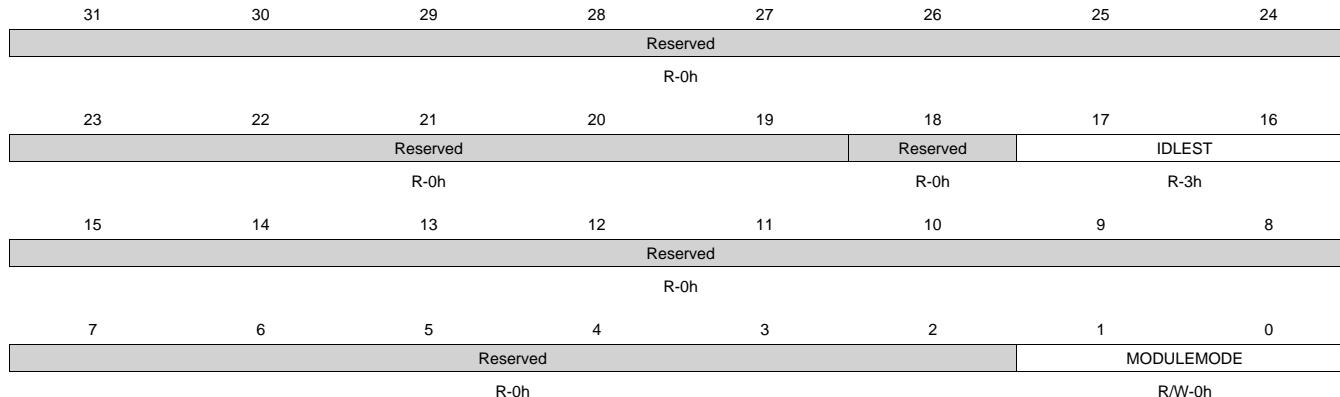
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.47 CM\_PER\_MMC1\_CLKCTRL Register (offset = F4h) [reset = 30000h]

CM\_PER\_MMC1\_CLKCTRL is shown in [Figure 8-68](#) and described in [Table 8-74](#).

This register manages the MMC1 clocks.

**Figure 8-68. CM\_PER\_MMC1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-74. CM\_PER\_MMC1\_CLKCTRL Register Field Descriptions**

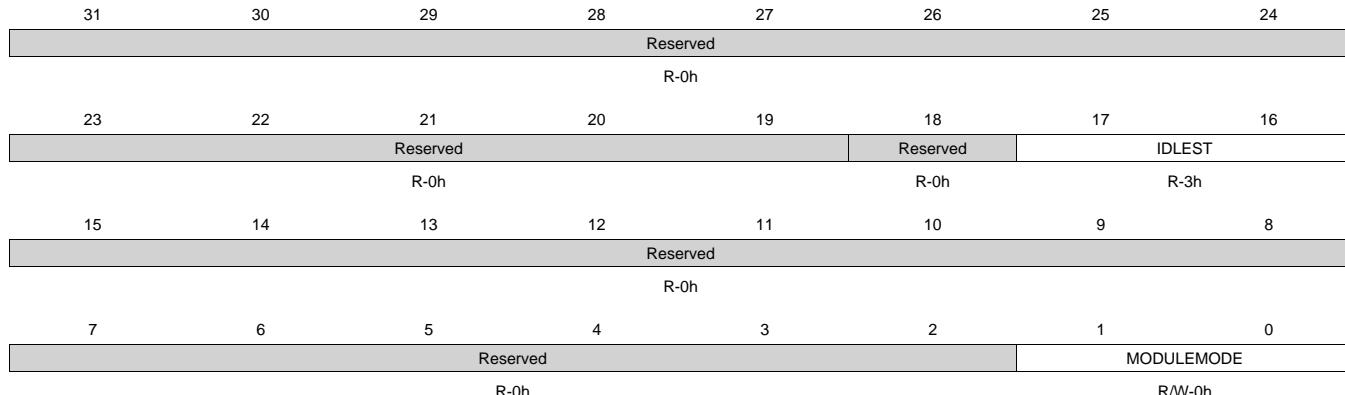
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.48 CM\_PER\_MMC2\_CLKCTRL Register (offset = F8h) [reset = 30000h]

CM\_PER\_MMC2\_CLKCTRL is shown in [Figure 8-69](#) and described in [Table 8-75](#).

This register manages the MMC2 clocks.

**Figure 8-69. CM\_PER\_MMC2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-75. CM\_PER\_MMC2\_CLKCTRL Register Field Descriptions**

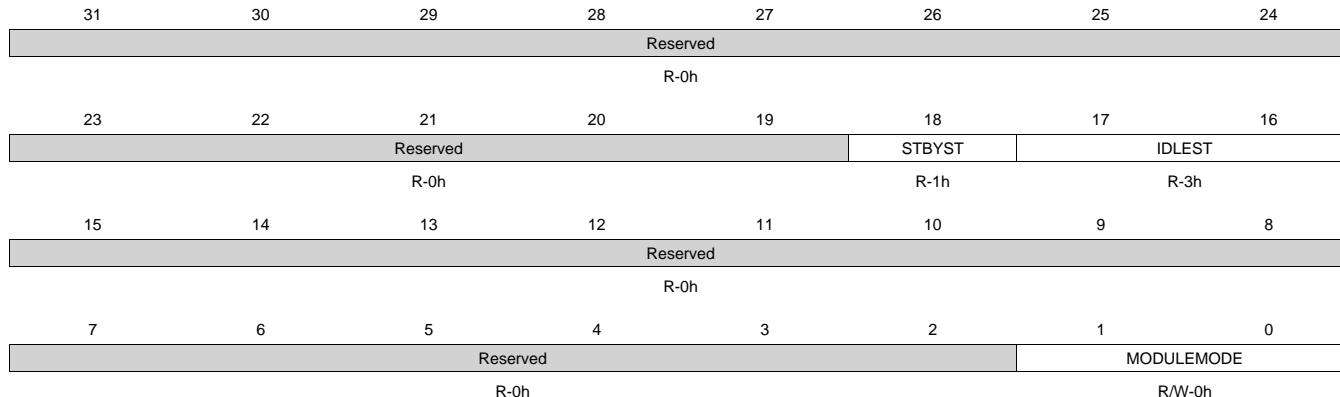
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.49 CM\_PER\_TPTC1\_CLKCTRL Register (offset = FCh) [reset = 70000h]

CM\_PER\_TPTC1\_CLKCTRL is shown in [Figure 8-70](#) and described in [Table 8-76](#).

This register manages the TPTC1 clocks.

**Figure 8-70. CM\_PER\_TPTC1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-76. CM\_PER\_TPTC1\_CLKCTRL Register Field Descriptions**

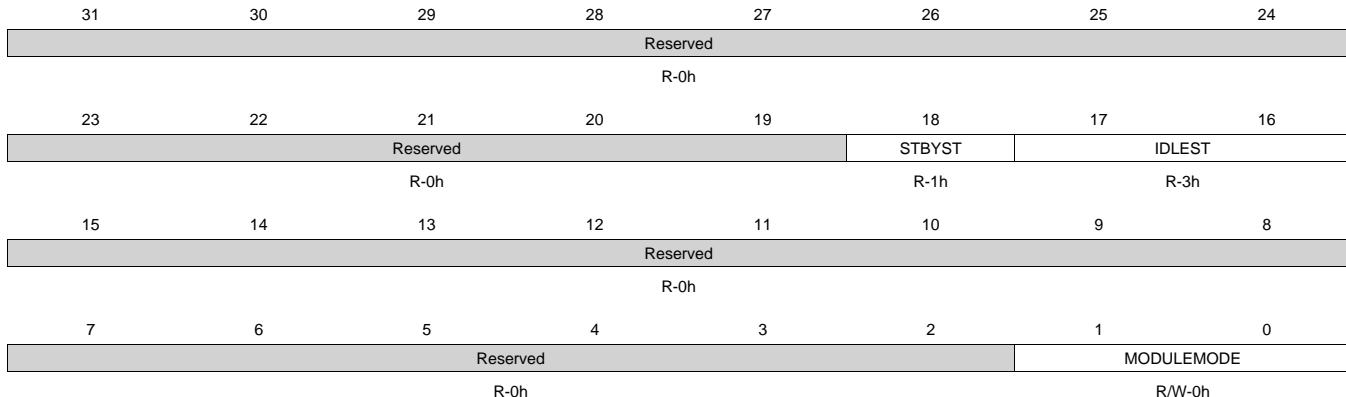
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.50 CM\_PER\_TPTC2\_CLKCTRL Register (offset = 100h) [reset = 70000h]

CM\_PER\_TPTC2\_CLKCTRL is shown in [Figure 8-71](#) and described in [Table 8-77](#).

This register manages the TPTC2 clocks.

**Figure 8-71. CM\_PER\_TPTC2\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-77. CM\_PER\_TPTC2\_CLKCTRL Register Field Descriptions**

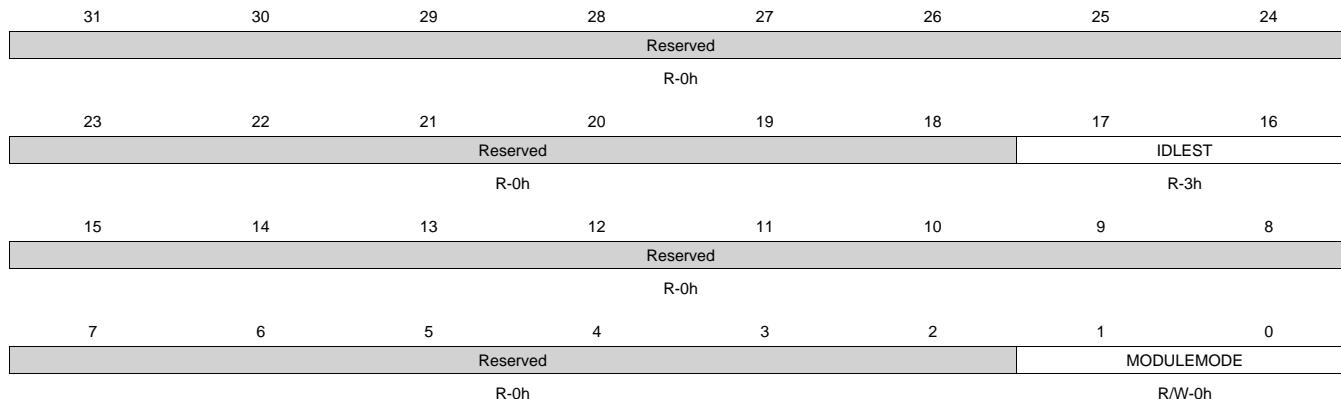
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.51 CM\_PER\_SPINLOCK\_CLKCTRL Register (offset = 10Ch) [reset = 30000h]

CM\_PER\_SPINLOCK\_CLKCTRL is shown in [Figure 8-72](#) and described in [Table 8-78](#).

This register manages the SPINLOCK clocks.

**Figure 8-72. CM\_PER\_SPINLOCK\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-78. CM\_PER\_SPINLOCK\_CLKCTRL Register Field Descriptions**

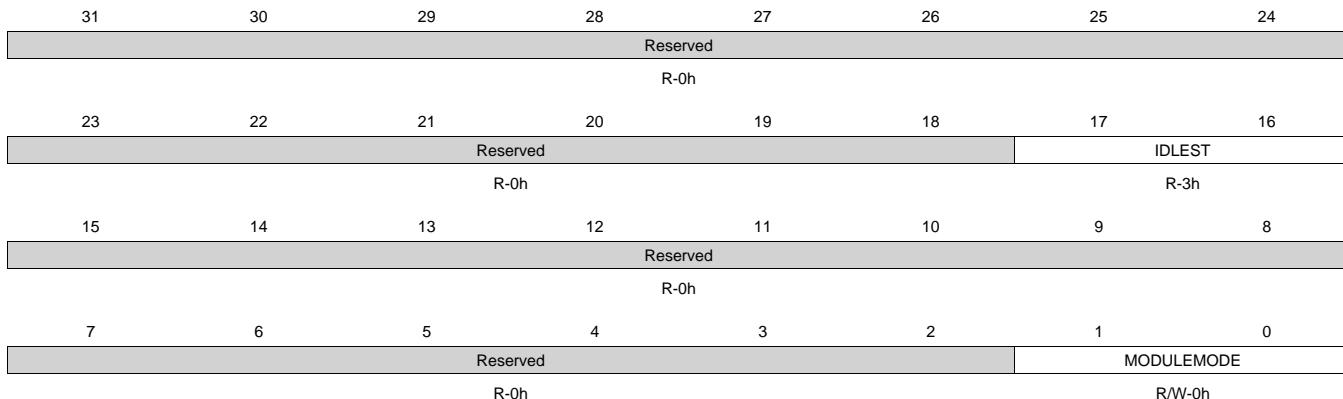
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.52 CM\_PER\_MAILBOX0\_CLKCTRL Register (offset = 110h) [reset = 30000h]

CM\_PER\_MAILBOX0\_CLKCTRL is shown in Figure 8-73 and described in Table 8-79.

This register manages the MAILBOX0 clocks.

**Figure 8-73. CM\_PER\_MAILBOX0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-79. CM\_PER\_MAILBOX0\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.53 CM\_PER\_L4HS\_CLKSTCTRL Register (offset = 11Ch) [reset = 7Ah]

CM\_PER\_L4HS\_CLKSTCTRL is shown in [Figure 8-74](#) and described in [Table 8-80](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-74. CM\_PER\_L4HS\_CLKSTCTRL Register**

|          |                            |                             |                              |                       |          |           |    |
|----------|----------------------------|-----------------------------|------------------------------|-----------------------|----------|-----------|----|
| 31       | 30                         | 29                          | 28                           | 27                    | 26       | 25        | 24 |
| Reserved |                            |                             |                              |                       |          |           |    |
| R-0h     |                            |                             |                              |                       |          |           |    |
| 23       | 22                         | 21                          | 20                           | 19                    | 18       | 17        | 16 |
| Reserved |                            |                             |                              |                       |          |           |    |
| R-0h     |                            |                             |                              |                       |          |           |    |
| 15       | 14                         | 13                          | 12                           | 11                    | 10       | 9         | 8  |
| Reserved |                            |                             |                              |                       |          |           |    |
| R-0h     |                            |                             |                              |                       |          |           |    |
| 7        | 6                          | 5                           | 4                            | 3                     | 2        | 1         | 0  |
| Reserved | CLKACTIVITY_CPSW_5MHZ_GCLK | CLKACTIVITY_CPSW_50MHZ_GCLK | CLKACTIVITY_CPSW_250MHZ_GCLK | CLKACTIVITY_L4HS_GCLK | Reserved | CLKTRCTRL |    |
| R-0h     | R-1h                       | R-1h                        | R-1h                         | R-1h                  | R-0h     | R/W-2h    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-80. CM\_PER\_L4HS\_CLKSTCTRL Register Field Descriptions**

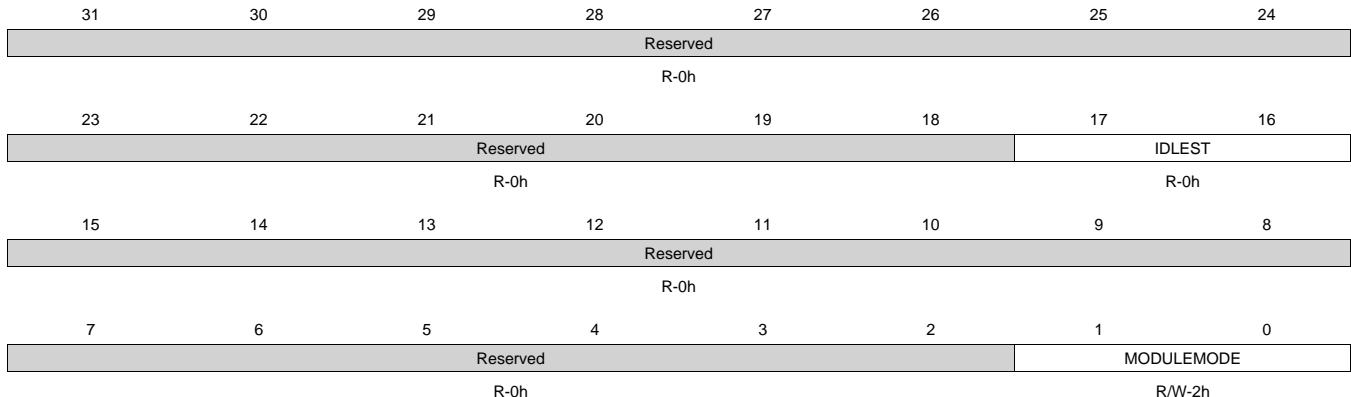
| Bit  | Field                        | Type | Reset | Description   |
|------|------------------------------|------|-------|---|
| 31-7 | Reserved                     | R    | 0h    |   |
| 6    | CLKACTIVITY_CPSW_5MHZ_GCLK   | R    | 1h    | This field indicates the state of the CPSW_5MHZ_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 5    | CLKACTIVITY_CPSW_50MHZ_GCLK  | R    | 1h    | This field indicates the state of the CPSW_50MHZ_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 4    | CLKACTIVITY_CPSW_250MHZ_GCLK | R    | 1h    | This field indicates the state of the CPSW_250MHZ_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 3    | CLKACTIVITY_L4HS_GCLK        | R    | 1h    | This field indicates the state of the L4HS_GCLK clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 2    | Reserved                     | R    | 0h    |   |
| 1-0  | CLKTRCTRL                    | R/W  | 2h    | Controls the clock state transition of the L4 Fast clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.54 CM\_PER\_L4HS\_CLKCTRL Register (offset = 120h) [reset = 2h]

CM\_PER\_L4HS\_CLKCTRL is shown in [Figure 8-75](#) and described in [Table 8-81](#).

This register manages the L4 Fast clocks.

**Figure 8-75. CM\_PER\_L4HS\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-81. CM\_PER\_L4HS\_CLKCTRL Register Field Descriptions**

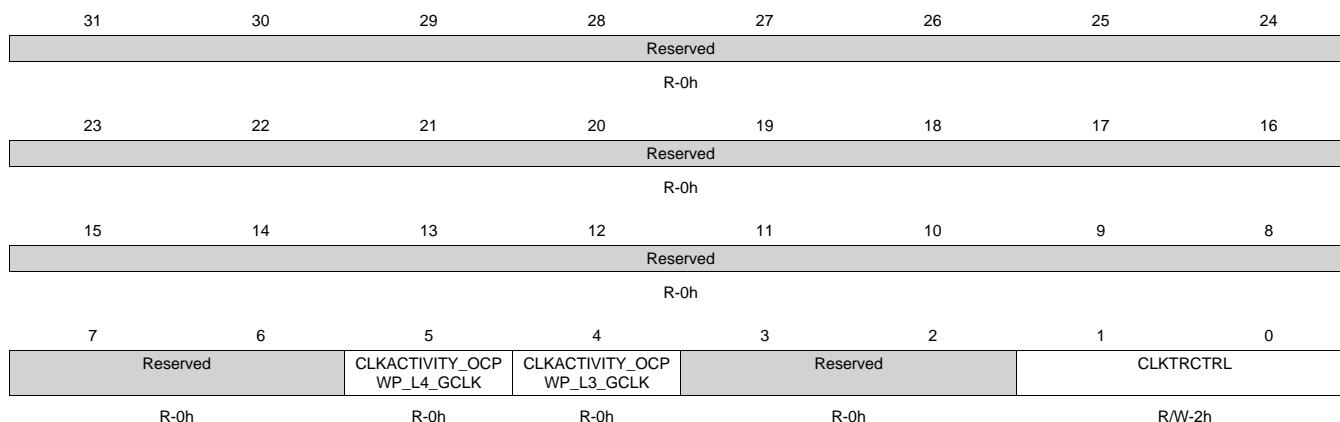
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 0h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 2h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.55 CM\_PER\_OCPWP\_L3\_CLKSTCTRL Register (offset = 12Ch) [reset = 2h]

CM\_PER\_OCPWP\_L3\_CLKSTCTRL is shown in [Figure 8-76](#) and described in [Table 8-82](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-76. CM\_PER\_OCPWP\_L3\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-82. CM\_PER\_OCPWP\_L3\_CLKSTCTRL Register Field Descriptions**

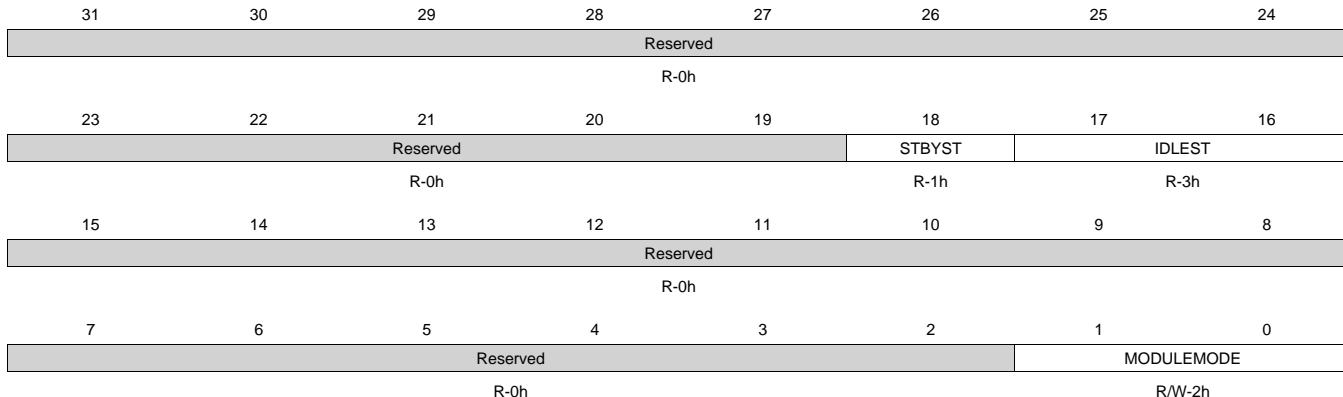
| Bit  | Field                     | Type | Reset | Description   |
|------|---------------------------|------|-------|---|
| 31-6 | Reserved                  | R    | 0h    |   |
| 5    | CLKACTIVITY_OCPWP_L4_GCLK | R    | 0h    | This field indicates the state of the OCPWP L4 clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 4    | CLKACTIVITY_OCPWP_L3_GCLK | R    | 0h    | This field indicates the state of the OCPWP L3 clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 3-2  | Reserved                  | R    | 0h    |   |
| 1-0  | CLKTRCTRL                 | R/W  | 2h    | Controls the clock state transition of the OCPWP clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.56 CM\_PER\_OCPWP\_CLKCTRL Register (offset = 130h) [reset = 70002h]

CM\_PER\_OCPWP\_CLKCTRL is shown in [Figure 8-77](#) and described in [Table 8-83](#).

This register manages the OCPWP clocks.

**Figure 8-77. CM\_PER\_OCPWP\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-83. CM\_PER\_OCPWP\_CLKCTRL Register Field Descriptions**

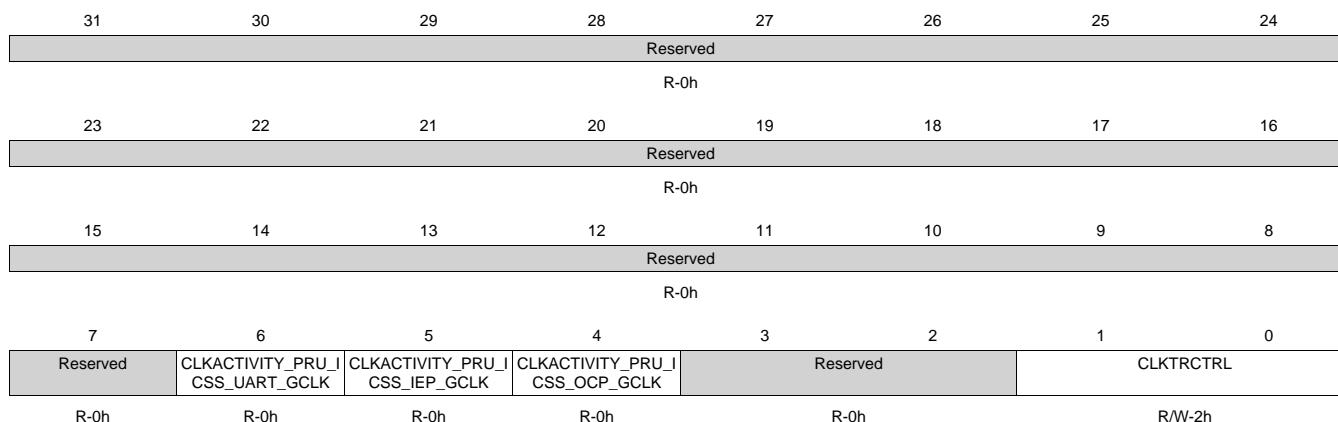
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.1.57 CM\_PER\_PRU\_ICSS\_CLKSTCTRL Register (offset = 140h) [reset = 2h]

CM\_PER\_PRU\_ICSS\_CLKSTCTRL is shown in [Figure 8-78](#) and described in [Table 8-84](#).

This register enables the clock domain state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-78. CM\_PER\_PRU\_ICSS\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-84. CM\_PER\_PRU\_ICSS\_CLKSTCTRL Register Field Descriptions**

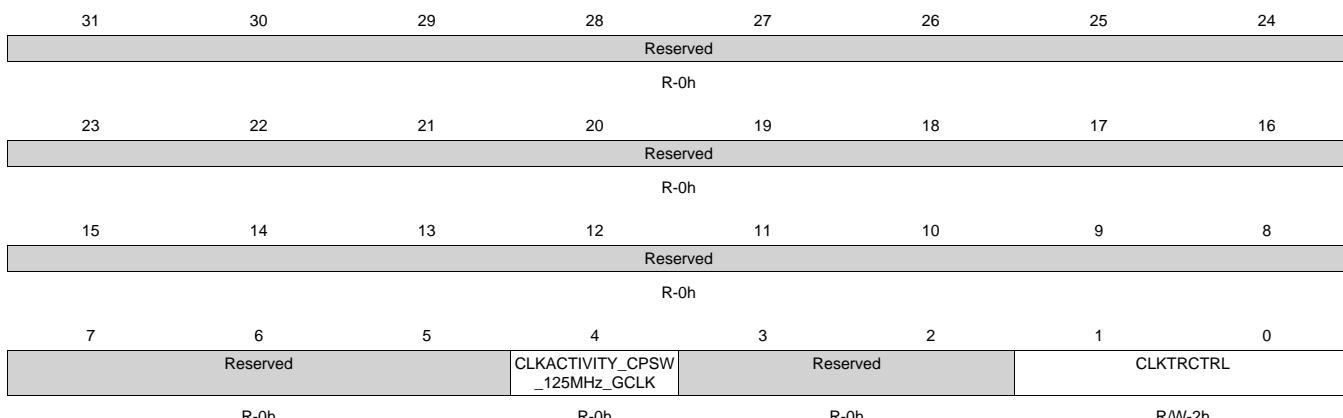
| Bit  | Field                          | Type | Reset | Description  |
|------|--------------------------------|------|-------|--|
| 31-7 | Reserved                       | R    | 0h    |  |
| 6    | CLKACTIVITY_PRU_ICSS_UART_GCLK | R    | 0h    | This field indicates the state of the PRU-ICSS UART clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 5    | CLKACTIVITY_PRU_ICSS_IEP_GCLK  | R    | 0h    | This field indicates the state of the PRU-ICSS IEP clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 4    | CLKACTIVITY_PRU_ICSS_OCP_GCLK  | R    | 0h    | This field indicates the state of the PRU-ICSS OCP clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 3-2  | Reserved                       | R    | 0h    |  |
| 1-0  | CLKTRCTRL                      | R/W  | 2h    | Controls the clock state transition of the PRU-ICSS OCP clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.58 CM\_PER\_CPSW\_CLKSTCTRL Register (offset = 144h) [reset = 2h]

CM\_PER\_CPSW\_CLKSTCTRL is shown in [Figure 8-79](#) and described in [Table 8-85](#).

This register enables the clock domain state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-79. CM\_PER\_CPSW\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-85. CM\_PER\_CPSW\_CLKSTCTRL Register Field Descriptions**

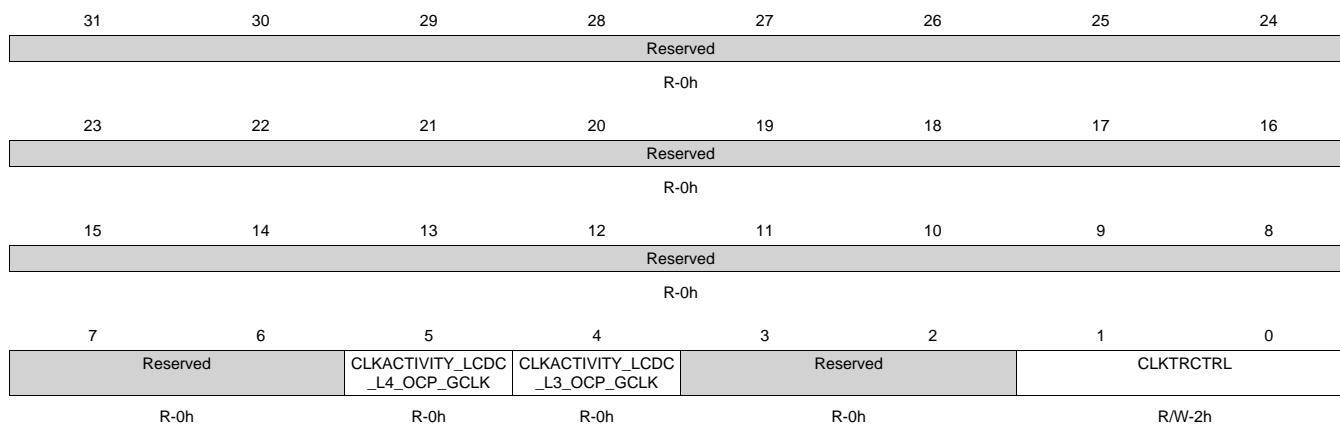
| Bit  | Field                        | Type | Reset | Description  |
|------|------------------------------|------|-------|--|
| 31-5 | Reserved                     | R    | 0h    |  |
| 4    | CLKACTIVITY_CPSW_125MHz_GCLK | R    | 0h    | This field indicates the state of the CPSW 125 MHz OCP clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 3-2  | Reserved                     | R    | 0h    |  |
| 1-0  | CLKTRCTRL                    | R/W  | 2h    | Controls the clock state transition of the CPSW OCP clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.59 CM\_PER\_LCDC\_CLKSTCTRL Register (offset = 148h) [reset = 2h]

CM\_PER\_LCDC\_CLKSTCTRL is shown in [Figure 8-80](#) and described in [Table 8-86](#).

This register enables the clock domain state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-80. CM\_PER\_LCDC\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-86. CM\_PER\_LCDC\_CLKSTCTRL Register Field Descriptions**

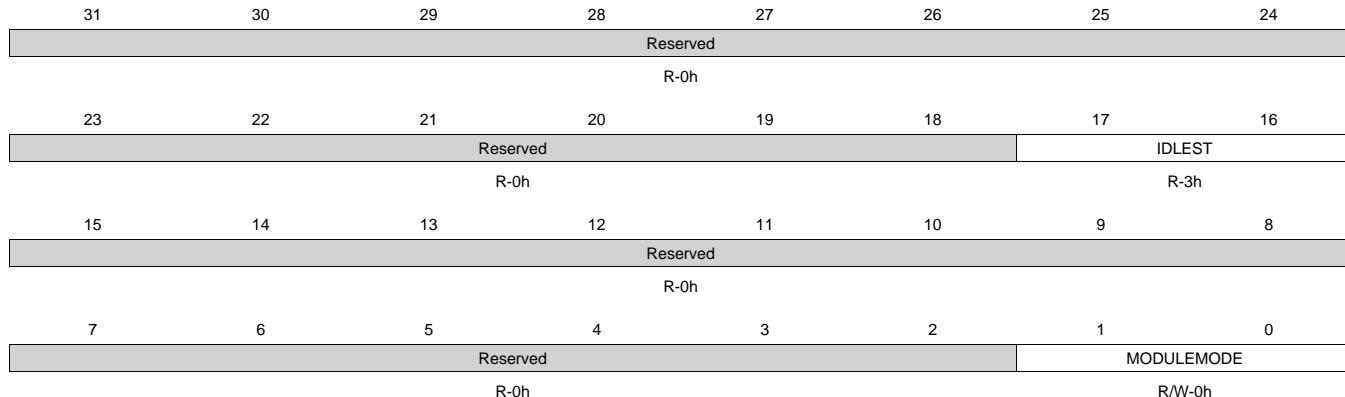
| Bit  | Field                        | Type | Reset | Description  |
|------|------------------------------|------|-------|--|
| 31-6 | Reserved                     | R    | 0h    |  |
| 5    | CLKACTIVITY_LCDC_L4_OCP_GCLK | R    | 0h    | This field indicates the state of the LCDC L4 OCP clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 4    | CLKACTIVITY_LCDC_L3_OCP_GCLK | R    | 0h    | This field indicates the state of the LCDC L3 OCP clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 3-2  | Reserved                     | R    | 0h    |  |
| 1-0  | CLKTRCTRL                    | R/W  | 2h    | Controls the clock state transition of the LCDC OCP clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.1.60 CM\_PER\_CLKDIV32K\_CLKCTRL Register (offset = 14Ch) [reset = 30000h]

CM\_PER\_CLKDIV32K\_CLKCTRL is shown in [Figure 8-81](#) and described in [Table 8-87](#).

This register manages the CLKDIV32K clocks.

**Figure 8-81. CM\_PER\_CLKDIV32K\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-87. CM\_PER\_CLKDIV32K\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.1.61 CM\_PER\_CLK\_24MHZ\_CLKSTCTRL Register (offset = 150h) [reset = 2h]

CM\_PER\_CLK\_24MHZ\_CLKSTCTRL is shown in [Figure 8-82](#) and described in [Table 8-88](#).

This register enables the clock domain state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-82. CM\_PER\_CLK\_24MHZ\_CLKSTCTRL Register**

|          |    |                            |    |          |    |           |    |
|----------|----|----------------------------|----|----------|----|-----------|----|
| 31       | 30 | 29                         | 28 | 27       | 26 | 25        | 24 |
| Reserved |    |                            |    |          |    |           |    |
| R-0h     |    |                            |    |          |    |           |    |
| 23       | 22 | 21                         | 20 | 19       | 18 | 17        | 16 |
| Reserved |    |                            |    |          |    |           |    |
| R-0h     |    |                            |    |          |    |           |    |
| 15       | 14 | 13                         | 12 | 11       | 10 | 9         | 8  |
| Reserved |    |                            |    |          |    |           |    |
| R-0h     |    |                            |    |          |    |           |    |
| 7        | 6  | 5                          | 4  | 3        | 2  | 1         | 0  |
| Reserved |    | CLKACTIVITY_CLK_24MHZ_GCLK |    | Reserved |    | CLKTRCTRL |    |
| R-0h     |    | R-0h                       |    | R-0h     |    | R/W-2h    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-88. CM\_PER\_CLK\_24MHZ\_CLKSTCTRL Register Field Descriptions**

| Bit  | Field                      | Type | Reset | Description   |
|------|----------------------------|------|-------|---|
| 31-5 | Reserved                   | R    | 0h    |   |
| 4    | CLKACTIVITY_CLK_24MHZ_GCLK | R    | 0h    | This field indicates the state of the 24MHz clock in the domain.<br>0x0 = Inact<br>0x1 = Act  |
| 3-2  | Reserved                   | R    | 0h    |   |
| 1-0  | CLKTRCTRL                  | R/W  | 2h    | Controls the clock state transition of the 24MHz clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.2 CM\_WKUP Registers

[Table 8-89](#) lists the memory-mapped registers for the CM\_WKUP. All register offset addresses not listed in [Table 8-89](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-89. CM\_WKUP REGISTERS**

| Offset | Acronym                 | Register Name   | Section                            |
|--------|-------------------------|---|------------------------------------|
| 0h     | CM_WKUP_CLKSTCTRL       | This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.2.1</a> |
| 4h     | CM_WKUP_CONTROL_CLKCTRL | This register manages the Control Module clocks.  | <a href="#">Section 8.1.12.2.2</a> |
| 8h     | CM_WKUP_GPIO0_CLKCTRL   | This register manages the GPIO0 clocks.   | <a href="#">Section 8.1.12.2.3</a> |

**Table 8-89. CM\_WKUP REGISTERS (continued)**

| Offset | Acronym                    | Register Name   | Section                             |
|--------|----------------------------|---|-------------------------------------|
| Ch     | CM_WKUP_L4WKUP_CLKCTRL     | This register manages the L4WKUP clocks.  | <a href="#">Section 8.1.12.2.4</a>  |
| 10h    | CM_WKUP_TIMER0_CLKCTRL     | This register manages the TIMER0 clocks.  | <a href="#">Section 8.1.12.2.5</a>  |
| 14h    | CM_WKUP_DEBUGSS_CLKCTRL    | This register manages the DEBUGSS clocks.   | <a href="#">Section 8.1.12.2.6</a>  |
| 18h    | CM_L3_AON_CLKSTCTRL        | This register enables the domain power state transition.<br>It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain.           | <a href="#">Section 8.1.12.2.7</a>  |
| 1Ch    | CM_AUTOIDLE_DPLL_MPU       | This register provides automatic control over the DPLL activity.  | <a href="#">Section 8.1.12.2.8</a>  |
| 20h    | CM_IDLEST_DPLL_MPU         | This register allows monitoring the master clock activity.<br>This register is read only and automatically updated.[warm reset insensitive]   | <a href="#">Section 8.1.12.2.9</a>  |
| 24h    | CM_SSC_DELTAMSTEP_DPLL_MPU | Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part.<br>[warm reset insensitive]   | <a href="#">Section 8.1.12.2.10</a> |
| 28h    | CM_SSC_MODFREQDIV_DPLL_MPU | Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL_REFCLK/4 Fm = [DPLL_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV_MANTISSA * 2^MODFREQDIV_EXPONENT [warm reset insensitive] | <a href="#">Section 8.1.12.2.11</a> |
| 2Ch    | CM_CLKSEL_DPLL_MPU         | This register provides controls over the DPLL.  | <a href="#">Section 8.1.12.2.12</a> |
| 30h    | CM_AUTOIDLE_DPLL_DDR       | This register provides automatic control over the DPLL activity.  | <a href="#">Section 8.1.12.2.13</a> |
| 34h    | CM_IDLEST_DPLL_DDR         | This register allows monitoring the master clock activity.<br>This register is read only and automatically updated.<br>[warm reset insensitive]   | <a href="#">Section 8.1.12.2.14</a> |
| 38h    | CM_SSC_DELTAMSTEP_DPLL_DR  | Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part.<br>[warm reset insensitive]   | <a href="#">Section 8.1.12.2.15</a> |
| 3Ch    | CM_SSC_MODFREQDIV_DPLL_DR  | Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL_REFCLK/4 Fm = [DPLL_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV_MANTISSA * 2^MODFREQDIV_EXPONENT [warm reset insensitive] | <a href="#">Section 8.1.12.2.16</a> |
| 40h    | CM_CLKSEL_DPLL_DDR         | This register provides controls over the DPLL.  | <a href="#">Section 8.1.12.2.17</a> |
| 44h    | CM_AUTOIDLE_DPLL_DISP      | This register provides automatic control over the DPLL activity.  | <a href="#">Section 8.1.12.2.18</a> |
| 48h    | CM_IDLEST_DPLL_DISP        | This register allows monitoring the master clock activity.<br>This register is read only and automatically updated.<br>[warm reset insensitive]   | <a href="#">Section 8.1.12.2.19</a> |
| 4Ch    | CM_SSC_DELTAMSTEP_DPLL_ISP | Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part.<br>[warm reset insensitive]   | <a href="#">Section 8.1.12.2.20</a> |
| 50h    | CM_SSC_MODFREQDIV_DPLL_ISP | Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL_REFCLK/4 Fm = [DPLL_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV_MANTISSA * 2^MODFREQDIV_EXPONENT [warm reset insensitive] | <a href="#">Section 8.1.12.2.21</a> |
| 54h    | CM_CLKSEL_DPLL_DISP        | This register provides controls over the DPLL.  | <a href="#">Section 8.1.12.2.22</a> |
| 58h    | CM_AUTOIDLE_DPLL_CORE      | This register provides automatic control over the DPLL activity.  | <a href="#">Section 8.1.12.2.23</a> |

**Table 8-89. CM\_WKUP REGISTERS (continued)**

| Offset | Acronym                      | Register Name   | Section                             |
|--------|------------------------------|---|-------------------------------------|
| 5Ch    | CM_IDLEST_DPLL_CORE          | This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]   | <a href="#">Section 8.1.12.2.24</a> |
| 60h    | CM_SSC_DELTAMSTEP_DPLL_CORE  | Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]  | <a href="#">Section 8.1.12.2.25</a> |
| 64h    | CM_SSC_MODFREQDIV_DPLL_CORE  | Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DLL_REFCLK/4 Fm = [DLL_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV_MANTISSA * 2^MODFREQDIV_EXPONENT [warm reset insensitive] | <a href="#">Section 8.1.12.2.26</a> |
| 68h    | CM_CLKSEL_DPLL_CORE          | This register provides controls over the DLL.   | <a href="#">Section 8.1.12.2.27</a> |
| 6Ch    | CM_AUTOIDLE_DPLL_PER         | This register provides automatic control over the DLL activity.   | <a href="#">Section 8.1.12.2.28</a> |
| 70h    | CM_IDLEST_DPLL_PER           | This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]   | <a href="#">Section 8.1.12.2.29</a> |
| 74h    | CM_SSC_DELTAMSTEP_DPLL_PER   | Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]  | <a href="#">Section 8.1.12.2.30</a> |
| 78h    | CM_SSC_MODFREQDIV_DPLL_PER   | Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DLL_REFCLK/4 Fm = [DLL_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV_MANTISSA * 2^MODFREQDIV_EXPONENT [warm reset insensitive] | <a href="#">Section 8.1.12.2.31</a> |
| 7Ch    | CM_CLKDCOLDO_DPLL_PER        | This register provides controls over the digitally controlled oscillator output of the PER DLL.   | <a href="#">Section 8.1.12.2.32</a> |
| 80h    | CM_DIV_M4_DPLL_CORE          | This register provides controls over the CLKOUT1 o/p of the HSDIVIDER.  | <a href="#">Section 8.1.12.2.33</a> |
| 84h    | CM_DIV_M5_DPLL_CORE          | This register provides controls over the CLKOUT2 o/p of the HSDIVIDER.  | <a href="#">Section 8.1.12.2.34</a> |
| 88h    | CM_CLKMODE_DPLL_MPU          | This register allows controlling the DLL modes.   | <a href="#">Section 8.1.12.2.35</a> |
| 8Ch    | CM_CLKMODE_DPLL_PER          | This register allows controlling the DLL modes.   | <a href="#">Section 8.1.12.2.36</a> |
| 90h    | CM_CLKMODE_DPLL_CORE         | This register allows controlling the DLL modes.   | <a href="#">Section 8.1.12.2.37</a> |
| 94h    | CM_CLKMODE_DPLL_DDR          | This register allows controlling the DLL modes.   | <a href="#">Section 8.1.12.2.38</a> |
| 98h    | CM_CLKMODE_DPLL_DISP         | This register allows controlling the DLL modes.   | <a href="#">Section 8.1.12.2.39</a> |
| 9Ch    | CM_CLKSEL_DPLL_PERIPH        | This register provides controls over the DLL.   | <a href="#">Section 8.1.12.2.40</a> |
| A0h    | CM_DIV_M2_DPLL_DDR           | This register provides controls over the M2 divider of the DLL.   | <a href="#">Section 8.1.12.2.41</a> |
| A4h    | CM_DIV_M2_DPLL_DISP          | This register provides controls over the M2 divider of the DLL.   | <a href="#">Section 8.1.12.2.42</a> |
| A8h    | CM_DIV_M2_DPLL_MPU           | This register provides controls over the M2 divider of the DLL.   | <a href="#">Section 8.1.12.2.43</a> |
| ACh    | CM_DIV_M2_DPLL_PER           | This register provides controls over the M2 divider of the DLL.   | <a href="#">Section 8.1.12.2.44</a> |
| B0h    | CM_WKUP_WKUP_M3_CLKCTRL      | This register manages the WKUP M3 clocks.   | <a href="#">Section 8.1.12.2.45</a> |
| B4h    | CM_WKUP_UART0_CLKCTRL        | This register manages the UART0 clocks.   | <a href="#">Section 8.1.12.2.46</a> |
| B8h    | CM_WKUP_I2C0_CLKCTRL         | This register manages the I2C0 clocks.  | <a href="#">Section 8.1.12.2.47</a> |
| BCh    | CM_WKUP_ADC_TSC_CLKCTRL      | This register manages the ADC clocks.   | <a href="#">Section 8.1.12.2.48</a> |
| C0h    | CM_WKUP_SMARTREFLEX0_CLKCTRL | This register manages the SmartReflex0 clocks.  | <a href="#">Section 8.1.12.2.49</a> |
| C4h    | CM_WKUP_TIMER1_CLKCTRL       | This register manages the TIMER1 clocks.  | <a href="#">Section 8.1.12.2.50</a> |

**Table 8-89. CM\_WKUP REGISTERS (continued)**

| Offset | Acronym                       | Register Name   | Section                             |
|--------|-------------------------------|---|-------------------------------------|
| C8h    | CM_WKUP_SMARTREFLEX1_CL_KCTRL | This register manages the SmartReflex1 clocks.  | <a href="#">Section 8.1.12.2.51</a> |
| CCh    | CM_L4_WKUP_AON_CLKSTCTR_L     | This register enables the domain power state transition.<br>It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.2.52</a> |
| D4h    | CM_WKUP_WDT1_CLKCTRL          | This register manages the WDT1 clocks.  | <a href="#">Section 8.1.12.2.53</a> |
| D8h    | CM_DIV_M6_DPLL_CORE           | This register provides controls over the CLKOUT3 o/p of the HSDIVIDER.<br>[warm reset insensitive]  | <a href="#">Section 8.1.12.2.54</a> |

### 8.1.12.2.1 CM\_WKUP\_CLKSTCTRL Register (offset = 0h) [reset = 6h]

CM\_WKUP\_CLKSTCTRL is shown in [Figure 8-83](#) and described in [Table 8-90](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-83. CM\_WKUP\_CLKSTCTRL Register**

|          |                      |                          |                         |                        |                          |           |                          |
|----------|----------------------|--------------------------|-------------------------|------------------------|--------------------------|-----------|--------------------------|
| 31       | 30                   | 29                       | 28                      | 27                     | 26                       | 25        | 24                       |
| Reserved |                      |                          |                         |                        |                          |           |                          |
| R-0h     |                      |                          |                         |                        |                          |           |                          |
| 23       | 22                   | 21                       | 20                      | 19                     | 18                       | 17        | 16                       |
| Reserved |                      |                          |                         |                        |                          |           |                          |
| R-0h     |                      |                          |                         |                        |                          |           |                          |
| 15       | 14                   | 13                       | 12                      | 11                     | 10                       | 9         | 8                        |
| Reserved | CLKACTIVITY_ADC_FCLK | CLKACTIVITY_TIME_R1_GCLK | CLKACTIVITY_UART0_GFCLK | CLKACTIVITY_I2C0_GFCLK | CLKACTIVITY_TIME_R0_GCLK | Reserved  | CLKACTIVITY_GPIO0_GDBCLK |
| R-0h     | R-0h                 | R-0h                     | R-0h                    | R-0h                   | R-0h                     | R-0h      | R-0h                     |
| 7        | 6                    | 5                        | 4                       | 3                      | 2                        | 1         | 0                        |
| Reserved |                      |                          | CLKACTIVITY_WDT1_GCLK   | CLKACTIVITY_SR_SYSCLK  | CLKACTIVITY_L4_WKUP_GCLK | CLKTRCTRL |                          |
| R-0h     |                      |                          | R-0h                    | R-0h                   | R-1h                     | R/W-2h    |                          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-90. CM\_WKUP\_CLKSTCTRL Register Field Descriptions**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 31-15 | Reserved                 | R    | 0h    |  |
| 14    | CLKACTIVITY_ADC_FCLK     | R    | 0h    | This field indicates the state of the ADC clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active              |
| 13    | CLKACTIVITY_TIMER1_GCLK  | R    | 0h    | This field indicates the state of the TIMER1 clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active           |
| 12    | CLKACTIVITY_UART0_GFCLK  | R    | 0h    | This field indicates the state of the UART0 clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active            |
| 11    | CLKACTIVITY_I2C0_GFCLK   | R    | 0h    | This field indicates the state of the I2C0 clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active             |
| 10    | CLKACTIVITY_TIMER0_GCLK  | R    | 0h    | This field indicates the state of the WKUPTIMER_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 9     | Reserved                 | R    | 0h    |  |
| 8     | CLKACTIVITY_GPIO0_GDBCLK | R    | 0h    | This field indicates the state of the WKUPGPIO_DBGICLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active |
| 7-5   | Reserved                 | R    | 0h    |  |
| 4     | CLKACTIVITY_WDT1_GCLK    | R    | 0h    | This field indicates the state of the WDT1_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active        |

**Table 8-90. CM\_WKUP\_CLKSTCTRL Register Field Descriptions (continued)**

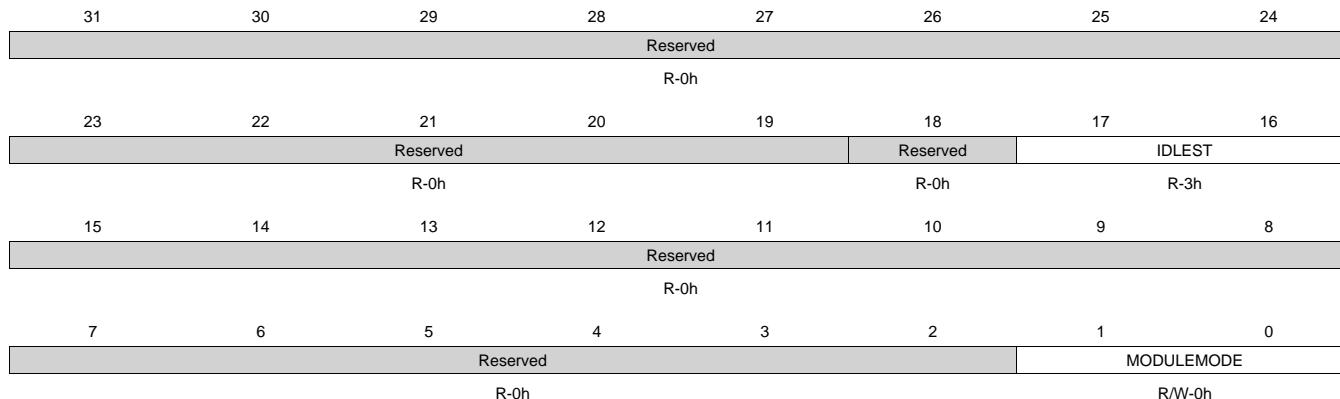
| Bit | Field                    | Type | Reset | Description   |
|-----|--------------------------|------|-------|---|
| 3   | CLKACTIVITY_SR_SYSC_LK   | R    | 0h    | This field indicates the state of the SMARTREFGLEX SYSCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 2   | CLKACTIVITY_L4_WKUP_GCLK | R    | 1h    | This field indicates the state of the L4_WKUP clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 1-0 | CLKTRCTRL                | R/W  | 2h    | Controls the clock state transition of the always on clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.2.2 CM\_WKUP\_CONTROL\_CLKCTRL Register (offset = 4h) [reset = 30000h]

CM\_WKUP\_CONTROL\_CLKCTRL is shown in [Figure 8-84](#) and described in [Table 8-91](#).

This register manages the Control Module clocks.

**Figure 8-84. CM\_WKUP\_CONTROL\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-91. CM\_WKUP\_CONTROL\_CLKCTRL Register Field Descriptions**

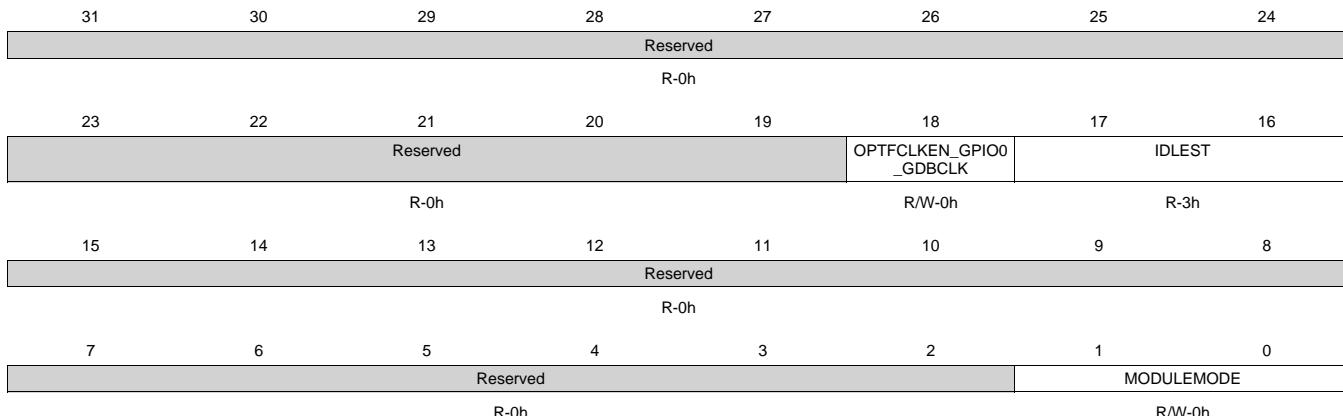
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.3 CM\_WKUP\_GPIO0\_CLKCTRL Register (offset = 8h) [reset = 30000h]

CM\_WKUP\_GPIO0\_CLKCTRL is shown in [Figure 8-85](#) and described in [Table 8-92](#).

This register manages the GPIO0 clocks.

**Figure 8-85. CM\_WKUP\_GPIO0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-92. CM\_WKUP\_GPIO0\_CLKCTRL Register Field Descriptions**

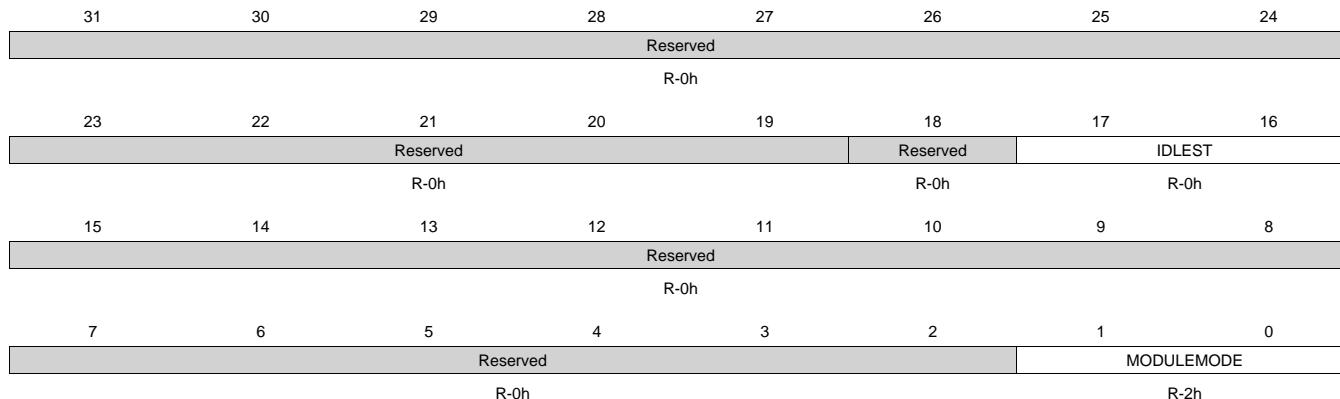
| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-19 | Reserved                | R    | 0h    |   |
| 18    | OPTFCLKEN_GPIO0_GD_BCLK | R/W  | 0h    | Optional functional clock control.<br>0x0 = FCLK_DIS : Optional functional clock is disabled<br>0x1 = FCLK_EN : Optional functional clock is enabled  |
| 17-16 | IDLEST                  | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved                | R    | 0h    |   |
| 1-0   | MODULEMODE              | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

#### 8.1.12.2.4 CM\_WKUP\_L4WKUP\_CLKCTRL Register (offset = Ch) [reset = 2h]

CM\_WKUP\_L4WKUP\_CLKCTRL is shown in [Figure 8-86](#) and described in [Table 8-93](#).

This register manages the L4WKUP clocks.

**Figure 8-86. CM\_WKUP\_L4WKUP\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-93. CM\_WKUP\_L4WKUP\_CLKCTRL Register Field Descriptions**

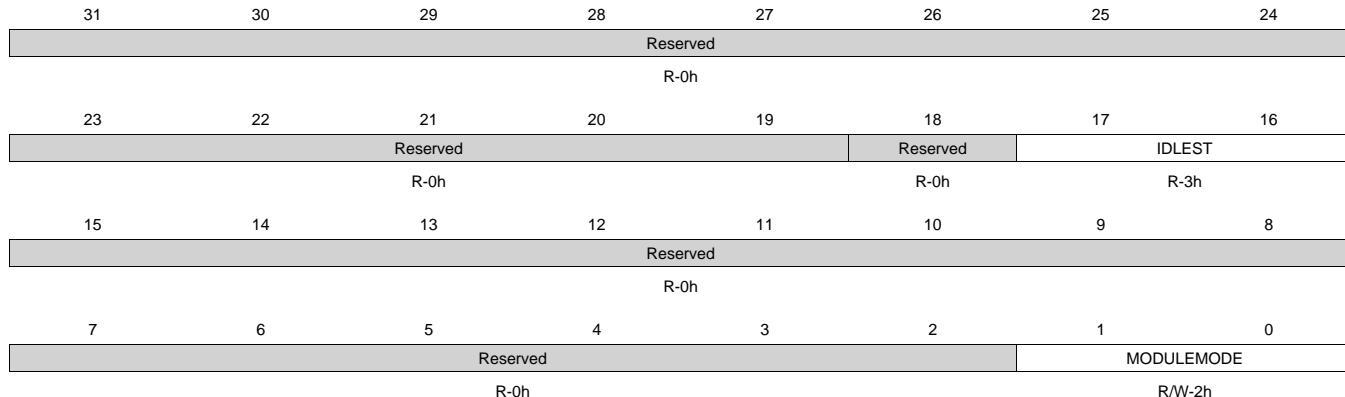
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-19 | Reserved   | R    | 0h    |  |
| 18    | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R    | 2h    | Control the way mandatory clocks are managed.  |

### 8.1.12.2.5 CM\_WKUP\_TIMER0\_CLKCTRL Register (offset = 10h) [reset = 30002h]

CM\_WKUP\_TIMER0\_CLKCTRL is shown in [Figure 8-87](#) and described in [Table 8-94](#).

This register manages the TIMER0 clocks.

**Figure 8-87. CM\_WKUP\_TIMER0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-94. CM\_WKUP\_TIMER0\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.6 CM\_WKUP\_DEBUGSS\_CLKCTRL Register (offset = 14h) [reset = 52580002h]

CM\_WKUP\_DEBUGSS\_CLKCTRL is shown in [Figure 8-88](#) and described in [Table 8-95](#).

This register manages the DEBUGSS clocks.

**Figure 8-88. CM\_WKUP\_DEBUGSS\_CLKCTRL Register**

| 31             | 30                | 29                   | 28     | 27     | 26                | 25         | 24 |
|----------------|-------------------|----------------------|--------|--------|-------------------|------------|----|
| Reserved       | OPTCLK_DEBUG_CLKA | stm_pmd_clkdivsel    |        |        | trc_pmd_clkdivsel |            |    |
| R-0h           | R/W-1h            |                      | R/W-2h |        |                   | R/W-2h     |    |
| 23             | 22                | 21                   | 20     | 19     | 18                | 17         | 16 |
| stm_pmd_clksel | trc_pmd_clksel    | OPTFCLKEN_DBGSY_SCLK |        | STBYST |                   | IDLEST     |    |
| R/W-1h         | R/W-1h            | R/W-1h               |        | R-0h   |                   | R-0h       |    |
| 15             | 14                | 13                   | 12     | 11     | 10                | 9          | 8  |
| Reserved       |                   |                      |        |        |                   |            |    |
| 7              | 6                 | 5                    | 4      | 3      | 2                 | 1          | 0  |
| Reserved       |                   |                      |        |        |                   | MODULEMODE |    |
|                |                   |                      | R-0h   |        |                   | R/W-2h     |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-95. CM\_WKUP\_DEBUGSS\_CLKCTRL Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description   |
|-------|----------------------|------|-------|---|
| 31    | Reserved             | R    | 0h    |   |
| 30    | OPTCLK_DEBUG_CLKA    | R/W  | 1h    |   |
| 29-27 | stm_pmd_clkdivsel    | R/W  | 2h    |   |
| 26-24 | trc_pmd_clkdivsel    | R/W  | 2h    |   |
| 23-22 | stm_pmd_clksel       | R/W  | 1h    |   |
| 21-20 | trc_pmd_clksel       | R/W  | 1h    |   |
| 19    | OPTFCLKEN_DBGSY_SCLK | R/W  | 1h    | Optional functional clock control.<br>0x0 = FCLK_DIS : Optional functional clock is disabled<br>0x1 = FCLK_EN : Optional functional clock is enabled  |
| 18    | STBYST               | R    | 0h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST               | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved             | R    | 0h    |   |
| 1-0   | MODULEMODE           | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.7 CM\_L3\_AON\_CLKSTCTRL Register (offset = 18h) [reset = 1Ah]

CM\_L3\_AON\_CLKSTCTRL is shown in [Figure 8-89](#) and described in [Table 8-96](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-89. CM\_L3\_AON\_CLKSTCTRL Register**

|          |    |    |                            |                             |                           |           |      |  |
|----------|----|----|----------------------------|-----------------------------|---------------------------|-----------|------|--|
| 31       | 30 | 29 | 28                         | 27                          | 26                        | 25        | 24   |  |
| Reserved |    |    |                            |                             |                           | Reserved  |      |  |
| R-0h     |    |    |                            |                             |                           | R-0h      |      |  |
| 23       | 22 | 21 | 20                         | 19                          | 18                        | 17        | 16   |  |
| Reserved |    |    |                            |                             |                           | R-0h      |      |  |
| R-0h     |    |    |                            |                             |                           | R-0h      |      |  |
| 15       | 14 | 13 | 12                         | 11                          | 10                        | 9         | 8    |  |
| Reserved |    |    |                            |                             |                           | Reserved  |      |  |
| R-0h     |    |    |                            |                             |                           | R-0h      |      |  |
| 7        | 6  | 5  | 4                          | 3                           | 2                         | 1         | 0    |  |
| Reserved |    |    | CLKACTIVITY_DEBUG_C<br>LKA | CLKACTIVITY_L3_A<br>ON_GCLK | CLKACTIVITY_DBGSYS<br>CLK | CLKTRCTRL |      |  |
| R-0h     |    |    | R-1h                       |                             | R-1h                      |           | R-0h |  |
| R/W-2h   |    |    |                            |                             |                           |           |      |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-96. CM\_L3\_AON\_CLKSTCTRL Register Field Descriptions**

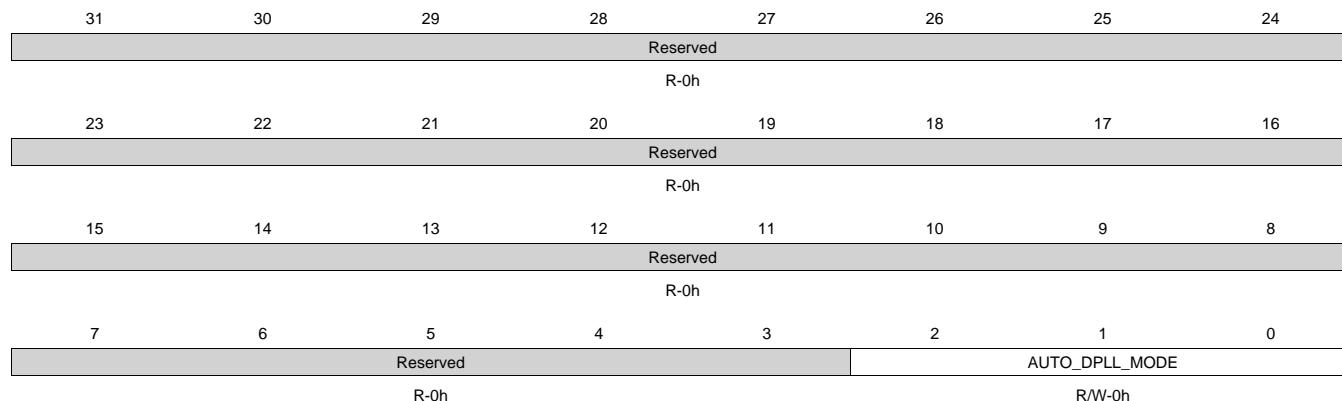
| Bit   | Field                       | Type | Reset | Description  |
|-------|-----------------------------|------|-------|--|
| 31-26 | Reserved                    | R    | 0h    |  |
| 25-11 | Reserved                    | R    | 0h    |  |
| 10-8  | Reserved                    | R    | 0h    |  |
| 7-5   | Reserved                    | R    | 0h    |  |
| 4     | CLKACTIVITY_DEBUG_C<br>LKA  | R    | 1h    | This field indicates the state of the Debugss CLKA clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 3     | CLKACTIVITY_L3_AON_<br>GCLK | R    | 1h    | This field indicates the state of the L3_AON clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 2     | CLKACTIVITY_DBGSYS<br>CLK   | R    | 0h    | This field indicates the state of the Debugss sysclk clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 1-0   | CLKTRCTRL                   | R/W  | 2h    | Controls the clock state transition of the L3 AON clock domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

#### **8.1.12.2.8 CM\_AUTOIDLE\_DPLL\_MPU Register (offset = 1Ch) [reset = 0h]**

CM\_AUTOIDLE\_DPLL\_MPU is shown in [Figure 8-90](#) and described in [Table 8-97](#).

This register provides automatic control over the DPLL activity.

**Figure 8-90. CM\_AUTOIDLE\_DPLL\_MPU Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-97. CM\_AUTOIDLE\_DPLL\_MPU Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description                    |
|------|----------------|------|-------|--------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                |
| 2-0  | AUTO_DPLL_MODE | R/W  | 0h    | This feature is not supported. |

### 8.1.12.2.9 CM\_IDLEST\_DPLL\_MPRegister (offset = 20h) [reset = 0h]

CM\_IDLEST\_DPLL\_MPRegister is shown in Figure 8-91 and described in Table 8-98.

This register allows monitoring the master clock activity. This register is read only and automatically updated.[warm reset insensitive]

**Figure 8-91. CM\_IDLEST\_DPLL\_MPRegister**

|          |    |    |    |    |    |    |    |
|----------|----|----|----|----|----|----|----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 23       | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 15       | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 7        | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-98. CM\_IDLEST\_DPLL\_MPRegister Field Descriptions**

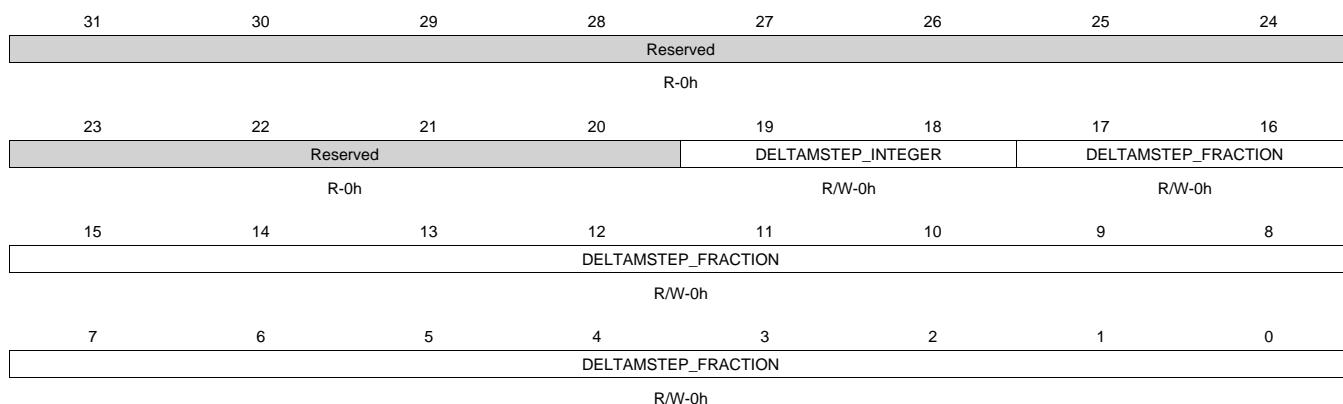
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-9 | Reserved     | R    | 0h    |   |
| 8    | ST_MN_BYPASS | R    | 0h    | DPLL MN_BYPASS status<br>0x0 = NO_MNBYPASS : DPLL is not in MN_Bypass<br>0x1 = MN_BYPASS : DPLL is in MN_Bypass                   |
| 7-1  | Reserved     | R    | 0h    |   |
| 0    | ST_DPLL_CLK  | R    | 0h    | DPLL clock activity<br>0x0 = DPLL_UNLOCKED : DPLL is either in bypass mode or in stop mode.<br>0x1 = DPLL_LOCKED : DPLL is LOCKED |

### 8.1.12.2.10 CM\_SSC\_DELTAMSTEP\_DPLL\_MPRegister (offset = 24h) [reset = 0h]

CM\_SSC\_DELTAMSTEP\_DPLL\_MPRegister is shown in [Figure 8-92](#) and described in [Table 8-99](#).

Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]

**Figure 8-92. CM\_SSC\_DELTAMSTEP\_DPLL\_MPRegister**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-99. CM\_SSC\_DELTAMSTEP\_DPLL\_MPRegister Field Descriptions**

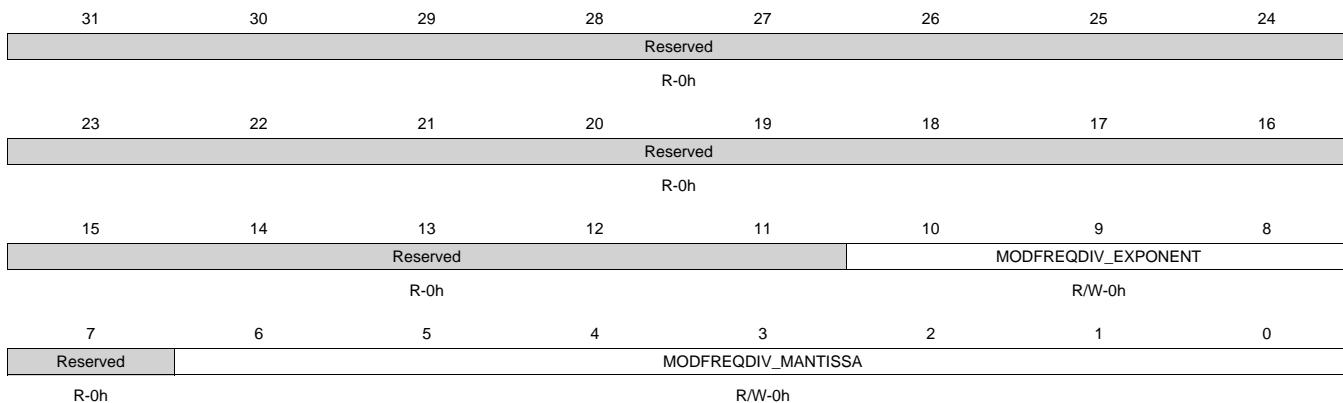
| Bit   | Field               | Type | Reset | Description                            |
|-------|---------------------|------|-------|--|
| 31-20 | Reserved            | R    | 0h    |  |
| 19-18 | DELTAMSTEP_INTEGER  | R/W  | 0h    | Integer part for DeltaM coefficient    |
| 17-0  | DELTAMSTEP_FRACTION | R/W  | 0h    | Fractional part for DeltaM coefficient |

### 8.1.12.2.11 CM\_SSC\_MODFREQDIV\_DPLL\_MPRegister (offset = 28h) [reset = 0h]

CM\_SSC\_MODFREQDIV\_DPLL\_MPRegister is shown in [Figure 8-93](#) and described in [Table 8-100](#).

Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL\_REFCLK/4 Fm = [DPLL\_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV\_MANTISSA \* 2^MODFREQDIV\_EXPONENT [warm reset insensitive]

**Figure 8-93. CM\_SSC\_MODFREQDIV\_DPLL\_MPRegister**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-100. CM\_SSC\_MODFREQDIV\_DPLL\_MPRegister Field Descriptions**

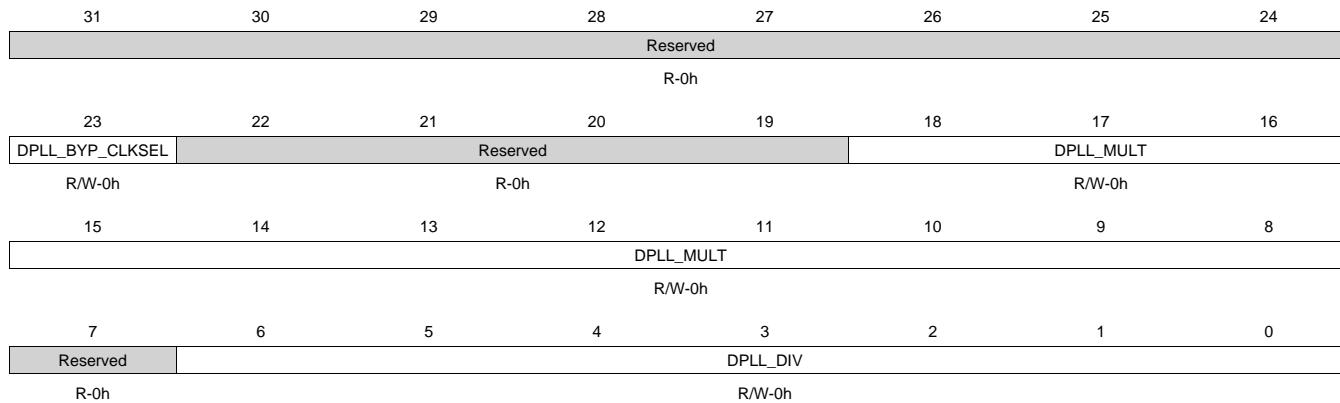
| Bit   | Field               | Type | Reset | Description                                     |
|-------|---------------------|------|-------|---|
| 31-11 | Reserved            | R    | 0h    |   |
| 10-8  | MODFREQDIV_EXPONENT | R/W  | 0h    | Set the Exponent component of MODFREQDIV factor |
| 7     | Reserved            | R    | 0h    |   |
| 6-0   | MODFREQDIV_MANTISSA | R/W  | 0h    | Set the Mantissa component of MODFREQDIV factor |

### 8.1.12.2.12 CM\_CLKSEL\_DPLL\_MPRegister (offset = 2Ch) [reset = 0h]

CM\_CLKSEL\_DPLL\_MPRegister is shown in [Figure 8-94](#) and described in [Table 8-101](#).

This register provides controls over the DPLL.

**Figure 8-94. CM\_CLKSEL\_DPLL\_MPRegister**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-101. CM\_CLKSEL\_DPLL\_MPRegister Field Descriptions**

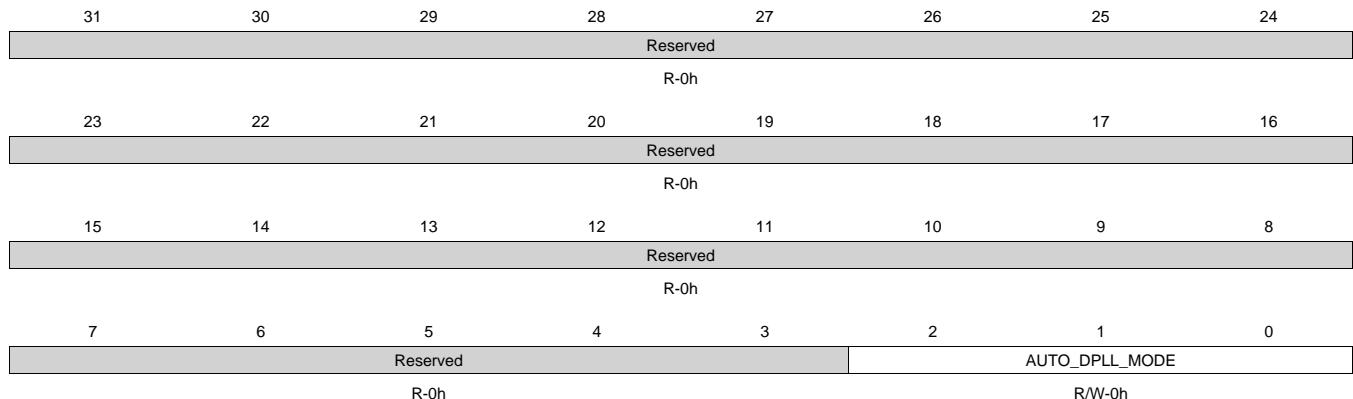
| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | Reserved        | R    | 0h    |   |
| 23    | DPLL_BYP_CLKSEL | R/W  | 0h    | Selects CLKINP or CLKINPULOW as Bypass Clock<br>0x0 = Sel0 : Selects CLKINP Clock as BYPASS Clock<br>0x1 = Sel1 : Selects CLKINPULOW as Bypass Clock  |
| 22-19 | Reserved        | R    | 0h    |   |
| 18-8  | DPLL_MULT       | R/W  | 0h    | DPLL multiplier factor (2 to 2047).<br>This register is automatically cleared to 0 when the DPLL_EN field in the *CLKMODE_DPLL* register is set to select MN Bypass mode.<br>M=2 to<br>2047 => DPLL multiplies by M).<br>0x0 = 0 : Reserved<br>0x1 = 1 : Reserved |
| 7     | Reserved        | R    | 0h    |   |
| 6-0   | DPLL_DIV        | R/W  | 0h    | DPLL divider factor (0 to 127) (equal to input N of DPLL<br>actual division factor is N+1).   |

### 8.1.12.2.13 CM\_AUTOIDLE\_DPLL\_DDR Register (offset = 30h) [reset = 0h]

CM\_AUTOIDLE\_DPLL\_DDR is shown in [Figure 8-95](#) and described in [Table 8-102](#).

This register provides automatic control over the DPLL activity.

**Figure 8-95. CM\_AUTOIDLE\_DPLL\_DDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-102. CM\_AUTOIDLE\_DPLL\_DDR Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description                      |
|------|----------------|------|-------|----------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                  |
| 2-0  | AUTO_DPLL_MODE | R/W  | 0h    | AUTO_DPLL_MODE is not supported. |

#### **8.1.12.2.14 CM\_IDLEST\_DPLL\_DDR Register (offset = 34h) [reset = 0h]**

CM\_IDLEST\_DPLL\_DDR is shown in Figure 8-96 and described in Table 8-103.

This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]

**Figure 8-96. CM\_IDLEST\_DPLL\_DDR Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|----------|----|----|----|----|----|----|----|
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 23       | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 15       | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |
| 7        | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved |    |    |    |    |    |    |    |
| R-0h     |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-103. CM\_IDLEST\_DPLL\_DDR Register Field Descriptions**

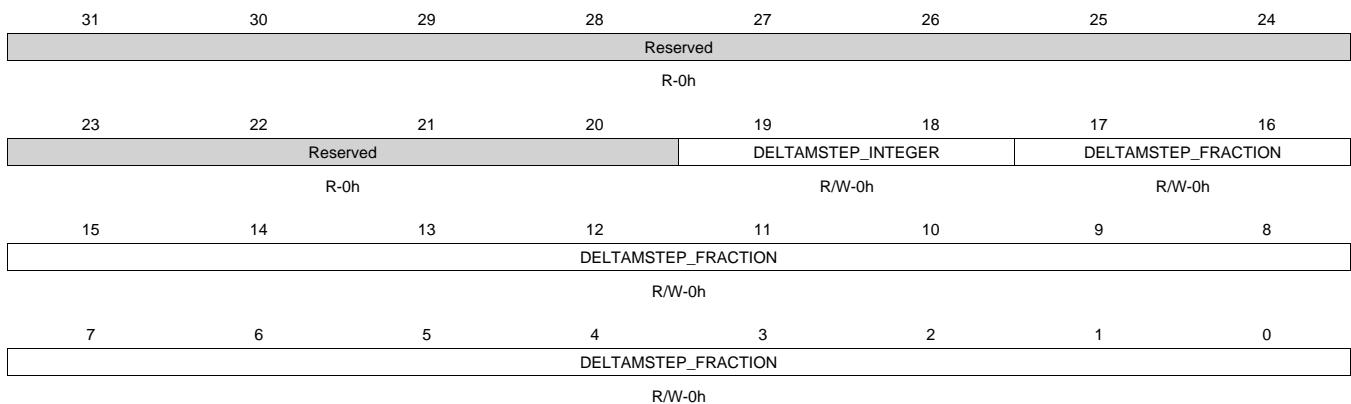
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-9 | Reserved     | R    | 0h    |   |
| 8    | ST_MN_BYPASS | R    | 0h    | DPLL MN_BYPASS status<br>0x0 = NO_MNBYPASS : DPLL is not in MN_Bypass<br>0x1 = MN_BYPASS : DPLL is in MN_Bypass                   |
| 7-1  | Reserved     | R    | 0h    |   |
| 0    | ST_DPLL_CLK  | R    | 0h    | DPLL clock activity<br>0x0 = DPLL_UNLOCKED : DPLL is either in bypass mode or in stop mode.<br>0x1 = DPLL_LOCKED : DPLL is LOCKED |

### 8.1.12.2.15 CM\_SSC\_DELTAMSTEP\_DPLL\_DDR Register (offset = 38h) [reset = 0h]

CM\_SSC\_DELTAMSTEP\_DPLL\_DDR is shown in [Figure 8-97](#) and described in [Table 8-104](#).

Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]

**Figure 8-97. CM\_SSC\_DELTAMSTEP\_DPLL\_DDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-104. CM\_SSC\_DELTAMSTEP\_DPLL\_DDR Register Field Descriptions**

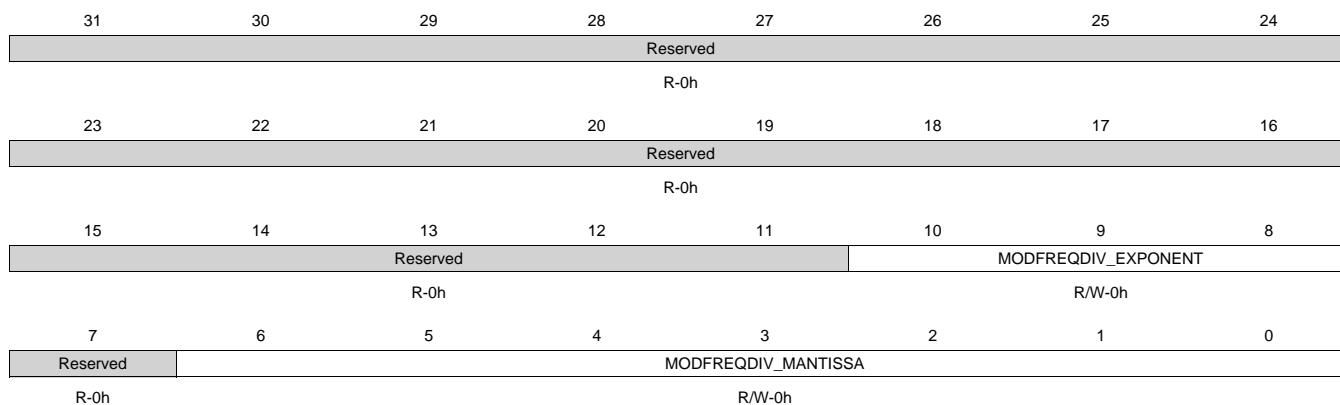
| Bit   | Field               | Type | Reset | Description                                 |
|-------|---------------------|------|-------|---|
| 31-20 | Reserved            | R    | 0h    |   |
| 19-18 | DELTAMSTEP_INTEGER  | R/W  | 0h    | Integer part for DeltaM coefficient         |
| 17-0  | DELTAMSTEP_FRACTION | R/W  | 0h    | Fractional setting for DeltaMStep parameter |

### 8.1.12.2.16 CM\_SSC\_MODFREQDIV\_DPLL\_DDR Register (offset = 3Ch) [reset = 0h]

CM\_SSC\_MODFREQDIV\_DPLL\_DDR is shown in [Figure 8-98](#) and described in [Table 8-105](#).

Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL\_REFCLK/4 Fm = [DPLL\_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV\_MANTISSA \* 2^MODFREQDIV\_EXPONENT [warm reset insensitive]

**Figure 8-98. CM\_SSC\_MODFREQDIV\_DPLL\_DDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-105. CM\_SSC\_MODFREQDIV\_DPLL\_DDR Register Field Descriptions**

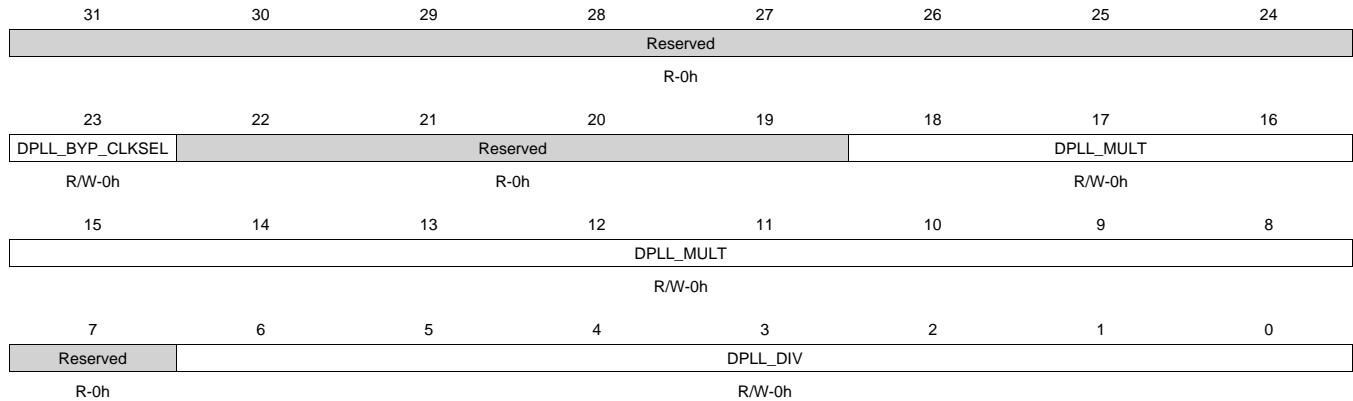
| Bit   | Field               | Type | Reset | Description                                     |
|-------|---------------------|------|-------|---|
| 31-11 | Reserved            | R    | 0h    |   |
| 10-8  | MODFREQDIV_EXPONENT | R/W  | 0h    | Set the Exponent component of MODFREQDIV factor |
| 7     | Reserved            | R    | 0h    |   |
| 6-0   | MODFREQDIV_MANTISSA | R/W  | 0h    | Set the Mantissa component of MODFREQDIV factor |

### 8.1.12.2.17 CM\_CLKSEL\_DPLL\_DDR Register (offset = 40h) [reset = 0h]

CM\_CLKSEL\_DPLL\_DDR is shown in [Figure 8-99](#) and described in [Table 8-106](#).

This register provides controls over the DPLL.

**Figure 8-99. CM\_CLKSEL\_DPLL\_DDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-106. CM\_CLKSEL\_DPLL\_DDR Register Field Descriptions**

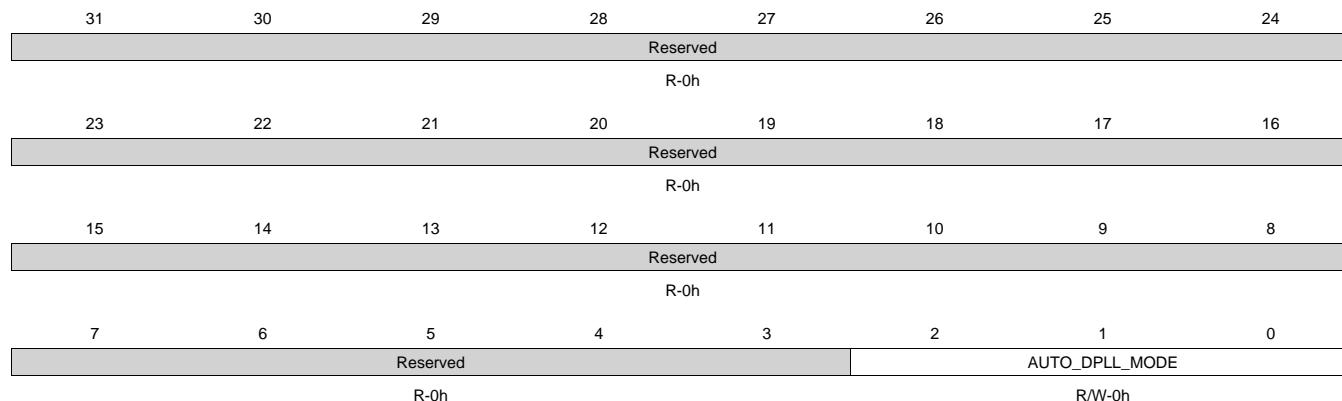
| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | Reserved        | R    | 0h    |   |
| 23    | DPLL_BYP_CLKSEL | R/W  | 0h    | Select CLKINP or CLKINPULOW as bypass clock<br>0x0 = Sel0 : Selects CLKINP Clock as BYPASS Clock<br>0x1 = Sel1 : Selects CLKINPULOW as Bypass Clock   |
| 22-19 | Reserved        | R    | 0h    |   |
| 18-8  | DPLL_MULT       | R/W  | 0h    | DPLL multiplier factor (2 to 2047).<br>This register is automatically cleared to 0 when the DPLL_EN field in the *CLKMODE_DPLL* register is set to select MN Bypass mode.<br>M=2 to<br>2047 => DPLL multiplies by M).<br>0x0 = 0 : Reserved<br>0x1 = 1 : Reserved |
| 7     | Reserved        | R    | 0h    |   |
| 6-0   | DPLL_DIV        | R/W  | 0h    | DPLL divider factor (0 to 127) (equal to input N of DPLL<br>actual division factor is N+1).   |

### 8.1.12.2.18 CM\_AUTOIDLE\_DPLL\_DISP Register (offset = 44h) [reset = 0h]

CM\_AUTOIDLE\_DPLL\_DISP is shown in [Figure 8-100](#) and described in [Table 8-107](#).

This register provides automatic control over the DPLL activity.

**Figure 8-100. CM\_AUTOIDLE\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-107. CM\_AUTOIDLE\_DPLL\_DISP Register Field Descriptions**

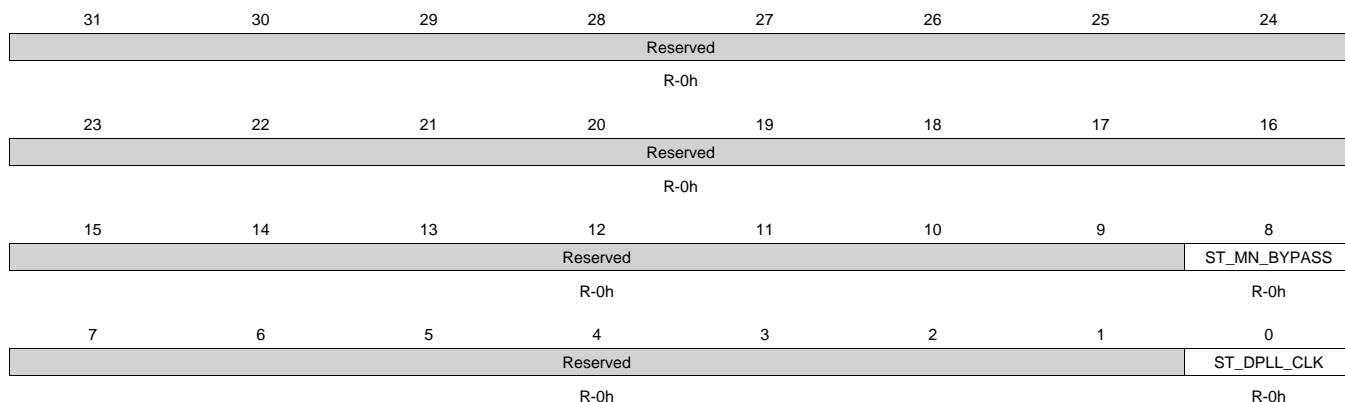
| Bit  | Field          | Type | Reset | Description                      |
|------|----------------|------|-------|----------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                  |
| 2-0  | AUTO_DPLL_MODE | R/W  | 0h    | AUTO_DPLL_MODE is not supported. |

### 8.1.12.2.19 CM\_IDLEST\_DPLL\_DISP Register (offset = 48h) [reset = 0h]

CM\_IDLEST\_DPLL\_DISP is shown in [Figure 8-101](#) and described in [Table 8-108](#).

This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]

**Figure 8-101. CM\_IDLEST\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-108. CM\_IDLEST\_DPLL\_DISP Register Field Descriptions**

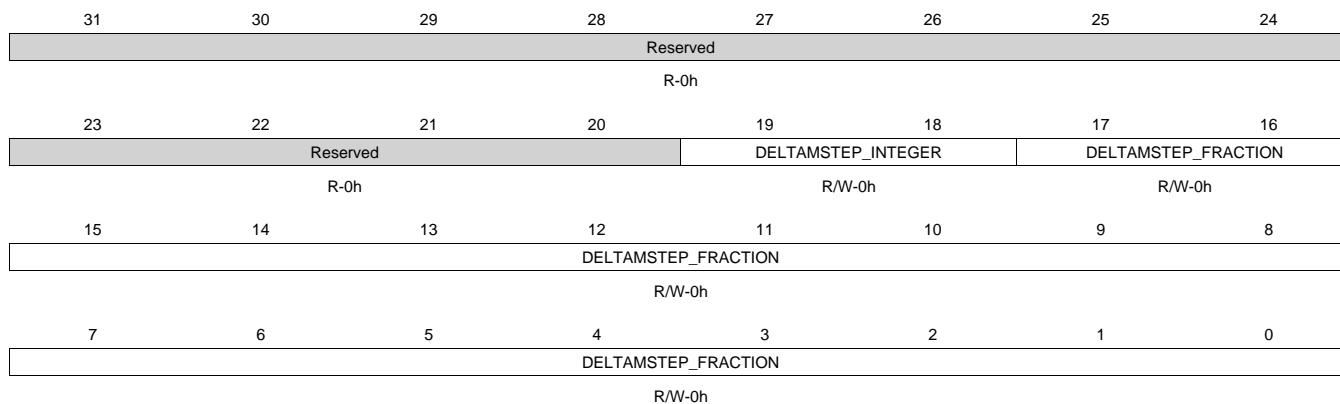
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-9 | Reserved     | R    | 0h    |   |
| 8    | ST_MN_BYPASS | R    | 0h    | DPLL MN_BYPASS status<br>0x0 = NO_MNBYPASS : DPLL is not in MN_Bypass<br>0x1 = MN_BYPASS : DPLL is in MN_Bypass                   |
| 7-1  | Reserved     | R    | 0h    |   |
| 0    | ST_DPLL_CLK  | R    | 0h    | DPLL clock activity<br>0x0 = DPLL_UNLOCKED : DPLL is either in bypass mode or in stop mode.<br>0x1 = DPLL_LOCKED : DPLL is LOCKED |

### 8.1.12.2.20 CM\_SSC\_DELTAMSTEP\_DPLL\_DISP Register (offset = 4Ch) [reset = 0h]

CM\_SSC\_DELTAMSTEP\_DPLL\_DISP is shown in [Figure 8-102](#) and described in [Table 8-109](#).

Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]

**Figure 8-102. CM\_SSC\_DELTAMSTEP\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-109. CM\_SSC\_DELTAMSTEP\_DPLL\_DISP Register Field Descriptions**

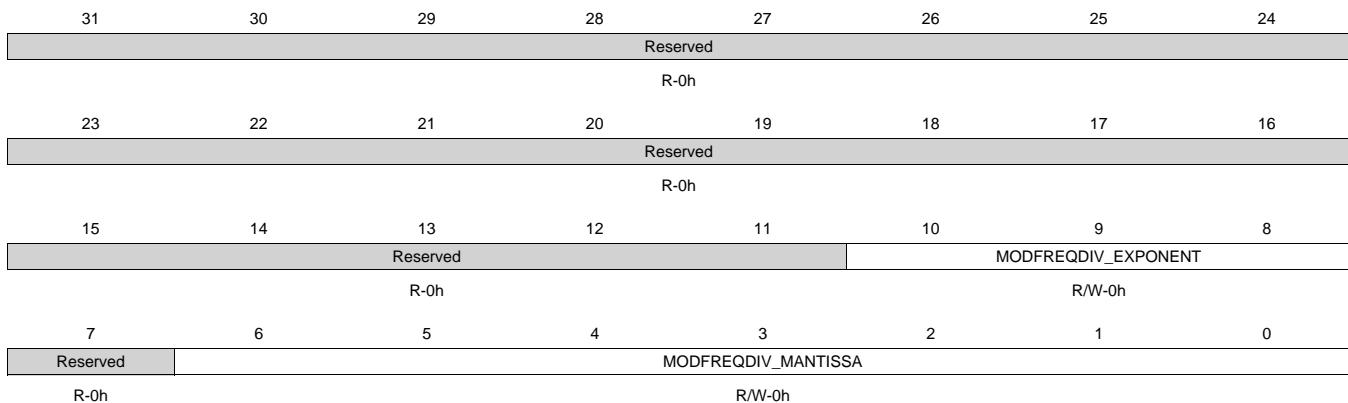
| Bit   | Field               | Type | Reset | Description                                 |
|-------|---------------------|------|-------|---|
| 31-20 | Reserved            | R    | 0h    |   |
| 19-18 | DELTAMSTEP_INTEGER  | R/W  | 0h    | Integer part for DeltaM coefficient         |
| 17-0  | DELTAMSTEP_FRACTION | R/W  | 0h    | Fractional setting for DeltaMStep parameter |

### 8.1.12.2.21 CM\_SSC\_MODFREQDIV\_DPLL\_DISP Register (offset = 50h) [reset = 0h]

CM\_SSC\_MODFREQDIV\_DPLL\_DISP is shown in [Figure 8-103](#) and described in [Table 8-110](#).

Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL\_REFCLK/4 Fm = [DPLL\_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV\_MANTISSA \* 2^MODFREQDIV\_EXPONENT [warm reset insensitive]

**Figure 8-103. CM\_SSC\_MODFREQDIV\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-110. CM\_SSC\_MODFREQDIV\_DPLL\_DISP Register Field Descriptions**

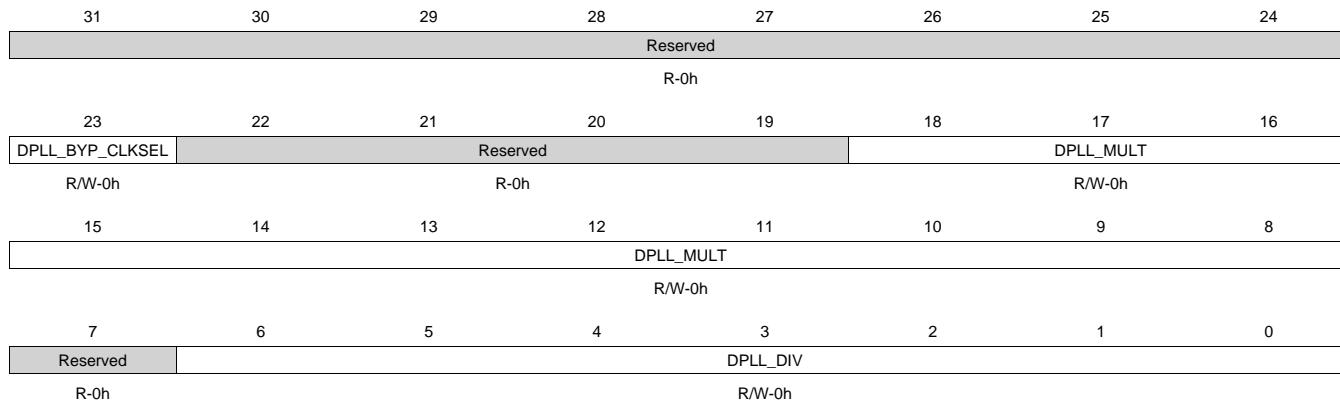
| Bit   | Field               | Type | Reset | Description                                     |
|-------|---------------------|------|-------|---|
| 31-11 | Reserved            | R    | 0h    |   |
| 10-8  | MODFREQDIV_EXPONENT | R/W  | 0h    | Set the Exponent component of MODFREQDIV factor |
| 7     | Reserved            | R    | 0h    |   |
| 6-0   | MODFREQDIV_MANTISSA | R/W  | 0h    | Set the Mantissa component of MODFREQDIV factor |

### 8.1.12.2.22 CM\_CLKSEL\_DPLL\_DISP Register (offset = 54h) [reset = 0h]

CM\_CLKSEL\_DPLL\_DISP is shown in [Figure 8-104](#) and described in [Table 8-111](#).

This register provides controls over the DPLL.

**Figure 8-104. CM\_CLKSEL\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-111. CM\_CLKSEL\_DPLL\_DISP Register Field Descriptions**

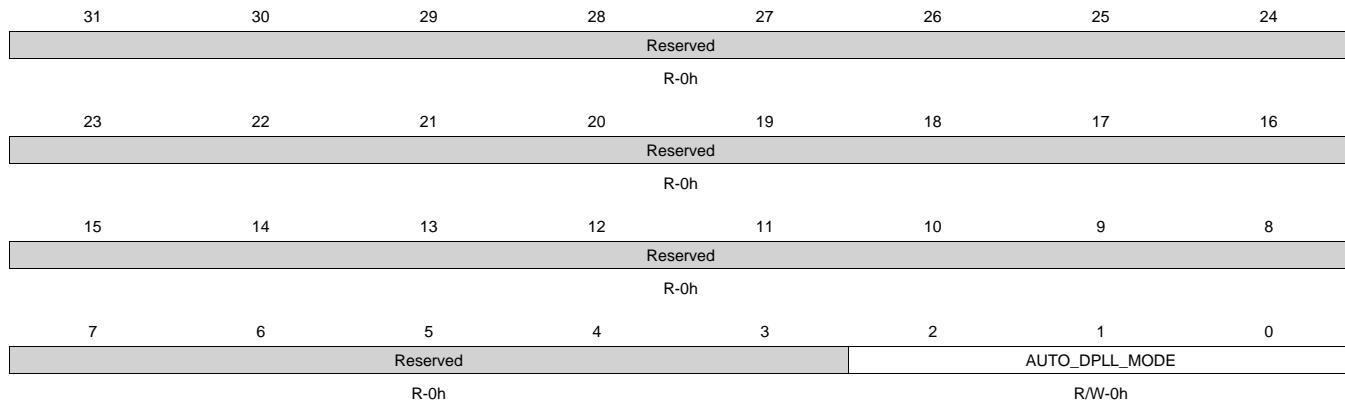
| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | Reserved        | R    | 0h    |   |
| 23    | DPLL_BYP_CLKSEL | R/W  | 0h    | Select CLKINP or CLKINPULOW as bypass clock<br>0x0 = Sel0 : Selects CLKINP Clock as BYPASS Clock<br>0x1 = Sel1 : Selects CLKINPULOW as Bypass Clock   |
| 22-19 | Reserved        | R    | 0h    |   |
| 18-8  | DPLL_MULT       | R/W  | 0h    | DPLL multiplier factor (2 to 2047).<br>This register is automatically cleared to 0 when the DPLL_EN field in the *CLKMODE_DPLL* register is set to select MN Bypass mode.<br>M=2 to<br>2047 => DPLL multiplies by M).<br>0x0 = 0 : Reserved<br>0x1 = 1 : Reserved |
| 7     | Reserved        | R    | 0h    |   |
| 6-0   | DPLL_DIV        | R/W  | 0h    | DPLL divider factor (0 to 127) (equal to input N of DPLL<br>actual division factor is N+1).   |

### 8.1.12.2.23 CM\_AUTOIDLE\_DPLL\_CORE Register (offset = 58h) [reset = 0h]

CM\_AUTOIDLE\_DPLL\_CORE is shown in [Figure 8-105](#) and described in [Table 8-112](#).

This register provides automatic control over the DPLL activity.

**Figure 8-105. CM\_AUTOIDLE\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-112. CM\_AUTOIDLE\_DPLL\_CORE Register Field Descriptions**

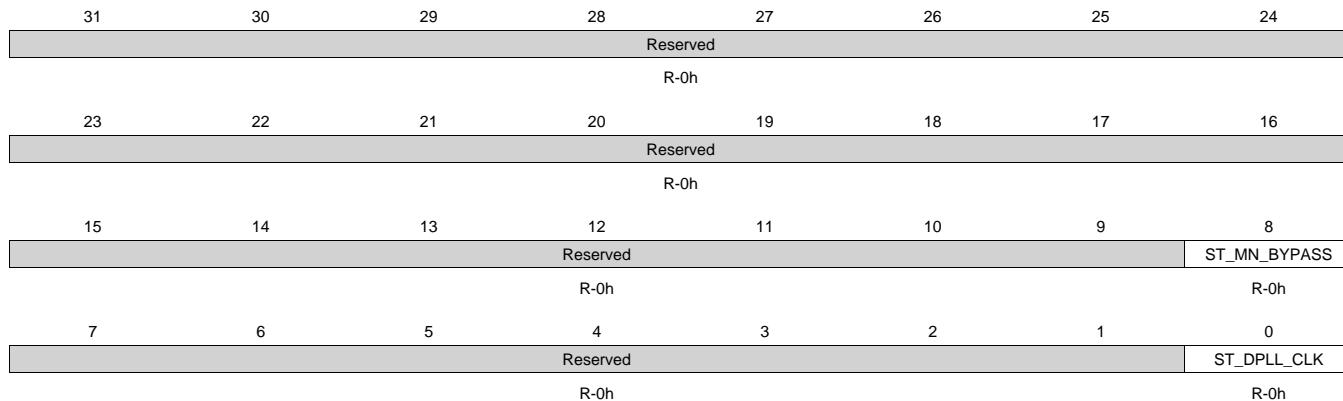
| Bit  | Field          | Type | Reset | Description                    |
|------|----------------|------|-------|--------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                |
| 2-0  | AUTO_DPLL_MODE | R/W  | 0h    | This feature is not supported. |

### 8.1.12.2.24 CM\_IDLEST\_DPLL\_CORE Register (offset = 5Ch) [reset = 0h]

CM\_IDLEST\_DPLL\_CORE is shown in [Figure 8-106](#) and described in [Table 8-113](#).

This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]

**Figure 8-106. CM\_IDLEST\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-113. CM\_IDLEST\_DPLL\_CORE Register Field Descriptions**

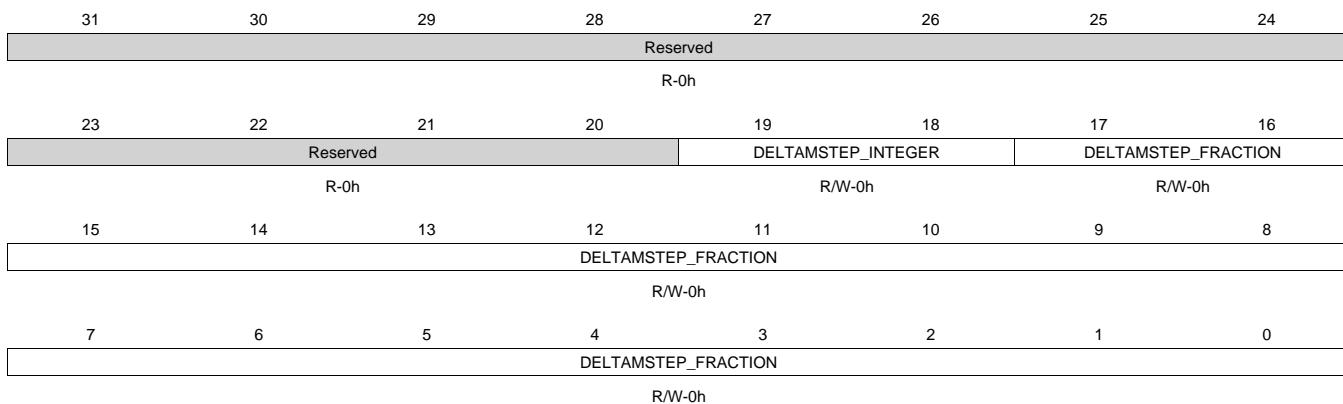
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-9 | Reserved     | R    | 0h    |   |
| 8    | ST_MN_BYPASS | R    | 0h    | DPLL MN_BYPASS status<br>0x0 = NO_MNBYPASS : DPLL is not in MN_Bypass<br>0x1 = MN_BYPASS : DPLL is in MN_Bypass                   |
| 7-1  | Reserved     | R    | 0h    |   |
| 0    | ST_DPLL_CLK  | R    | 0h    | DPLL clock activity<br>0x0 = DPLL_UNLOCKED : DPLL is either in bypass mode or in stop mode.<br>0x1 = DPLL_LOCKED : DPLL is LOCKED |

### 8.1.12.2.25 CM\_SSC\_DELTAMSTEP\_DPLL\_CORE Register (offset = 60h) [reset = 0h]

CM\_SSC\_DELTAMSTEP\_DPLL\_CORE is shown in [Figure 8-107](#) and described in [Table 8-114](#).

Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]

**Figure 8-107. CM\_SSC\_DELTAMSTEP\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-114. CM\_SSC\_DELTAMSTEP\_DPLL\_CORE Register Field Descriptions**

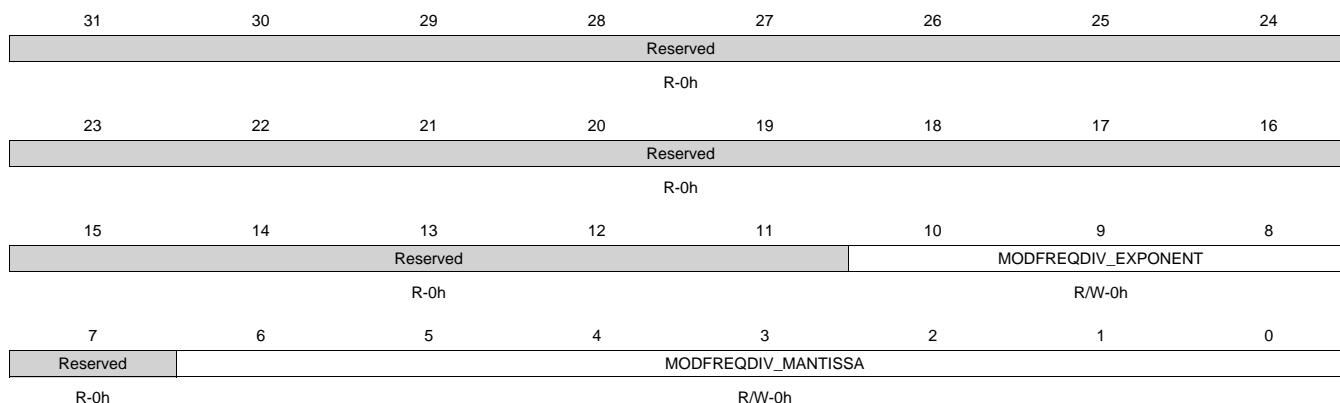
| Bit   | Field               | Type | Reset | Description                                 |
|-------|---------------------|------|-------|---|
| 31-20 | Reserved            | R    | 0h    |   |
| 19-18 | DELTAMSTEP_INTEGER  | R/W  | 0h    | Integer part for DeltaM coefficient         |
| 17-0  | DELTAMSTEP_FRACTION | R/W  | 0h    | Fractional setting for DeltaMStep parameter |

### 8.1.12.2.26 CM\_SSC\_MODFREQDIV\_DPLL\_CORE Register (offset = 64h) [reset = 0h]

CM\_SSC\_MODFREQDIV\_DPLL\_CORE is shown in [Figure 8-108](#) and described in [Table 8-115](#).

Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL\_REFCLK/4 Fm = [DPLL\_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV\_MANTISSA \* 2^MODFREQDIV\_EXPONENT [warm reset insensitive]

**Figure 8-108. CM\_SSC\_MODFREQDIV\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-115. CM\_SSC\_MODFREQDIV\_DPLL\_CORE Register Field Descriptions**

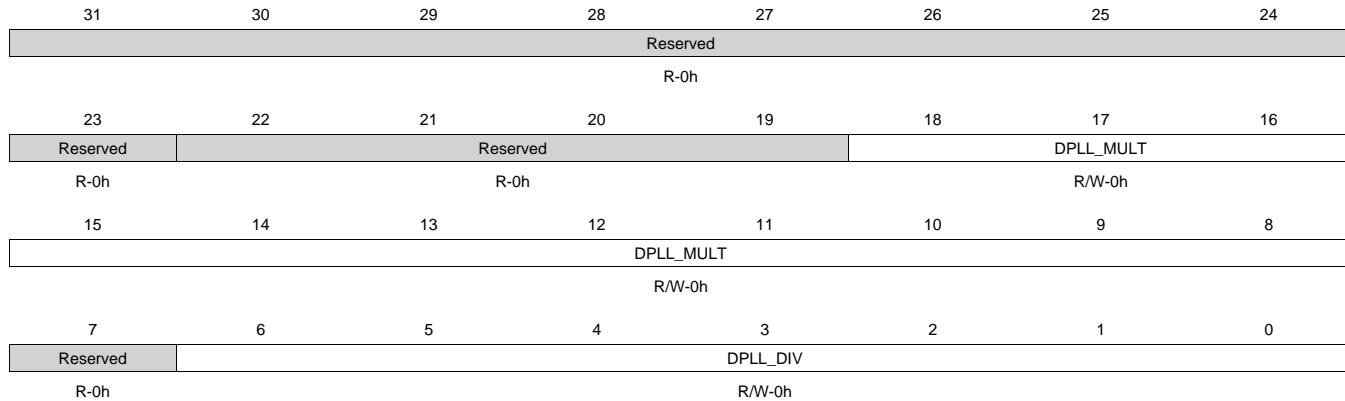
| Bit   | Field               | Type | Reset | Description                                     |
|-------|---------------------|------|-------|---|
| 31-11 | Reserved            | R    | 0h    |   |
| 10-8  | MODFREQDIV_EXPONENT | R/W  | 0h    | Set the Exponent component of MODFREQDIV factor |
| 7     | Reserved            | R    | 0h    |   |
| 6-0   | MODFREQDIV_MANTISSA | R/W  | 0h    | Set the Mantissa component of MODFREQDIV factor |

### 8.1.12.2.27 CM\_CLKSEL\_DPLL\_CORE Register (offset = 68h) [reset = 0h]

CM\_CLKSEL\_DPLL\_CORE is shown in [Figure 8-109](#) and described in [Table 8-116](#).

This register provides controls over the DPLL.

**Figure 8-109. CM\_CLKSEL\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-116. CM\_CLKSEL\_DPLL\_CORE Register Field Descriptions**

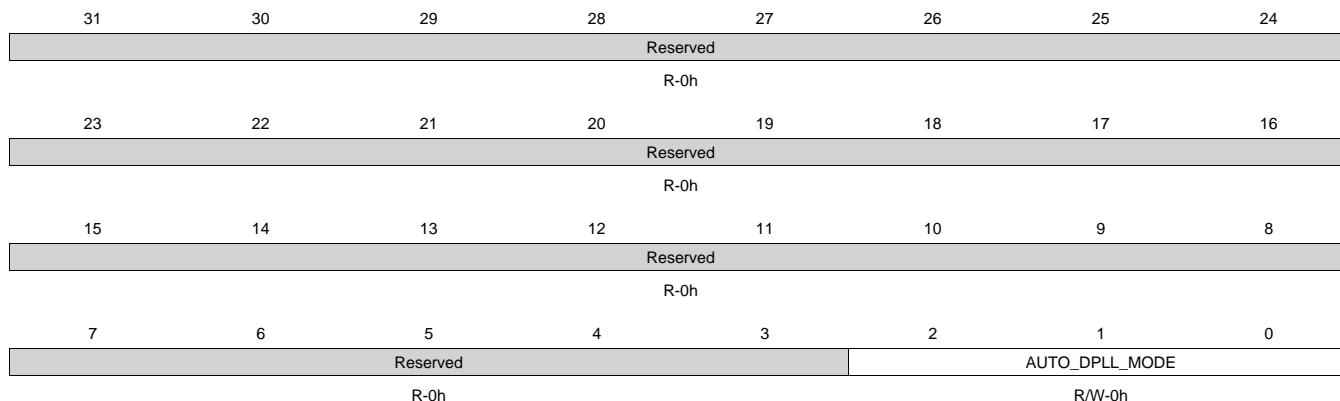
| Bit   | Field     | Type | Reset | Description   |
|-------|-----------|------|-------|---|
| 31-23 | Reserved  | R    | 0h    |   |
| 22-19 | Reserved  | R    | 0h    |   |
| 18-8  | DPLL_MULT | R/W  | 0h    | DPLL multiplier factor (2 to 2047).<br>This register is automatically cleared to 0 when the DPLL_EN field in the *CLKMODE_DPLL* register is set to select MN Bypass mode.<br>M=2 to<br>2047 => DPLL multiplies by M<br>0x0 = Reserved_0 : Reserved<br>0x1 = Reserved_1 : Reserved |
| 7     | Reserved  | R    | 0h    |   |
| 6-0   | DPLL_DIV  | R/W  | 0h    | DPLL divider factor (0 to 127) (equal to input N of DPLL actual division factor is N+1).  |

### 8.1.12.2.28 CM\_AUTOIDLE\_DPLL\_PER Register (offset = 6Ch) [reset = 0h]

CM\_AUTOIDLE\_DPLL\_PER is shown in [Figure 8-110](#) and described in [Table 8-117](#).

This register provides automatic control over the DPLL activity.

**Figure 8-110. CM\_AUTOIDLE\_DPLL\_PER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-117. CM\_AUTOIDLE\_DPLL\_PER Register Field Descriptions**

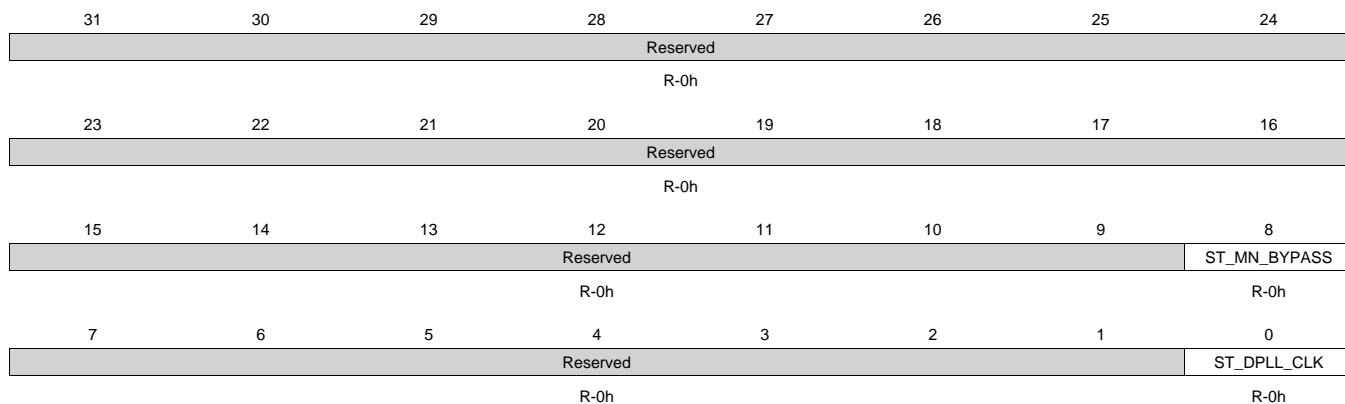
| Bit  | Field          | Type | Reset | Description                      |
|------|----------------|------|-------|----------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                  |
| 2-0  | AUTO_DPLL_MODE | R/W  | 0h    | AUTO_DPLL_MODE is not supported. |

### 8.1.12.2.29 CM\_IDLEST\_DPLL\_PER Register (offset = 70h) [reset = 0h]

CM\_IDLEST\_DPLL\_PER is shown in [Figure 8-111](#) and described in [Table 8-118](#).

This register allows monitoring the master clock activity. This register is read only and automatically updated. [warm reset insensitive]

**Figure 8-111. CM\_IDLEST\_DPLL\_PER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-118. CM\_IDLEST\_DPLL\_PER Register Field Descriptions**

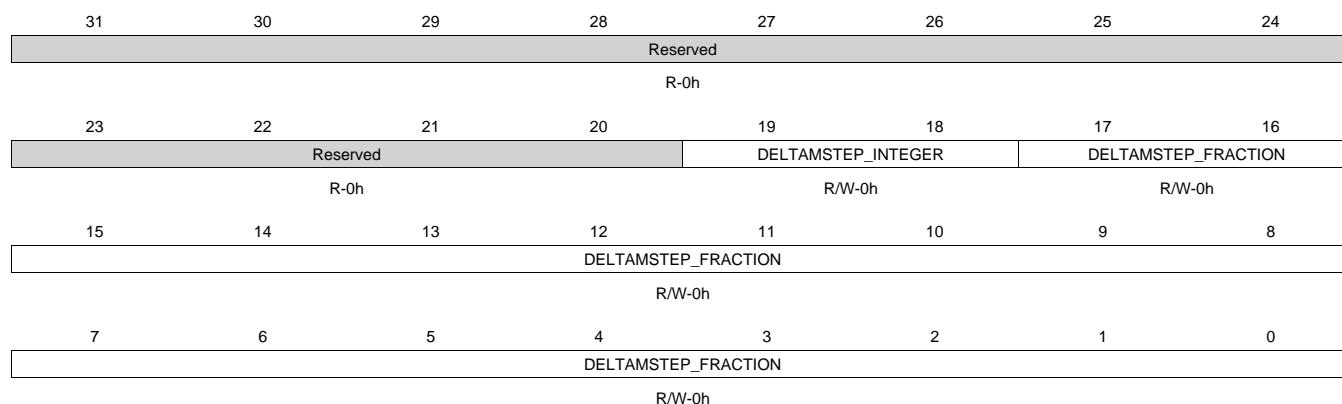
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-9 | Reserved     | R    | 0h    |   |
| 8    | ST_MN_BYPASS | R    | 0h    | DPLL MN_BYPASS status<br>0x0 = NO_MNBYPASS : DPLL is not in MN_Bypass<br>0x1 = MN_BYPASS : DPLL is in MN_Bypass                   |
| 7-1  | Reserved     | R    | 0h    |   |
| 0    | ST_DPLL_CLK  | R    | 0h    | DPLL clock activity<br>0x0 = DPLL_UNLOCKED : DPLL is either in bypass mode or in stop mode.<br>0x1 = DPLL_LOCKED : DPLL is LOCKED |

### 8.1.12.2.30 CM\_SSC\_DELTAMSTEP\_DPLL\_PER Register (offset = 74h) [reset = 0h]

CM\_SSC\_DELTAMSTEP\_DPLL\_PER is shown in [Figure 8-112](#) and described in [Table 8-119](#).

Control the DeltaMStep parameter for Spread Spectrum Clocking technique DeltaMStep is split into fractional and integer part. [warm reset insensitive]

**Figure 8-112. CM\_SSC\_DELTAMSTEP\_DPLL\_PER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

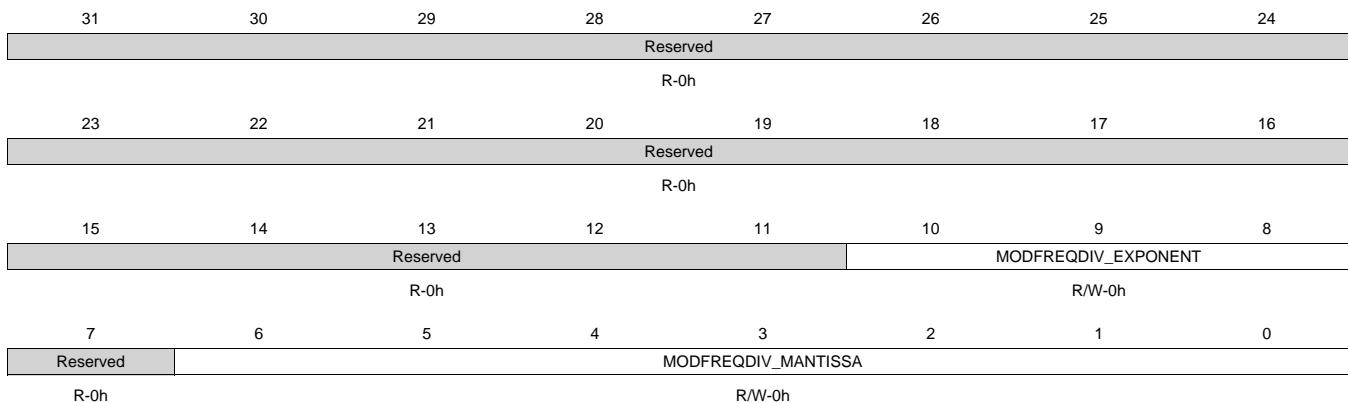
**Table 8-119. CM\_SSC\_DELTAMSTEP\_DPLL\_PER Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description                                 |
|-------|---------------------|------|-------|---|
| 31-20 | Reserved            | R    | 0h    |   |
| 19-18 | DELTAMSTEP_INTEGER  | R/W  | 0h    | Integer part for DeltaM coefficient         |
| 17-0  | DELTAMSTEP_FRACTION | R/W  | 0h    | Fractional setting for DeltaMStep parameter |

**8.1.12.2.31 CM\_SSC\_MODFREQDIV\_DPLL\_PER Register (offset = 78h) [reset = 0h]**

CM\_SSC\_MODFREQDIV\_DPLL\_PER is shown in [Figure 8-113](#) and described in [Table 8-120](#).

Control the Modulation Frequency (Fm) for Spread Spectrum Clocking technique by defining it as a ratio of DPLL\_REFCLK/4 Fm = [DPLL\_REFCLK/4]/MODFREQDIV MODFREQDIV = MODFREQDIV\_MANTISSA \* 2^MODFREQDIV\_EXPONENT [warm reset insensitive]

**Figure 8-113. CM\_SSC\_MODFREQDIV\_DPLL\_PER Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-120. CM\_SSC\_MODFREQDIV\_DPLL\_PER Register Field Descriptions**

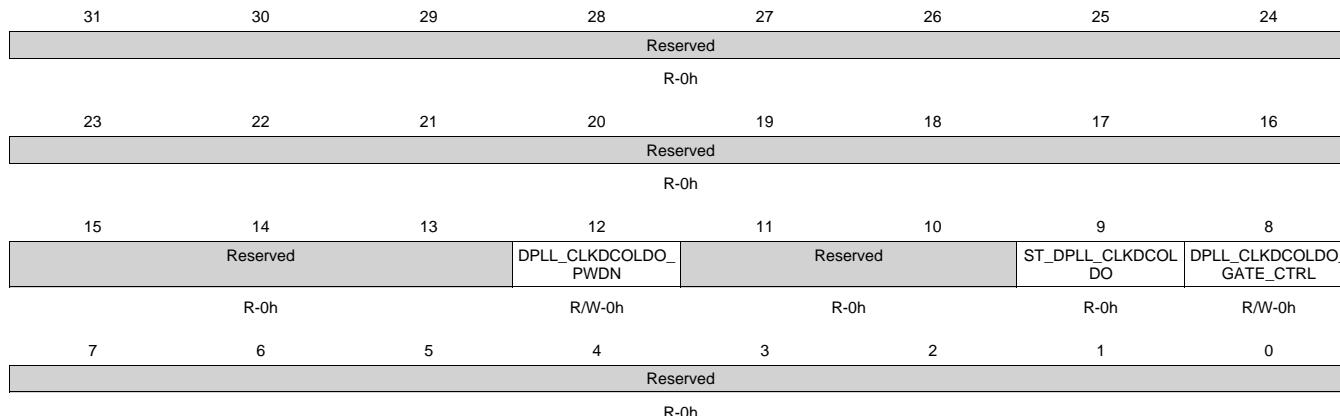
| Bit   | Field               | Type | Reset | Description                                     |
|-------|---------------------|------|-------|---|
| 31-11 | Reserved            | R    | 0h    |   |
| 10-8  | MODFREQDIV_EXPONENT | R/W  | 0h    | Set the Exponent component of MODFREQDIV factor |
| 7     | Reserved            | R    | 0h    |   |
| 6-0   | MODFREQDIV_MANTISSA | R/W  | 0h    | Set the Mantissa component of MODFREQDIV factor |

### 8.1.12.2.32 CM\_CLKDCOLDO\_DPLL\_PER Register (offset = 7Ch) [reset = 0h]

CM\_CLKDCOLDO\_DPLL\_PER is shown in [Figure 8-114](#) and described in [Table 8-121](#).

This register provides controls over the digitally controlled oscillator output of the PER DPLL.

**Figure 8-114. CM\_CLKDCOLDO\_DPLL\_PER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-121. CM\_CLKDCOLDO\_DPLL\_PER Register Field Descriptions**

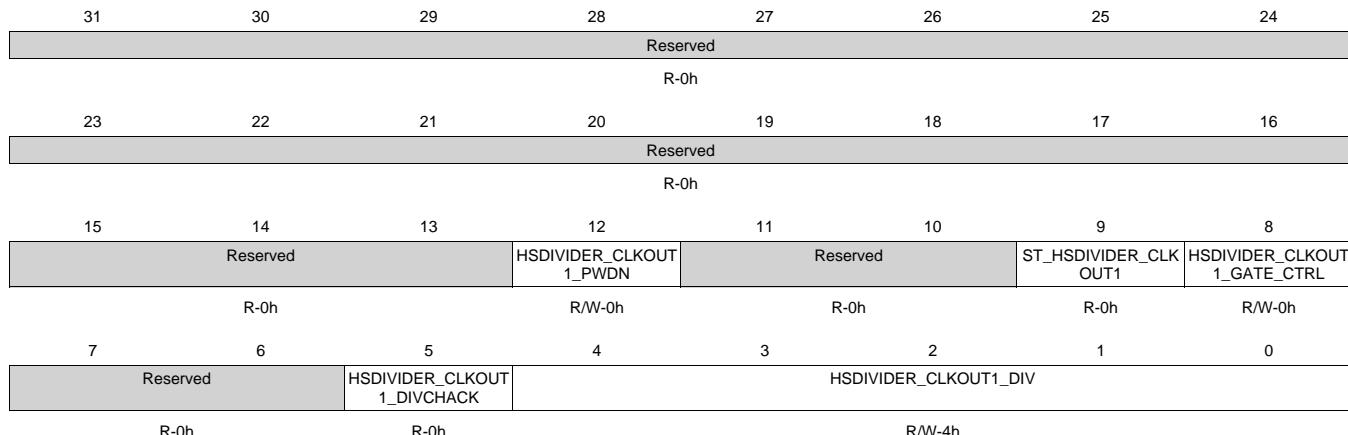
| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 31-13 | Reserved                 | R    | 0h    |  |
| 12    | DPLL_CLKDCOLDO_PWDN      | R/W  | 0h    | Automatic power down for CLKDCOLDO o/p when it is gated<br>0x0 = ALWAYS_ACTIVE : Keep CLKDCOLDO o/p powered on even when it is gated<br>0x1 = AUTO_PWDN : Automatically power down CLKDCOLDO o/p when it is gated. |
| 11-10 | Reserved                 | R    | 0h    |  |
| 9     | ST_DPLL_CLKDCOLDO        | R    | 0h    | DPLL CLKDCOLDO status<br>0x0 = CLK_ENABLED : The clock output is enabled<br>0x1 = CLK_GATED : The clock output is gated  |
| 8     | DPLL_CLKDCOLDO_GATE_CTRL | R/W  | 0h    | Control gating of DPLL CLKDCOLDO<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request       |
| 7-0   | Reserved                 | R    | 0h    |  |

### 8.1.12.2.33 CM\_DIV\_M4\_DPLL\_CORE Register (offset = 80h) [reset = 4h]

CM\_DIV\_M4\_DPLL\_CORE is shown in [Figure 8-115](#) and described in [Table 8-122](#).

This register provides controls over the CLKOUT1 o/p of the HSDIVIDER.

**Figure 8-115. CM\_DIV\_M4\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-122. CM\_DIV\_M4\_DPLL\_CORE Register Field Descriptions**

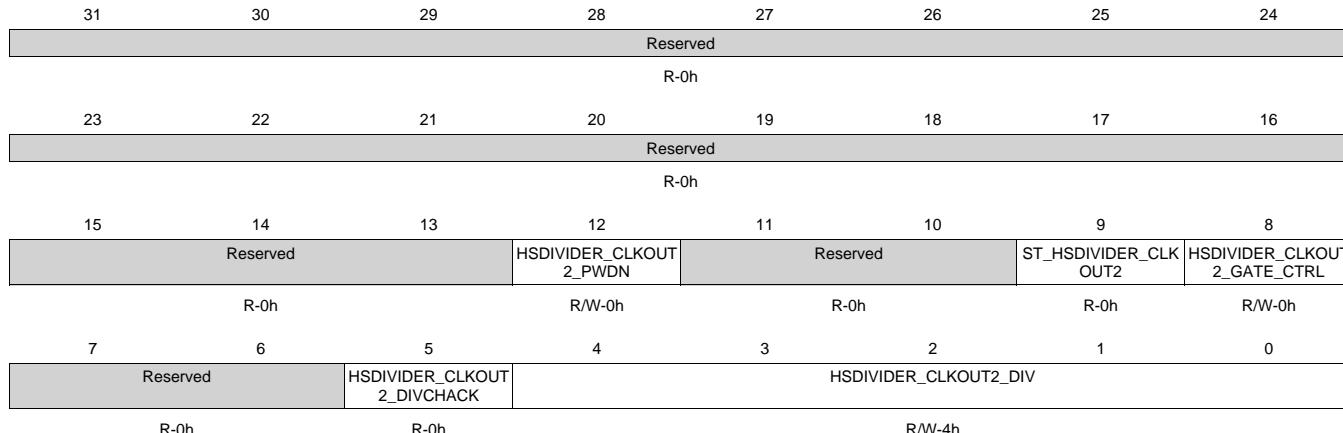
| Bit   | Field                       | Type | Reset | Description   |
|-------|-----------------------------|------|-------|---|
| 31-13 | Reserved                    | R    | 0h    |   |
| 12    | HSDIVIDER_CLKOUT1_PWDN      | R/W  | 0h    | Automatic power down for HSDIVIDER M4 divider and hence CLKOUT1 output when the o/p clock is gated.<br>0x0 = ALWAYS_ACTIVE : Keep M4 divider powered on even when CLKOUT1 is gated.<br>0x1 = AUTO_PWDN : Automatically power down M4 divider when CLKOUT1 is gated. |
| 11-10 | Reserved                    | R    | 0h    |   |
| 9     | ST_HSDIVIDER_CLKOUT1        | R    | 0h    | HSDIVIDER CLKOUT1 status<br>0x0 = CLK_ENABLED : The clock output is enabled<br>0x1 = CLK_GATED : The clock output is gated  |
| 8     | HSDIVIDER_CLKOUT1_GATE_CTRL | R/W  | 0h    | Control gating of HSDIVIDER CLKOUT1<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request   |
| 7-6   | Reserved                    | R    | 0h    |   |
| 5     | HSDIVIDER_CLKOUT1_DIVCHACK  | R    | 0h    | Toggle on this status bit after changing HSDIVIDER_CLKOUT1_DIV indicates that the change in divider value has taken effect  |
| 4-0   | HSDIVIDER_CLKOUT1_DIV       | R/W  | 4h    | DPLL post-divider factor, M4, for internal clock generation.<br>Divide values from 1 to 31.   |

### 8.1.12.2.34 CM\_DIV\_M5\_DPLL\_CORE Register (offset = 84h) [reset = 4h]

CM\_DIV\_M5\_DPLL\_CORE is shown in [Figure 8-116](#) and described in [Table 8-123](#).

This register provides controls over the CLKOUT2 o/p of the HSDIVIDER.

**Figure 8-116. CM\_DIV\_M5\_DPLL\_CORE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-123. CM\_DIV\_M5\_DPLL\_CORE Register Field Descriptions**

| Bit   | Field                       | Type | Reset | Description   |
|-------|-----------------------------|------|-------|---|
| 31-13 | Reserved                    | R    | 0h    |   |
| 12    | HSDIVIDER_CLKOUT2_PWDN      | R/W  | 0h    | Automatic power down for HSDIVIDER M5 divider and hence CLKOUT2 output when the o/p clock is gated.<br>0x0 = ALWAYS_ACTIVE : Keep M5 divider powered on even when CLKOUT2 is gated.<br>0x1 = AUTO_PWDN : Automatically power down M5 divider when CLKOUT2 is gated. |
| 11-10 | Reserved                    | R    | 0h    |   |
| 9     | ST_HSDIVIDER_CLKOUT2        | R    | 0h    | HSDIVIDER CLKOUT2 status<br>0x0 = CLK_ENABLED : The clock output is enabled<br>0x1 = CLK_GATED : The clock output is gated  |
| 8     | HSDIVIDER_CLKOUT2_GATE_CTRL | R/W  | 0h    | Control gating of HSDIVIDER CLKOUT2<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request   |
| 7-6   | Reserved                    | R    | 0h    |   |
| 5     | HSDIVIDER_CLKOUT2_DIVCHACK  | R    | 0h    | Toggle on this status bit after changing HSDIVIDER_CLKOUT2_DIV indicates that the change in divider value has taken effect  |
| 4-0   | HSDIVIDER_CLKOUT2_DIV       | R/W  | 4h    | DPLL post-divider factor, M5, for internal clock generation.<br>Divide values from 1 to 31.   |

### 8.1.12.2.35 CM\_CLKMODE\_DPLL\_MPU Register (offset = 88h) [reset = 4h]

CM\_CLKMODE\_DPLL\_MPU is shown in [Figure 8-117](#) and described in [Table 8-124](#).

This register allows controlling the DPLL modes.

**Figure 8-117. CM\_CLKMODE\_DPLL\_MPU Register**

| 31             | 30                   | 29              | 28          | 27            | 26             | 25                    | 24                 |
|----------------|----------------------|-----------------|-------------|---------------|----------------|-----------------------|--------------------|
| Reserved       |                      |                 |             |               |                |                       |                    |
| R-0h           |                      |                 |             |               |                |                       |                    |
| 23             | 22                   | 21              | 20          | 19            | 18             | 17                    | 16                 |
| Reserved       |                      |                 |             |               |                |                       |                    |
| R-0h           |                      |                 |             |               |                |                       |                    |
| 15             | 14                   | 13              | 12          | 11            | 10             | 9                     | 8                  |
| DPLL_SSC_TYPE  | DPLL_SSC_DOWNS_PREAD | DPLL_SSC_ACK    | DPLL_SSC_EN | DPLL_REGM4XEN | DPLL_LPMODE_EN | DPLL_RELLOCK_RAM_P_EN | DPLL_DRIFTGUARD_EN |
| R/W-0h         | R/W-0h               | R-0h            | R/W-0h      | R-0h          | R/W-0h         | R/W-0h                | R/W-0h             |
| 7              | 6                    | 5               | 4           | 3             | 2              | 1                     | 0                  |
| DPLL_RAMP_RATE |                      | DPLL_RAMP_LEVEL |             |               | DPLL_EN        |                       |                    |
| R/W-0h         |                      | R/W-0h          |             |               | R/W-4h         |                       |                    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-124. CM\_CLKMODE\_DPLL\_MPU Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description  |
|-------|-----------------------|------|-------|--|
| 31-16 | Reserved              | R    | 0h    |  |
| 15    | DPLL_SSC_TYPE         | R/W  | 0h    | Select Triangular Spread Spectrum clocking.<br>Always write 0.<br>0 = Triangular Spread Spectrum Clocking is selected.<br>1 = Reserved.  |
| 14    | DPLL_SSC_DOWNSPREAD   | R/W  | 0h    | Control if only low frequency spread is required<br>0x0 = FULL_SPREAD : When SSC is enabled, clock frequency is spread on both sides of the programmed frequency<br>0x1 = LOW_SPREAD : When SSC is enabled, clock frequency is spread only on the lower side of the programmed frequency |
| 13    | DPLL_SSC_ACK          | R    | 0h    | Acknowledgement from the DPLL regarding start and stop of Spread Spectrum Clocking feature<br>0x0 = Disabled : SSC has been turned off on PLL o/p<br>0x1 = Enabled : SSC has been turned on on PLL o/p   |
| 12    | DPLL_SSC_EN           | R/W  | 0h    | Enable or disable Spread Spectrum Clocking<br>0x0 = Disabled : SSC disabled<br>0x1 = Enabled : SSC enabled   |
| 11    | DPLL_REGM4XEN         | R    | 0h    | Enable the REGM4XEN mode of the DPLL.<br>Please check the DPLL documentation to check when this mode can be enabled.   |
| 10    | DPLL_LPMODE_EN        | R/W  | 0h    | Set the DPLL in Low Power mode.<br>Check the DPLL documentation to see when this can be enabled.<br>0x0 = DISABLED : Low power mode of the DPLL is disabled<br>0x1 = ENABLED : Low power mode of the DPLL is enabled   |
| 9     | DPLL_RELLOCK_RAM_P_EN | R/W  | 0h    | If enabled, the clock ramping feature is used applied during the lock process, as well as the relock process.<br>If disabled, the clock ramping feature is used only during the first lock.  |

**Table 8-124. CM\_CLKMODE\_DPLL\_MPU Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 8   | DPLL_DRIFTGUARD_EN | R/W  | 0h    | This bit allows to enable or disable the automatic recalibration feature of the DPLL.<br>The DPLL will automatically start a recalibration process upon assertion of the DPLL's RECAL flag if this bit is set.<br>0x0 = Disabled : DRIFTGUARD feature is disabled<br>0x1 = Enabled : DRIFTGUARD feature is enabled   |
| 7-5 | DPLL_RAMP_RATE     | R/W  | 0h    | Selects the time in terms of DPLL REFCLKs spent at each stage of the clock ramping process<br>0x0 = REFCLKx2 : 2 REFCLKs<br>0x1 = REFCLKx4 : 4 REFCLKs<br>0x2 = REFCLKx8 : 8 REFCLKs<br>0x3 = REFCLKx16 : 16 REFCLKs<br>0x4 = REFCLKx32 : 32 REFCLKs<br>0x5 = REFCLKx64 : 64 REFCLKs<br>0x6 = REFCLKx128 : 128 REFCLKs<br>0x7 = REFCLKx512 : 512 REFCLKs   |
| 4-3 | DPLL_RAMP_LEVEL    | R/W  | 0h    | The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and re-lock. The frequency ramping will happen in a maximum of 4 steps in frequency before the DPLL's frequency lock indicator is asserted. This register is used to enable/disable the DPLL ramping feature. If enabled, it is also used to select the algorithm used for clock ramping<br>0x0 = RAMP_DISABLE : CLKOUT => No ramping CLKOUTX2 => No ramping<br>0x1 = RAMP_ALGO1 : CLKOUT => Bypass clk -> Fout/8 -> Fout/4 -> Fout/2 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/8 -> Foutx2/4 -> Foutx2/2 -> Foutx2<br>0x2 = RAMP_ALGO2 : CLKOUT => Bypass clk -> Fout/4 -> Fout/2 -> Fout/1.5 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/4 -> Foutx2/2 -> Foutx2/1.5 -> Foutx2<br>0x3 = Reserved : Reserved |
| 2-0 | DPLL_EN            | R/W  | 4h    | DPLL control.<br>Upon Warm Reset, the PRCM DPLL control state machine updates this register to reflect MN Bypass mode.<br>0x0 = Reserved : Reserved<br>0x1 = Reserved1 : Reserved<br>0x2 = Reserved2 : Reserved<br>0x3 = Reserved3 : Reserved<br>0x4 = DPLL_MN_BYP_MODE : Put the DPLL in MN Bypass mode. The DPLL_MULT register bits are reset to 0 automatically by putting the DPLL in this mode.<br>0x5 = DPLL_LP_BYP_MODE : Put the DPLL in Idle Bypass Low Power mode.<br>0x6 = DPLL_FR_BYP_MODE : Put the DPLL in Idle Bypass Fast Relock mode.<br>0x7 = DPLL_LOCK_MODE : Enables the DPLL in Lock mode   |

### 8.1.12.2.36 CM\_CLKMODE\_DPLL\_PER Register (offset = 8Ch) [reset = 4h]

CM\_CLKMODE\_DPLL\_PER is shown in [Figure 8-118](#) and described in [Table 8-125](#).

This register allows controlling the DPLL modes.

**Figure 8-118. CM\_CLKMODE\_DPLL\_PER Register**

|               |                      |              |             |          |         |    |    |
|---------------|----------------------|--------------|-------------|----------|---------|----|----|
| 31            | 30                   | 29           | 28          | 27       | 26      | 25 | 24 |
| Reserved      |                      |              |             |          |         |    |    |
| R-0h          |                      |              |             |          |         |    |    |
| 23            | 22                   | 21           | 20          | 19       | 18      | 17 | 16 |
| Reserved      |                      |              |             |          |         |    |    |
| R-0h          |                      |              |             |          |         |    |    |
| 15            | 14                   | 13           | 12          | 11       | 10      | 9  | 8  |
| DPLL_SSC_TYPE | DPLL_SSC_DOWNS_PREAD | DPLL_SSC_ACK | DPLL_SSC_EN | Reserved |         |    |    |
| R/W-0h        | R/W-0h               | R-0h         | R/W-0h      | R-0h     |         |    |    |
| 7             | 6                    | 5            | 4           | 3        | 2       | 1  | 0  |
| Reserved      |                      |              |             |          | DPLL_EN |    |    |
| R-0h          |                      |              |             |          | R/W-4h  |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-125. CM\_CLKMODE\_DPLL\_PER Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-16 | Reserved            | R    | 0h    |   |
| 15    | DPLL_SSC_TYPE       | R/W  | 0h    | Select Triangular Spread Spectrum clocking.<br>Always write 0.<br>0 = Triangular Spread Spectrum Clocking is selected.<br>1 = Reserved.   |
| 14    | DPLL_SSC_DOWNSPREAD | R/W  | 0h    | Control if only low frequency spread is required<br>0x0 = FULL_SPREAD : When SSC is enabled, clock frequency is spread on both sides of the programmed frequency<br>0x1 = LOW_SPREAD : When SSC is enabled, clock frequency is spread only on the lower side of the programmed frequency  |
| 13    | DPLL_SSC_ACK        | R    | 0h    | Acknowledgement from the DPLL regarding start and stop of Spread Spectrum Clocking feature<br>0x0 = Disabled : SSC has been turned off on PLL o/p/s<br>0x1 = Enabled : SSC has been turned on on PLL o/p/s  |
| 12    | DPLL_SSC_EN         | R/W  | 0h    | Enable or disable Spread Spectrum Clocking<br>0x0 = Disabled : SSC disabled<br>0x1 = Enabled : SSC enabled  |
| 11-3  | Reserved            | R    | 0h    |   |
| 2-0   | DPLL_EN             | R/W  | 4h    | DPLL control.<br>Upon Warm Reset, the PRCM DPLL control state machine updates this register to reflect DPLL Low Power Stop mode.<br>0x0 = Reserved : Reserved<br>0x1 = DPLL_LP_STP_MODE : Put the DPLL in Low Power Stop mode<br>0x2 = Reserved2 : Reserved2<br>0x3 = Reserved3 : Reserved<br>0x4 = DPLL_MN_BYP_MODE : Put the DPLL in MN Bypass mode. The DPLL_MULT register bits are reset to 0 automatically by putting the DPLL in this mode.<br>0x5 = DPLL_LP_BYP_MODE : Put the DPLL in Idle Bypass Low Power mode.<br>0x6 = Reserved6 : Reserved<br>0x7 = DPLL_LOCK_MODE : Enables the DPLL in Lock mode |

### 8.1.12.2.37 CM\_CLKMODE\_DPLL\_CORE Register (offset = 90h) [reset = 4h]

CM\_CLKMODE\_DPLL\_CORE is shown in [Figure 8-119](#) and described in [Table 8-126](#).

This register allows controlling the DPLL modes.

**Figure 8-119. CM\_CLKMODE\_DPLL\_CORE Register**

|                |                     |              |                 |               |                |                       |                    |
|----------------|---------------------|--------------|-----------------|---------------|----------------|-----------------------|--------------------|
| 31             | 30                  | 29           | 28              | 27            | 26             | 25                    | 24                 |
| Reserved       |                     |              |                 |               |                |                       |                    |
| R-0h           |                     |              |                 |               |                |                       |                    |
| 23             | 22                  | 21           | 20              | 19            | 18             | 17                    | 16                 |
| Reserved       |                     |              |                 |               |                |                       |                    |
| R-0h           |                     |              |                 |               |                |                       |                    |
| 15             | 14                  | 13           | 12              | 11            | 10             | 9                     | 8                  |
| DPLL_SSC_TYPE  | DPLL_SSC_DOWNSPREAD | DPLL_SSC_ACK | DPLL_SSC_EN     | DPLL_REGM4XEN | DPLL_LPMODE_EN | DPLL_RELLOCK_RAM_P_EN | DPLL_DRIFTGUARD_EN |
| R/W-0h         | R/W-0h              | R-0h         | R/W-0h          | R-0h          | R/W-0h         | R/W-0h                | R/W-0h             |
| 7              | 6                   | 5            | 4               | 3             | 2              | 1                     | 0                  |
| DPLL_RAMP_RATE |                     |              | DPLL_RAMP_LEVEL |               |                | DPLL_EN               |                    |
| R/W-0h         |                     |              | R/W-0h          |               |                | R/W-4h                |                    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-126. CM\_CLKMODE\_DPLL\_CORE Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description  |
|-------|-----------------------|------|-------|--|
| 31-16 | Reserved              | R    | 0h    |  |
| 15    | DPLL_SSC_TYPE         | R/W  | 0h    | Select Triangular Spread Spectrum clocking.<br>Always write 0.<br>0 = Triangular Spread Spectrum Clocking is selected.<br>1 = Reserved.  |
| 14    | DPLL_SSC_DOWNSPREAD   | R/W  | 0h    | Control if only low frequency spread is required<br>0x0 = FULL_SPREAD : When SSC is enabled, clock frequency is spread on both sides of the programmed frequency<br>0x1 = LOW_SPREAD : When SSC is enabled, clock frequency is spread only on the lower side of the programmed frequency |
| 13    | DPLL_SSC_ACK          | R    | 0h    | Acknowledgement from the DPLL regarding start and stop of Spread Spectrum Clocking feature<br>0x0 = Disabled : SSC has been turned off on PLL o/p<br>0x1 = Enabled : SSC has been turned on on PLL o/p   |
| 12    | DPLL_SSC_EN           | R/W  | 0h    | Enable or disable Spread Spectrum Clocking<br>0x0 = Disabled : SSC disabled<br>0x1 = Enabled : SSC enabled   |
| 11    | DPLL_REGM4XEN         | R    | 0h    | Enable the REGM4XEN mode of the DPLL.<br>Please check the DPLL documentation to check when this mode can be enabled.   |
| 10    | DPLL_LPMODE_EN        | R/W  | 0h    | Set the DPLL in Low Power mode.<br>Check the DPLL documentation to see when this can be enabled.<br>0x0 = DISABLED : Low power mode of the DPLL is disabled<br>0x1 = ENABLED : Low power mode of the DPLL is enabled   |
| 9     | DPLL_RELLOCK_RAM_P_EN | R/W  | 0h    | If enabled, the clock ramping feature is used applied during the lock process, as well as the relock process.<br>If disabled, the clock ramping feature is used only during the first lock.  |

**Table 8-126. CM\_CLKMODE\_DPLL\_CORE Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 8   | DPLL_DRIFTGUARD_EN | R/W  | 0h    | This bit allows to enable or disable the automatic recalibration feature of the DPLL.<br>The DPLL will automatically start a recalibration process upon assertion of the DPLL's RECAL flag if this bit is set.<br>0x0 = Disabled : DRIFTGUARD feature is disabled<br>0x1 = Enabled : DRIFTGUARD feature is enabled   |
| 7-5 | DPLL_RAMP_RATE     | R/W  | 0h    | Selects the time in terms of DPLL REFCLKs spent at each stage of the clock ramping process<br>0x0 = REFCLKx2 : 2 REFCLKs<br>0x1 = REFCLKx4 : 4 REFCLKs<br>0x2 = REFCLKx8 : 8 REFCLKs<br>0x3 = REFCLKx16 : 16 REFCLKs<br>0x4 = REFCLKx32 : 32 REFCLKs<br>0x5 = REFCLKx64 : 64 REFCLKs<br>0x6 = REFCLKx128 : 128 REFCLKs<br>0x7 = REFCLKx512 : 512 REFCLKs   |
| 4-3 | DPLL_RAMP_LEVEL    | R/W  | 0h    | The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and re-lock. The frequency ramping will happen in a maximum of 4 steps in frequency before the DPLL's frequency lock indicator is asserted. This register is used to enable/disable the DPLL ramping feature. If enabled, it is also used to select the algorithm used for clock ramping<br>0x0 = RAMP_DISABLE : CLKOUT => No ramping CLKOUTX2 => No ramping<br>0x1 = RAMP_ALGO1 : CLKOUT => Bypass clk -> Fout/8 -> Fout/4 -> Fout/2 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/8 -> Foutx2/4 -> Foutx2/2 -> Foutx2<br>0x2 = RAMP_ALGO2 : CLKOUT => Bypass clk -> Fout/4 -> Fout/2 -> Fout/1.5 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/4 -> Foutx2/2 -> Foutx2/1.5 -> Foutx2<br>0x3 = Reserved : Reserved |
| 2-0 | DPLL_EN            | R/W  | 4h    | DPLL control.<br>Upon Warm Reset, the PRCM DPLL control state machine updates this register to reflect MN Bypass mode.<br>0x0 = Reserved : Reserved<br>0x1 = Reserved1 : Reserved<br>0x2 = Reserved2 : Reserved<br>0x3 = Reserved3 : Reserved<br>0x4 = DPLL_MN_BYP_MODE : Put the DPLL in MN Bypass mode. The DPLL_MULT register bits are reset to 0 automatically by putting the DPLL in this mode.<br>0x5 = DPLL_LP_BYP_MODE : Put the DPLL in Idle Bypass Low Power mode.<br>0x6 = DPLL_FR_BYP_MODE : Put the DPLL in Idle Bypass Fast Relock mode.<br>0x7 = DPLL_LOCK_MODE : Enables the DPLL in Lock mode   |

### 8.1.12.2.38 CM\_CLKMODE\_DPLL\_DDR Register (offset = 94h) [reset = 4h]

CM\_CLKMODE\_DPLL\_DDR is shown in [Figure 8-120](#) and described in [Table 8-127](#).

This register allows controlling the DPLL modes.

**Figure 8-120. CM\_CLKMODE\_DPLL\_DDR Register**

|                |                      |              |                 |               |                |                       |                    |
|----------------|----------------------|--------------|-----------------|---------------|----------------|-----------------------|--------------------|
| 31             | 30                   | 29           | 28              | 27            | 26             | 25                    | 24                 |
| Reserved       |                      |              |                 |               |                |                       |                    |
| R-0h           |                      |              |                 |               |                |                       |                    |
| 23             | 22                   | 21           | 20              | 19            | 18             | 17                    | 16                 |
| Reserved       |                      |              |                 |               |                |                       |                    |
| R-0h           |                      |              |                 |               |                |                       |                    |
| 15             | 14                   | 13           | 12              | 11            | 10             | 9                     | 8                  |
| DPLL_SSC_TYPE  | DPLL_SSC_DOWNS_PREAD | DPLL_SSC_ACK | DPLL_SSC_EN     | DPLL_REGM4XEN | DPLL_LPMODE_EN | DPLL_RELLOCK_RAM_P_EN | DPLL_DRIFTGUARD_EN |
| R/W-0h         | R/W-0h               | R-0h         | R/W-0h          | R-0h          | R/W-0h         | R/W-0h                | R/W-0h             |
| 7              | 6                    | 5            | 4               | 3             | 2              | 1                     | 0                  |
| DPLL_RAMP_RATE |                      |              | DPLL_RAMP_LEVEL |               |                | DPLL_EN               |                    |
| R/W-0h         |                      |              | R/W-0h          |               |                | R/W-4h                |                    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-127. CM\_CLKMODE\_DPLL\_DDR Register Field Descriptions**

| Bit   | Field                  | Type | Reset | Description  |
|-------|------------------------|------|-------|--|
| 31-16 | Reserved               | R    | 0h    |  |
| 15    | DPLL_SSC_TYPE          | R/W  | 0h    | Select Triangular Spread Spectrum clocking.<br>Always write 0.<br>0 = Triangular Spread Spectrum Clocking is selected.<br>1 = Reserved.  |
| 14    | DPLL_SSC_DOWNSPRE AD   | R/W  | 0h    | Control if only low frequency spread is required<br>0x0 = FULL_SPREAD : When SSC is enabled, clock frequency is spread on both sides of the programmed frequency<br>0x1 = LOW_SPREAD : When SSC is enabled, clock frequency is spread only on the lower side of the programmed frequency |
| 13    | DPLL_SSC_ACK           | R    | 0h    | Acknowledgement from the DPLL regarding start and stop of Spread Spectrum Clocking feature<br>0x0 = Disabled : SSC has been turned off on PLL o/p/s<br>0x1 = Enabled : SSC has been turned on on PLL o/p/s   |
| 12    | DPLL_SSC_EN            | R/W  | 0h    | Enable or disable Spread Spectrum Clocking<br>0x0 = Disabled : SSC disabled<br>0x1 = Enabled : SSC enabled   |
| 11    | DPLL_REGM4XEN          | R    | 0h    | Enable the REGM4XEN mode of the DPLL.<br>Please check the DPLL documentation to check when this mode can be enabled.   |
| 10    | DPLL_LPMODE_EN         | R/W  | 0h    | Set the DPLL in Low Power mode.<br>Check the DPLL documentation to see when this can be enabled.<br>0x0 = DISABLED : Low power mode of the DPLL is disabled<br>0x1 = ENABLED : Low power mode of the DPLL is enabled   |
| 9     | DPLL_RELLOCK_RAM_P_E N | R/W  | 0h    | If enabled, the clock ramping feature is used applied during the lock process, as well as the relock process.<br>If disabled, the clock ramping feature is used only during the first lock.  |

**Table 8-127. CM\_CLKMODE\_DPLL\_DDR Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 8   | DPLL_DRIFTGUARD_EN | R/W  | 0h    | This bit allows to enable or disable the automatic recalibration feature of the DPLL.<br>The DPLL will automatically start a recalibration process upon assertion of the DPLL's RECAL flag if this bit is set.<br>0x0 = Disabled : DRIFTGUARD feature is disabled<br>0x1 = Enabled : DRIFTGUARD feature is enabled   |
| 7-5 | DPLL_RAMP_RATE     | R/W  | 0h    | Selects the time in terms of DPLL REFCLKs spent at each stage of the clock ramping process<br>0x0 = REFCLKx2 : 2 REFCLKs<br>0x1 = REFCLKx4 : 4 REFCLKs<br>0x2 = REFCLKx8 : 8 REFCLKs<br>0x3 = REFCLKx16 : 16 REFCLKs<br>0x4 = REFCLKx32 : 32 REFCLKs<br>0x5 = REFCLKx64 : 64 REFCLKs<br>0x6 = REFCLKx128 : 128 REFCLKs<br>0x7 = REFCLKx512 : 512 REFCLKs   |
| 4-3 | DPLL_RAMP_LEVEL    | R/W  | 0h    | The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and re-lock. The frequency ramping will happen in a maximum of 4 steps in frequency before the DPLL's frequency lock indicator is asserted. This register is used to enable/disable the DPLL ramping feature. If enabled, it is also used to select the algorithm used for clock ramping<br>0x0 = RAMP_DISABLE : CLKOUT => No ramping CLKOUTX2 => No ramping<br>0x1 = RAMP_ALGO1 : CLKOUT => Bypass clk -> Fout/8 -> Fout/4 -> Fout/2 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/8 -> Foutx2/4 -> Foutx2/2 -> Foutx2<br>0x2 = RAMP_ALGO2 : CLKOUT => Bypass clk -> Fout/4 -> Fout/2 -> Fout/1.5 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/4 -> Foutx2/2 -> Foutx2/1.5 -> Foutx2<br>0x3 = Reserved : Reserved |
| 2-0 | DPLL_EN            | R/W  | 4h    | DPLL control.<br>Upon Warm Reset, the PRCM DPLL control state machine updates this register to reflect MN Bypass mode.<br>0x0 = Reserved : Reserved<br>0x1 = Reserved1 : Reserved<br>0x2 = Reserved2 : Reserved<br>0x3 = Reserved3 : Reserved<br>0x4 = DPLL_MN_BYP_MODE : Put the DPLL in MN Bypass mode. The DPLL_MULT register bits are reset to 0 automatically by putting the DPLL in this mode.<br>0x5 = DPLL_LP_BYP_MODE : Put the DPLL in Idle Bypass Low Power mode.<br>0x6 = DPLL_FR_BYP_MODE : Put the DPLL in Idle Bypass Fast Relock mode.<br>0x7 = DPLL_LOCK_MODE : Enables the DPLL in Lock mode   |

### 8.1.12.2.39 CM\_CLKMODE\_DPLL\_DISP Register (offset = 98h) [reset = 4h]

CM\_CLKMODE\_DPLL\_DISP is shown in [Figure 8-121](#) and described in [Table 8-128](#).

This register allows controlling the DPLL modes.

**Figure 8-121. CM\_CLKMODE\_DPLL\_DISP Register**

|                |                      |              |                 |               |                |                      |                    |
|----------------|----------------------|--------------|-----------------|---------------|----------------|----------------------|--------------------|
| 31             | 30                   | 29           | 28              | 27            | 26             | 25                   | 24                 |
| Reserved       |                      |              |                 |               |                |                      |                    |
| R-0h           |                      |              |                 |               |                |                      |                    |
| 23             | 22                   | 21           | 20              | 19            | 18             | 17                   | 16                 |
| Reserved       |                      |              |                 |               |                |                      |                    |
| R-0h           |                      |              |                 |               |                |                      |                    |
| 15             | 14                   | 13           | 12              | 11            | 10             | 9                    | 8                  |
| DPLL_SSC_TYPE  | DPLL_SSC_DOWNS_PREAD | DPLL_SSC_ACK | DPLL_SSC_EN     | DPLL_REGM4XEN | DPLL_LPMODE_EN | DPLL_RELOCK_RAM_P_EN | DPLL_DRIFTGUARD_EN |
| R/W-0h         | R/W-0h               | R-0h         | R/W-0h          | R-0h          | R/W-0h         | R/W-0h               | R/W-0h             |
| 7              | 6                    | 5            | 4               | 3             | 2              | 1                    | 0                  |
| DPLL_RAMP_RATE |                      |              | DPLL_RAMP_LEVEL |               |                | DPLL_EN              |                    |
| R/W-0h         |                      |              | R/W-0h          |               |                | R/W-4h               |                    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-128. CM\_CLKMODE\_DPLL\_DISP Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31-16 | Reserved             | R    | 0h    |  |
| 15    | DPLL_SSC_TYPE        | R/W  | 0h    | Select Triangular Spread Spectrum clocking.<br>Always write 0.<br>0 = Triangular Spread Spectrum Clocking is selected.<br>1 = Reserved.  |
| 14    | DPLL_SSC_DOWNSPRE AD | R/W  | 0h    | Control if only low frequency spread is required<br>0x0 = FULL_SPREAD : When SSC is enabled, clock frequency is spread on both sides of the programmed frequency<br>0x1 = LOW_SPREAD : When SSC is enabled, clock frequency is spread only on the lower side of the programmed frequency |
| 13    | DPLL_SSC_ACK         | R    | 0h    | Acknowledgement from the DPLL regarding start and stop of Spread Spectrum Clocking feature<br>0x0 = Disabled : SSC has been turned off on PLL o/p/s<br>0x1 = Enabled : SSC has been turned on on PLL o/p/s   |
| 12    | DPLL_SSC_EN          | R/W  | 0h    | Enable or disable Spread Spectrum Clocking<br>0x0 = Disabled : SSC disabled<br>0x1 = Enabled : SSC enabled   |
| 11    | DPLL_REGM4XEN        | R    | 0h    | Enable the REGM4XEN mode of the DPLL.<br>Please check the DPLL documentation to check when this mode can be enabled.   |
| 10    | DPLL_LPMODE_EN       | R/W  | 0h    | Set the DPLL in Low Power mode.<br>Check the DPLL documentation to see when this can be enabled.<br>0x0 = DISABLED : Low power mode of the DPLL is disabled<br>0x1 = ENABLED : Low power mode of the DPLL is enabled   |
| 9     | DPLL_RELOCK_RAMP_E N | R/W  | 0h    | If enabled, the clock ramping feature is used applied during the lock process, as well as the relock process.<br>If disabled, the clock ramping feature is used only during the first lock.  |

**Table 8-128. CM\_CLKMODE\_DPLL\_DISP Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 8   | DPLL_DRIFTGUARD_EN | R/W  | 0h    | This bit allows to enable or disable the automatic recalibration feature of the DPLL.<br>The DPLL will automatically start a recalibration process upon assertion of the DPLL's RECAL flag if this bit is set.<br>0x0 = Disabled : DRIFTGUARD feature is disabled<br>0x1 = Enabled : DRIFTGUARD feature is enabled   |
| 7-5 | DPLL_RAMP_RATE     | R/W  | 0h    | Selects the time in terms of DPLL REFCLKs spent at each stage of the clock ramping process<br>0x0 = REFCLKx2 : 2 REFCLKs<br>0x1 = REFCLKx4 : 4 REFCLKs<br>0x2 = REFCLKx8 : 8 REFCLKs<br>0x3 = REFCLKx16 : 16 REFCLKs<br>0x4 = REFCLKx32 : 32 REFCLKs<br>0x5 = REFCLKx64 : 64 REFCLKs<br>0x6 = REFCLKx128 : 128 REFCLKs<br>0x7 = REFCLKx512 : 512 REFCLKs   |
| 4-3 | DPLL_RAMP_LEVEL    | R/W  | 0h    | The DPLL provides an output clock frequency ramping feature when switching from bypass clock to normal clock during lock and re-lock. The frequency ramping will happen in a maximum of 4 steps in frequency before the DPLL's frequency lock indicator is asserted. This register is used to enable/disable the DPLL ramping feature. If enabled, it is also used to select the algorithm used for clock ramping<br>0x0 = RAMP_DISABLE : CLKOUT => No ramping CLKOUTX2 => No ramping<br>0x1 = RAMP_ALGO1 : CLKOUT => Bypass clk -> Fout/8 -> Fout/4 -> Fout/2 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/8 -> Foutx2/4 -> Foutx2/2 -> Foutx2<br>0x2 = RAMP_ALGO2 : CLKOUT => Bypass clk -> Fout/4 -> Fout/2 -> Fout/1.5 -> Fout CLKOUTX2 => Bypass clk -> Foutx2/4 -> Foutx2/2 -> Foutx2/1.5 -> Foutx2<br>0x3 = Reserved : Reserved |
| 2-0 | DPLL_EN            | R/W  | 4h    | DPLL control.<br>Upon Warm Reset, the PRCM DPLL control state machine updates this register to reflect MN Bypass mode.<br>0x0 = Reserved : Reserved<br>0x1 = Reserved1 : Reserved<br>0x2 = Reserved2 : Reserved<br>0x3 = Reserved3 : Reserved<br>0x4 = DPLL_MN_BYP_MODE : Put the DPLL in MN Bypass mode. The DPLL_MULT register bits are reset to 0 automatically by putting the DPLL in this mode.<br>0x5 = DPLL_LP_BYP_MODE : Put the DPLL in Idle Bypass Low Power mode.<br>0x6 = DPLL_FR_BYP_MODE : Put the DPLL in Idle Bypass Fast Relock mode.<br>0x7 = DPLL_LOCK_MODE : Enables the DPLL in Lock mode   |

#### **8.1.12.2.40 CM\_CLKSEL\_DPLL\_PERIPH Register (offset = 9Ch) [reset = 0h]**

CM\_CLKSEL\_DPLL\_PERIPH is shown in [Figure 8-122](#) and described in [Table 8-129](#).

This register provides controls over the DPLL.

**Figure 8-122. CM\_CLKSEL\_DPLL\_PERIPH Register**

| 31          | 30 | 29       | 28 | 27 | 26        | 25 | 24 |
|-------------|----|----------|----|----|-----------|----|----|
| DPLL_SD_DIV |    |          |    |    |           |    |    |
| R/W-0h      |    |          |    |    |           |    |    |
| 23          | 22 | 21       | 20 | 19 | 18        | 17 | 16 |
| Reserved    |    | Reserved |    |    | DPLL_MULT |    |    |
| R-0h        |    | R-0h     |    |    | R/W-0h    |    |    |
| 15          | 14 | 13       | 12 | 11 | 10        | 9  | 8  |
| DPLL_MULT   |    |          |    |    |           |    |    |
| R/W-0h      |    |          |    |    |           |    |    |
| 7           | 6  | 5        | 4  | 3  | 2         | 1  | 0  |
| DPLL_DIV    |    |          |    |    |           |    |    |
| R/W-0h      |    |          |    |    |           |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-129. CM\_CLKSEL\_DPLL\_PERIPH Register Field Descriptions**

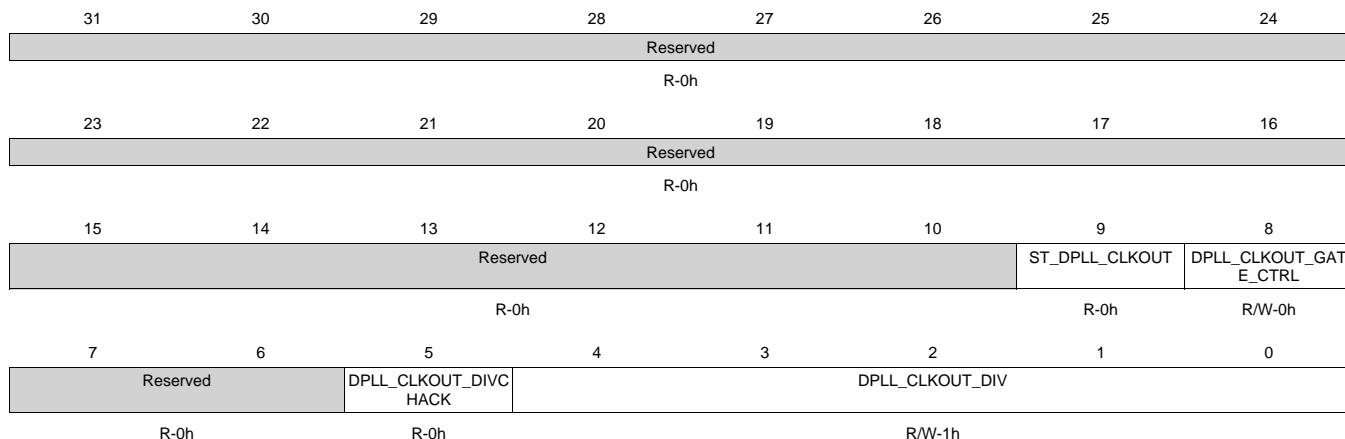
| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-24 | DPLL_SD_DIV | R/W  | 0h    | Sigma-Delta divider select (2-255).<br>This factor must be set by s/w to ensure optimum jitter performance.<br>$DPLL\_SD\_DIV = CEILING([DPLL\_MULT/(DPLL\_DIV+1)] * CLKINP / 250)$ , where CLKINP is the input clock of the DPLL in MHz.<br>Must be set with M and N factors, and must not be changed once DPLL is locked.<br>0x0 = Reserved : Reserved<br>0x1 = Reserved1 : Reserved |
| 23    | Reserved    | R    | 0h    |  |
| 22-20 | Reserved    | R    | 0h    |  |
| 19-8  | DPLL_MULT   | R/W  | 0h    | DPLL multiplier factor (2 to 4095).<br>This register is automatically cleared to 0 when the DPLL_EN field in the *CLKMODE_DPLL* register is set to select MN Bypass mode.<br>(equal to input M of DPLL)<br>M=2 to 4095 => DPLL multiplies by M).<br>0x0 = 0 : Reserved<br>0x1 = 1 : Reserved   |
| 7-0   | DPLL_DIV    | R/W  | 0h    | DPLL divider factor (0 to 255) (equal to input N of DPLL actual division factor is N+1).   |

#### 8.1.12.2.41 CM\_DIV\_M2\_DPLL\_DDR Register (offset = A0h) [reset = 1h]

CM\_DIV\_M2\_DPLL\_DDR is shown in Figure 8-123 and described in Table 8-130.

This register provides controls over the M2 divider of the DPLL.

**Figure 8-123. CM\_DIV\_M2\_DPLL\_DDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-130. CM\_DIV\_M2\_DPLL\_DDR Register Field Descriptions**

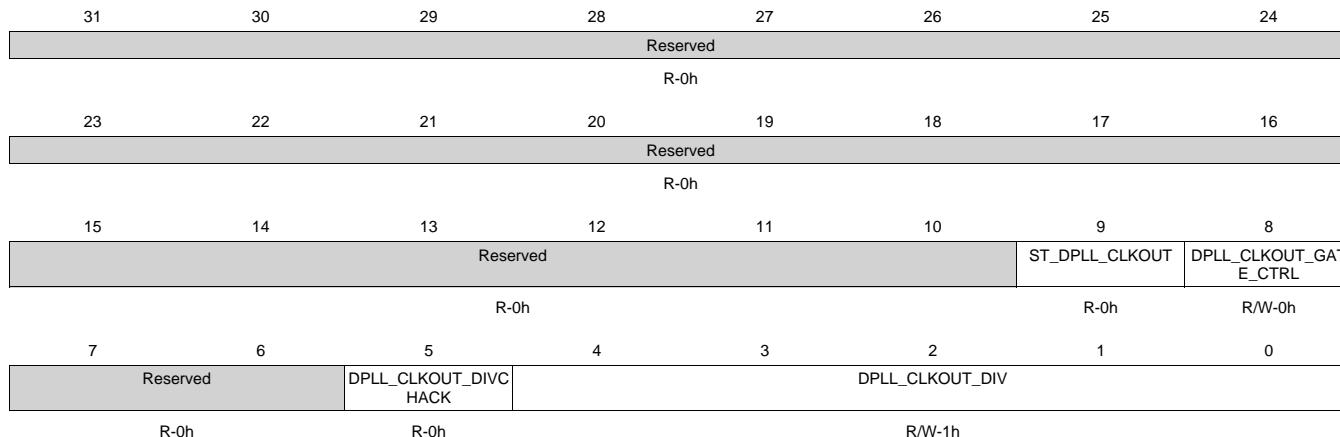
| Bit   | Field                 | Type | Reset | Description   |
|-------|-----------------------|------|-------|---|
| 31-10 | Reserved              | R    | 0h    |   |
| 9     | ST_DPLL_CLKOUT        | R    | 0h    | DPLL CLKOUT status<br>0x0 = CLK_GATED : The clock output is gated<br>0x1 = CLK_ENABLED : The clock output is enabled  |
| 8     | DPLL_CLKOUT_GATE_CTRL | R/W  | 0h    | Control gating of DPLL CLKOUT<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request |
| 7-6   | Reserved              | R    | 0h    |   |
| 5     | DPLL_CLKOUT_DIVCHA CK | R    | 0h    | Toggle on this status bit after changing DPLL_CLKOUT_DIV indicates that the change in divider value has taken effect  |
| 4-0   | DPLL_CLKOUT_DIV       | R/W  | 1h    | DPLL M2 post-divider factor (1 to 31).  |

### 8.1.12.2.42 CM\_DIV\_M2\_DPLL\_DISP Register (offset = A4h) [reset = 1h]

CM\_DIV\_M2\_DPLL\_DISP is shown in [Figure 8-124](#) and described in [Table 8-131](#).

This register provides controls over the M2 divider of the DPLL.

**Figure 8-124. CM\_DIV\_M2\_DPLL\_DISP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-131. CM\_DIV\_M2\_DPLL\_DISP Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description   |
|-------|-----------------------|------|-------|---|
| 31-10 | Reserved              | R    | 0h    |   |
| 9     | ST_DPLL_CLKOUT        | R    | 0h    | DPLL CLKOUT status<br>0x0 = CLK_GATED : The clock output is gated<br>0x1 = CLK_ENABLED : The clock output is enabled  |
| 8     | DPLL_CLKOUT_GATE_CTRL | R/W  | 0h    | Control gating of DPLL CLKOUT<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request |
| 7-6   | Reserved              | R    | 0h    |   |
| 5     | DPLL_CLKOUT_DIVCHA CK | R    | 0h    | Toggle on this status bit after changing DPLL_CLKOUT_DIV indicates that the change in divider value has taken effect  |
| 4-0   | DPLL_CLKOUT_DIV       | R/W  | 1h    | DPLL M2 post-divider factor (1 to 31).  |

### 8.1.12.2.43 CM\_DIV\_M2\_DPLL\_MPRegister (offset = A8h) [reset = 1h]

CM\_DIV\_M2\_DPLL\_MPRegister is shown in Figure 8-125 and described in Table 8-132.

This register provides controls over the M2 divider of the DPLL.

**Figure 8-125. CM\_DIV\_M2\_DPLL\_MPRegister**

|          |                          |                 |    |    |    |    |    |
|----------|--------------------------|-----------------|----|----|----|----|----|
| 31       | 30                       | 29              | 28 | 27 | 26 | 25 | 24 |
| Reserved |                          |                 |    |    |    |    |    |
| R-0h     |                          |                 |    |    |    |    |    |
| 23       | 22                       | 21              | 20 | 19 | 18 | 17 | 16 |
| Reserved |                          |                 |    |    |    |    |    |
| R-0h     |                          |                 |    |    |    |    |    |
| 15       | 14                       | 13              | 12 | 11 | 10 | 9  | 8  |
| Reserved |                          |                 |    |    |    |    |    |
| R-0h     |                          |                 |    |    |    |    |    |
| 7        | 6                        | 5               | 4  | 3  | 2  | 1  | 0  |
| Reserved | DPLL_CLKOUT_DIVC<br>HACK | DPLL_CLKOUT_DIV |    |    |    |    |    |
| R-0h     | R-0h                     | R/W-1h          |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-132. CM\_DIV\_M2\_DPLL\_MPRegister Field Descriptions**

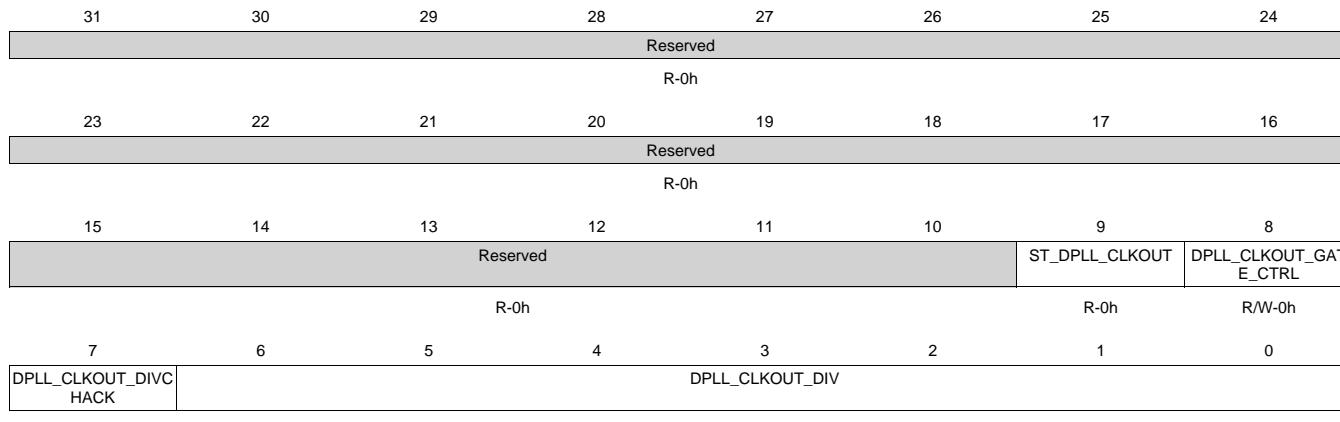
| Bit   | Field                     | Type | Reset | Description   |
|-------|---------------------------|------|-------|---|
| 31-10 | Reserved                  | R    | 0h    |   |
| 9     | ST_DPLL_CLKOUT            | R    | 0h    | DPLL CLKOUT status<br>0x0 = CLK_GATED : The clock output is gated<br>0x1 = CLK_ENABLED : The clock output is enabled  |
| 8     | DPLL_CLKOUT_GATE_C<br>TRL | R/W  | 0h    | Control gating of DPLL CLKOUT<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request |
| 7-6   | Reserved                  | R    | 0h    |   |
| 5     | DPLL_CLKOUT_DIVCHA<br>CK  | R    | 0h    | Toggle on this status bit after changing DPLL_CLKOUT_DIV indicates that the change in divider value has taken effect  |
| 4-0   | DPLL_CLKOUT_DIV           | R/W  | 1h    | DPLL M2 post-divider factor (1 to 31).  |

#### **8.1.12.2.44 CM\_DIV\_M2\_DPLL\_PER Register (offset = ACh) [reset = 1h]**

CM\_DIV\_M2\_DPLL\_PER is shown in [Figure 8-126](#) and described in [Table 8-133](#).

This register provides controls over the M2 divider of the DPLL.

**Figure 8-126. CM\_DIV\_M2\_DPLL\_PER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-133. CM\_DIV\_M2\_DPLL\_PER Register Field Descriptions**

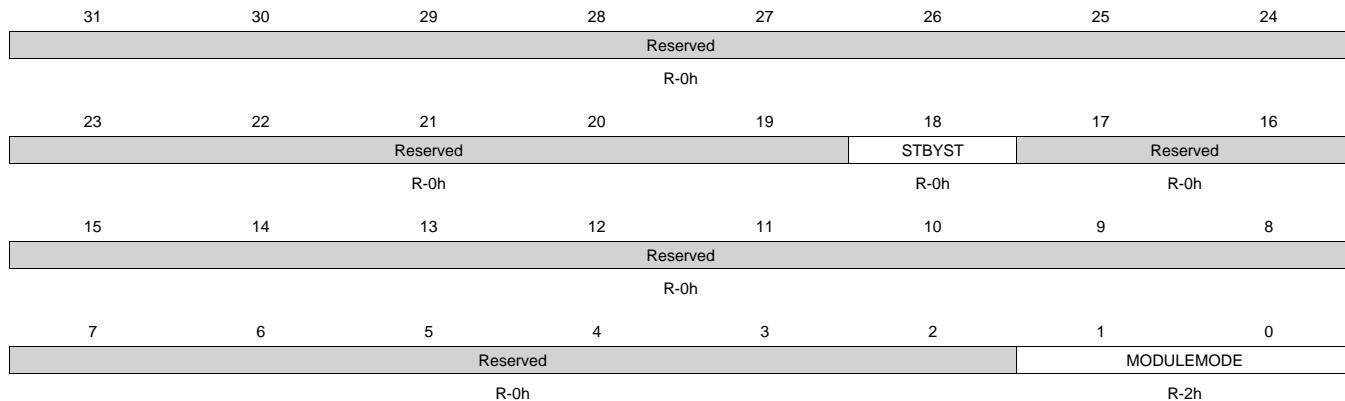
| Bit   | Field                    | Type | Reset | Description   |
|-------|--------------------------|------|-------|---|
| 31-10 | Reserved                 | R    | 0h    |   |
| 9     | ST_DPLL_CLKOUT           | R    | 0h    | DPLL CLKOUT status<br>0x0 = CLK_GATED : The clock output is gated<br>0x1 = CLK_ENABLED : The clock output is enabled  |
| 8     | DPLL_CLKOUT_GATE_CTRL    | R/W  | 0h    | Control gating of DPLL CLKOUT<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request |
| 7     | DPLL_CLKOUT_DIVCHA<br>CK | R    | 0h    | Toggle on this status bit after changing DPLL_CLKOUT_DIV indicates that the change in divider value has taken effect  |
| 6-0   | DPLL_CLKOUT_DIV          | R/W  | 1h    | DPLL M2 post-divider factor (1 to 31).  |

#### 8.1.12.2.45 CM\_WKUP\_WKUP\_M3\_CLKCTRL Register (offset = B0h) [reset = 2h]

CM\_WKUP\_WKUP\_M3\_CLKCTRL is shown in Figure 8-127 and described in Table 8-134.

This register manages the WKUP M3 clocks.

**Figure 8-127. CM\_WKUP\_WKUP\_M3\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-134. CM\_WKUP\_WKUP\_M3\_CLKCTRL Register Field Descriptions**

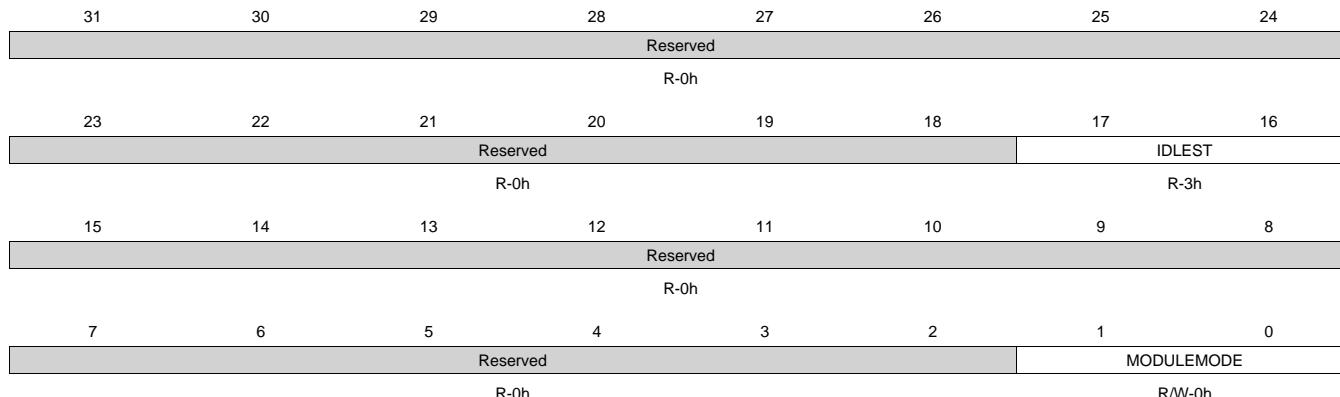
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-19 | Reserved   | R    | 0h    |  |
| 18    | STBYST     | R    | 0h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby |
| 17-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R    | 2h    | Control the way mandatory clocks are managed.  |

### 8.1.12.2.46 CM\_WKUP\_UART0\_CLKCTRL Register (offset = B4h) [reset = 30000h]

CM\_WKUP\_UART0\_CLKCTRL is shown in [Figure 8-128](#) and described in [Table 8-135](#).

This register manages the UART0 clocks.

**Figure 8-128. CM\_WKUP\_UART0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-135. CM\_WKUP\_UART0\_CLKCTRL Register Field Descriptions**

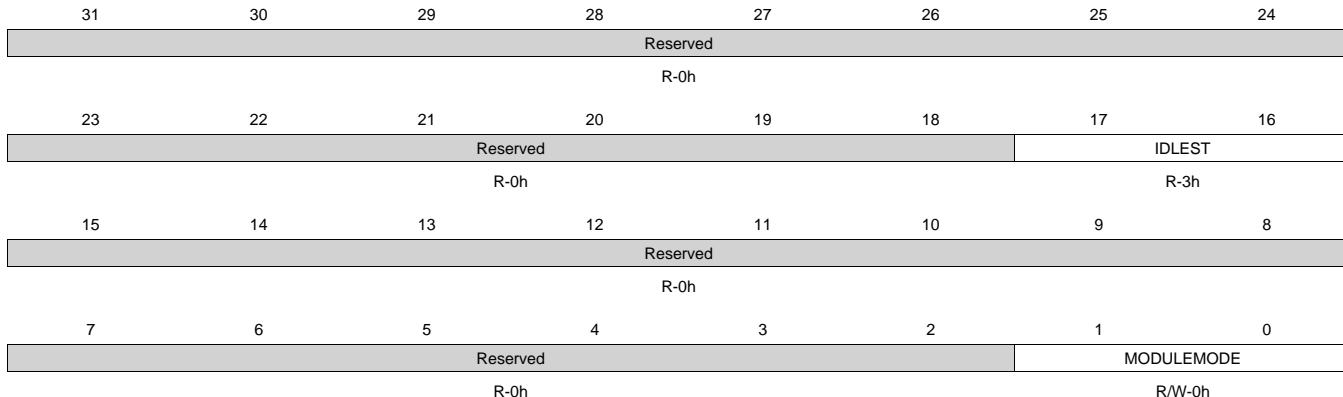
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.47 CM\_WKUP\_I2C0\_CLKCTRL Register (offset = B8h) [reset = 30000h]

CM\_WKUP\_I2C0\_CLKCTRL is shown in [Figure 8-129](#) and described in [Table 8-136](#).

This register manages the I2C0 clocks.

**Figure 8-129. CM\_WKUP\_I2C0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-136. CM\_WKUP\_I2C0\_CLKCTRL Register Field Descriptions**

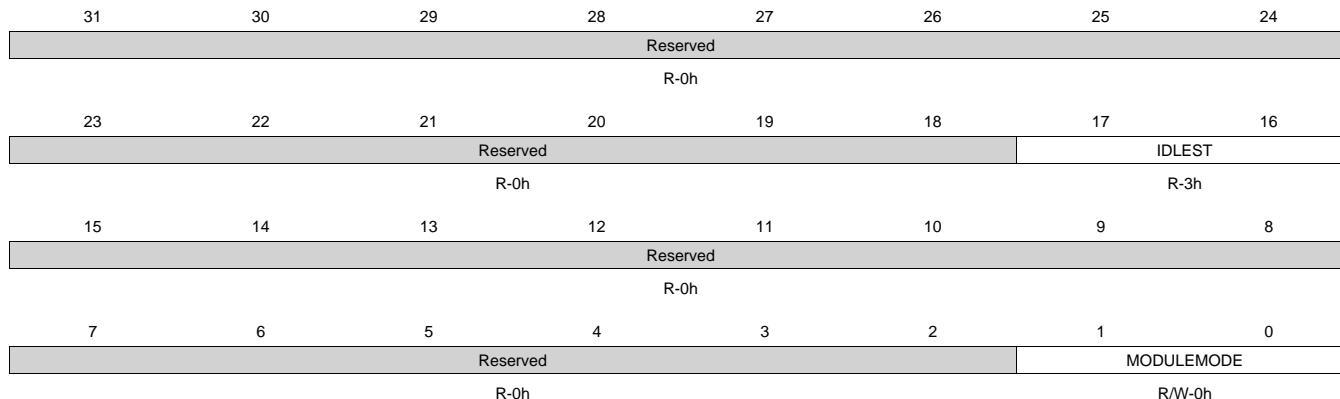
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.2.48 CM\_WKUP\_ADC\_TSC\_CLKCTRL Register (offset = BCh) [reset = 30000h]

CM\_WKUP\_ADC\_TSC\_CLKCTRL is shown in [Figure 8-130](#) and described in [Table 8-137](#).

This register manages the ADC clocks.

**Figure 8-130. CM\_WKUP\_ADC\_TSC\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-137. CM\_WKUP\_ADC\_TSC\_CLKCTRL Register Field Descriptions**

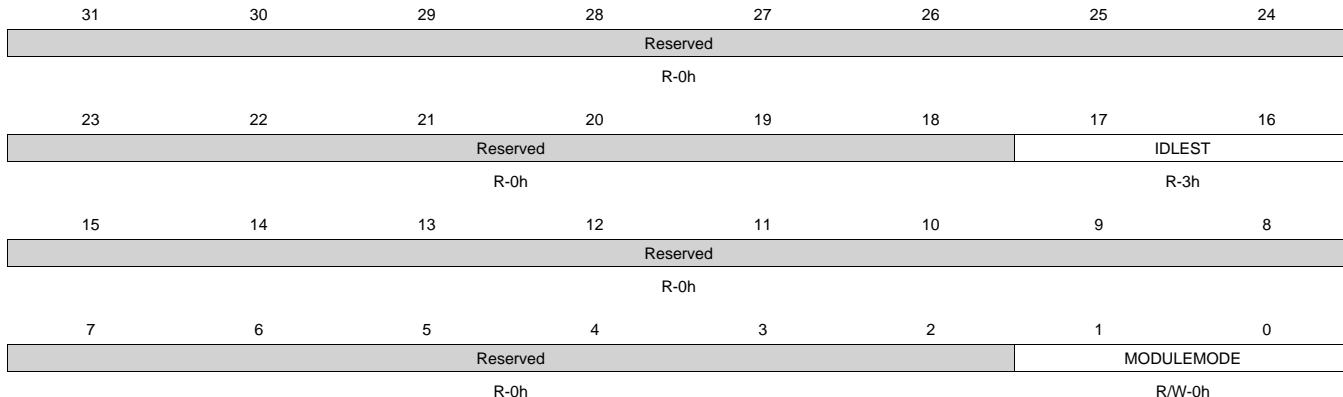
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.49 CM\_WKUP\_SMARTREFLEX0\_CLKCTRL Register (offset = C0h) [reset = 30000h]

CM\_WKUP\_SMARTREFLEX0\_CLKCTRL is shown in [Figure 8-131](#) and described in [Table 8-138](#).

This register manages the SmartReflex0 clocks.

**Figure 8-131. CM\_WKUP\_SMARTREFLEX0\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-138. CM\_WKUP\_SMARTREFLEX0\_CLKCTRL Register Field Descriptions**

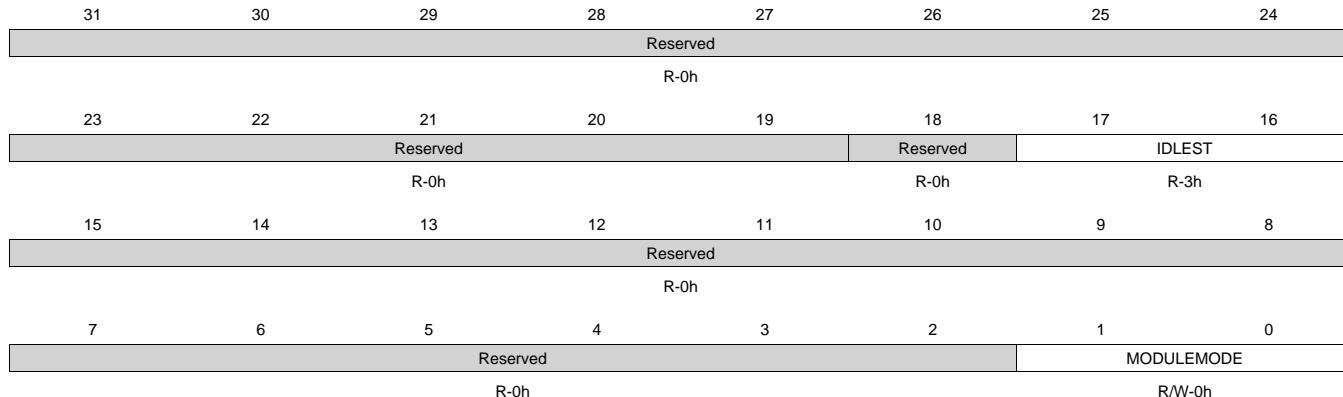
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.2.50 CM\_WKUP\_TIMER1\_CLKCTRL Register (offset = C4h) [reset = 30000h]

CM\_WKUP\_TIMER1\_CLKCTRL is shown in [Figure 8-132](#) and described in [Table 8-139](#).

This register manages the TIMER1 clocks.

**Figure 8-132. CM\_WKUP\_TIMER1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-139. CM\_WKUP\_TIMER1\_CLKCTRL Register Field Descriptions**

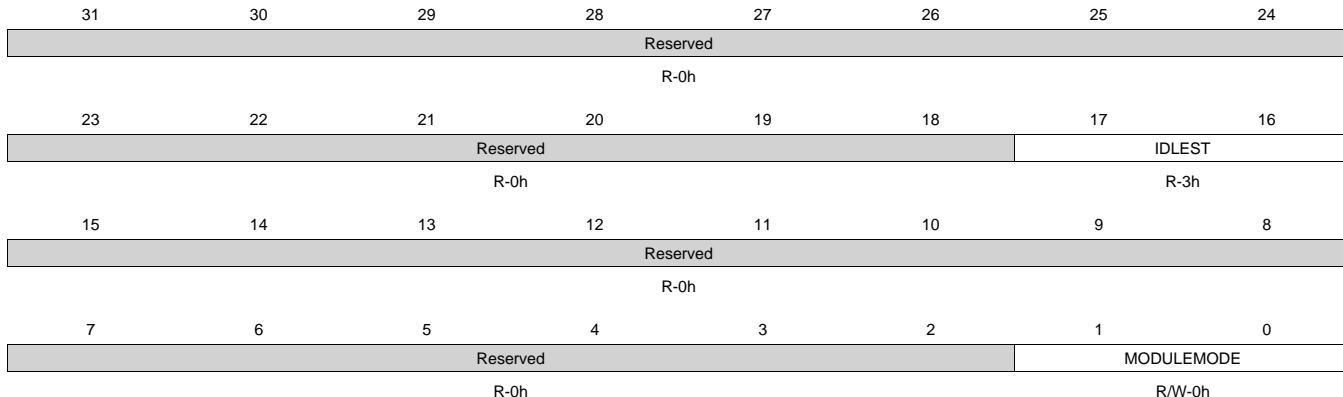
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.51 CM\_WKUP\_SMARTREFLEX1\_CLKCTRL Register (offset = C8h) [reset = 30000h]

CM\_WKUP\_SMARTREFLEX1\_CLKCTRL is shown in [Figure 8-133](#) and described in [Table 8-140](#).

This register manages the SmartReflex1 clocks.

**Figure 8-133. CM\_WKUP\_SMARTREFLEX1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-140. CM\_WKUP\_SMARTREFLEX1\_CLKCTRL Register Field Descriptions**

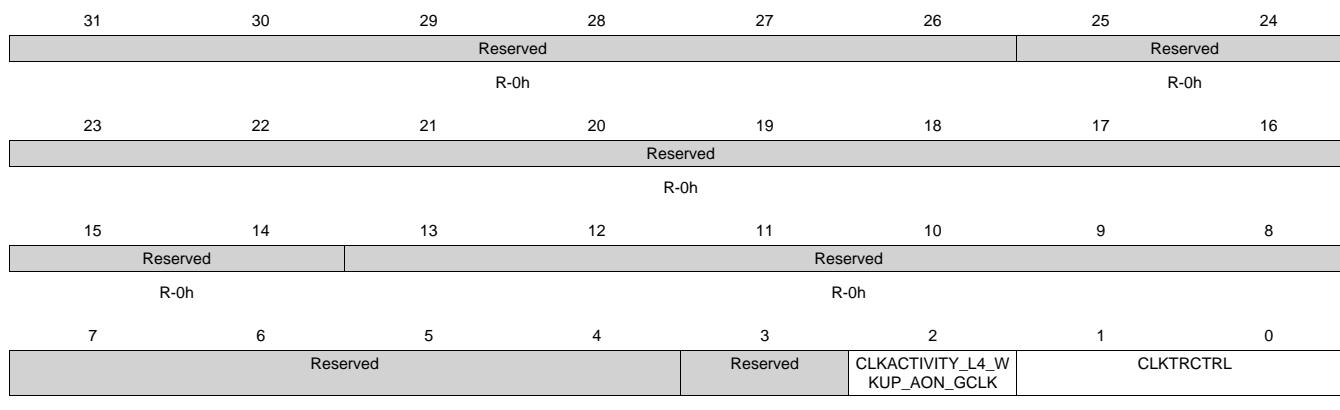
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.2.52 CM\_L4\_WKUP\_AON\_CLKSTCTRL Register (offset = CCh) [reset = 6h]

CM\_L4\_WKUP\_AON\_CLKSTCTRL is shown in [Figure 8-134](#) and described in [Table 8-141](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-134. CM\_L4\_WKUP\_AON\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-141. CM\_L4\_WKUP\_AON\_CLKSTCTRL Register Field Descriptions**

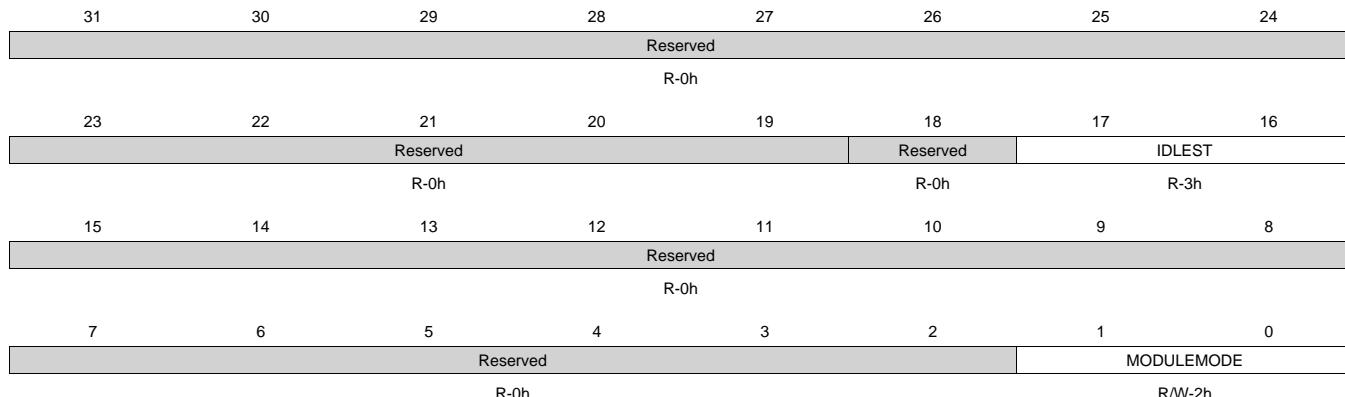
| Bit   | Field                        | Type | Reset | Description   |
|-------|------------------------------|------|-------|---|
| 31-26 | Reserved                     | R    | 0h    |   |
| 25-14 | Reserved                     | R    | 0h    |   |
| 13-8  | Reserved                     | R    | 0h    |   |
| 7-4   | Reserved                     | R    | 0h    |   |
| 3     | Reserved                     | R    | 0h    |   |
| 2     | CLKACTIVITY_L4_WKUP_AON_GCLK | R    | 1h    | This field indicates the state of the L4_WKUP clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 1-0   | CLKTRCTRL                    | R/W  | 2h    | Controls the clock state transition of the always on L4 clock domain.<br>0x0 = NO_SLEEP : Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.2.53 CM\_WKUP\_WDT1\_CLKCTRL Register (offset = D4h) [reset = 30002h]

CM\_WKUP\_WDT1\_CLKCTRL is shown in [Figure 8-135](#) and described in [Table 8-142](#).

This register manages the WDT1 clocks.

**Figure 8-135. CM\_WKUP\_WDT1\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-142. CM\_WKUP\_WDT1\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.2.54 CM\_DIV\_M6\_DPLL\_CORE Register (offset = D8h) [reset = 4h]

CM\_DIV\_M6\_DPLL\_CORE is shown in [Figure 8-136](#) and described in [Table 8-143](#).

This register provides controls over the CLKOUT3 o/p of the HSDIVIDER. [warm reset insensitive]

**Figure 8-136. CM\_DIV\_M6\_DPLL\_CORE Register**

|          |    |                            |                       |    |                      |                             |    |
|----------|----|----------------------------|-----------------------|----|----------------------|-----------------------------|----|
| 31       | 30 | 29                         | 28                    | 27 | 26                   | 25                          | 24 |
| Reserved |    |                            |                       |    |                      |                             |    |
| R-0h     |    |                            |                       |    |                      |                             |    |
| 23       | 22 | 21                         | 20                    | 19 | 18                   | 17                          | 16 |
| Reserved |    |                            |                       |    |                      |                             |    |
| R-0h     |    |                            |                       |    |                      |                             |    |
| 15       | 14 | 13                         | 12                    | 11 | 10                   | 9                           | 8  |
| Reserved |    | HSDIVIDER_CLKOUT3_PWDN     | Reserved              |    | ST_HSDIVIDER_CLKOUT3 | HSDIVIDER_CLKOUT3_GATE_CTRL |    |
| R-0h     |    | R/W-0h                     | R-0h                  |    | R-0h                 | R/W-0h                      |    |
| 7        | 6  | 5                          | 4                     | 3  | 2                    | 1                           | 0  |
| Reserved |    | HSDIVIDER_CLKOUT3_DIVCHACK | HSDIVIDER_CLKOUT3_DIV |    |                      |                             |    |
| R-0h     |    | R-0h                       | R/W-4h                |    |                      |                             |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-143. CM\_DIV\_M6\_DPLL\_CORE Register Field Descriptions**

| Bit   | Field                       | Type | Reset | Description   |
|-------|-----------------------------|------|-------|---|
| 31-13 | Reserved                    | R    | 0h    |   |
| 12    | HSDIVIDER_CLKOUT3_PWDN      | R/W  | 0h    | Automatic power down for HSDIVIDER M6 divider and hence CLKOUT3 output when the o/p clock is gated.<br>0x0 = ALWAYS_ACTIVE : Keep M6 divider powered on even when CLKOUT3 is gated.<br>0x1 = AUTO_PWDN : Automatically power down M6 divider when CLKOUT3 is gated. |
| 11-10 | Reserved                    | R    | 0h    |   |
| 9     | ST_HSDIVIDER_CLKOUT3        | R    | 0h    | HSDIVIDER CLKOUT3 status<br>0x0 = CLK_ENABLED : The clock output is enabled<br>0x1 = CLK_GATED : The clock output is gated  |
| 8     | HSDIVIDER_CLKOUT3_GATE_CTRL | R/W  | 0h    | Control gating of HSDIVIDER CLKOUT3<br>0x0 = CLK_AUTOGATE : Automatically gate this clock when there is no dependency for it<br>0x1 = CLK_ENABLE : Force this clock to stay enabled even if there is no request   |
| 7-6   | Reserved                    | R    | 0h    |   |
| 5     | HSDIVIDER_CLKOUT3_DIVCHACK  | R    | 0h    | Toggle on this status bit after changing HSDIVIDER_CLKOUT3_DIV indicates that the change in divider value has taken effect  |
| 4-0   | HSDIVIDER_CLKOUT3_DIV       | R/W  | 4h    | DPLL post-divider factor, M6, for internal clock generation.<br>Divide values from 1 to 31.   |

### 8.1.12.3 CM\_DPLL Registers

[Table 8-144](#) lists the memory-mapped registers for the CM\_DPLL. All register offset addresses not listed in [Table 8-144](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-144. CM\_DPLL REGISTERS**

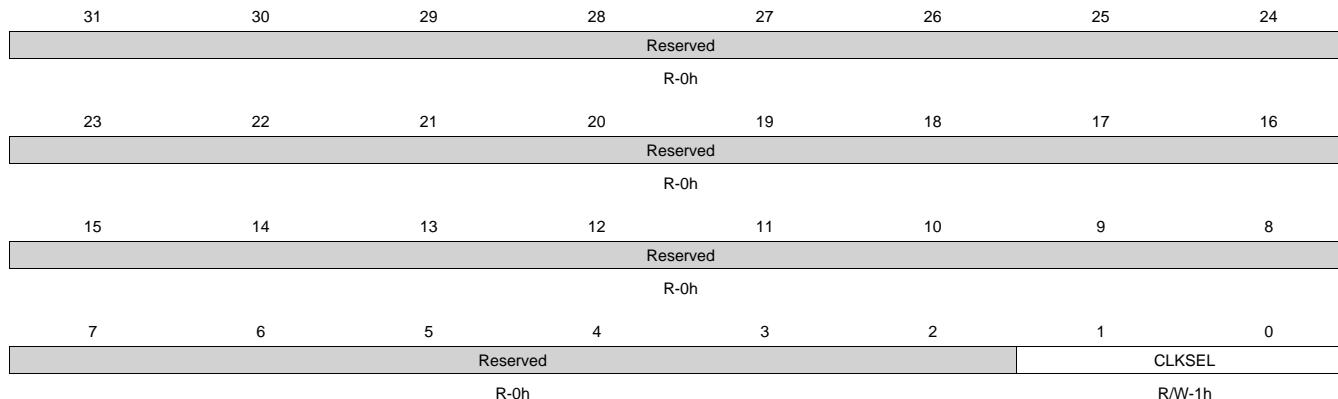
| Offset | Acronym                 | Register Name   | Section                             |
|--------|-------------------------|---|-------------------------------------|
| 4h     | CLKSEL_TIMER7_CLK       | Selects the Mux select line for TIMER7 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.1</a>  |
| 8h     | CLKSEL_TIMER2_CLK       | Selects the Mux select line for TIMER2 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.2</a>  |
| Ch     | CLKSEL_TIMER3_CLK       | Selects the Mux select line for TIMER3 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.3</a>  |
| 10h    | CLKSEL_TIMER4_CLK       | Selects the Mux select line for TIMER4 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.4</a>  |
| 14h    | CM_MAC_CLKSEL           | Selects the clock divide ration for MII clock [warm reset insensitive]        | <a href="#">Section 8.1.12.3.5</a>  |
| 18h    | CLKSEL_TIMER5_CLK       | Selects the Mux select line for TIMER5 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.6</a>  |
| 1Ch    | CLKSEL_TIMER6_CLK       | Selects the Mux select line for TIMER6 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.7</a>  |
| 20h    | CM_CPTS_RFT_CLKSEL      | Selects the Mux select line for CPTS RFT clock [warm reset insensitive]       | <a href="#">Section 8.1.12.3.8</a>  |
| 28h    | CLKSEL_TIMER1MS_CLK     | Selects the Mux select line for TIMER1 clock [warm reset insensitive]         | <a href="#">Section 8.1.12.3.9</a>  |
| 2Ch    | CLKSEL_GFX_FCLK         | Selects the divider value for GFX clock [warm reset insensitive]              | <a href="#">Section 8.1.12.3.10</a> |
| 30h    | CLKSEL_PRU_ICSS_OCP_CLK | Controls the Mux select line for PRU-ICSS OCP clock [warm reset insensitive]  | <a href="#">Section 8.1.12.3.11</a> |
| 34h    | CLKSEL_LCDC_PIXEL_CLK   | Controls the Mux select line for LCDC PIXEL clock [warm reset insensitive]    | <a href="#">Section 8.1.12.3.12</a> |
| 38h    | CLKSEL_WDT1_CLK         | Selects the Mux select line for Watchdog1 clock [warm reset insensitive]      | <a href="#">Section 8.1.12.3.13</a> |
| 3Ch    | CLKSEL_GPIO0_DBCLK      | Selects the Mux select line for GPIO0 debounce clock [warm reset insensitive] | <a href="#">Section 8.1.12.3.14</a> |

### 8.1.12.3.1 CLKSEL\_TIMER7\_CLK Register (offset = 4h) [reset = 1h]

CLKSEL\_TIMER7\_CLK is shown in [Figure 8-137](#) and described in [Table 8-145](#).

Selects the Mux select line for TIMER7 clock [warm reset insensitive]

**Figure 8-137. CLKSEL\_TIMER7\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-145. CLKSEL\_TIMER7\_CLK Register Field Descriptions**

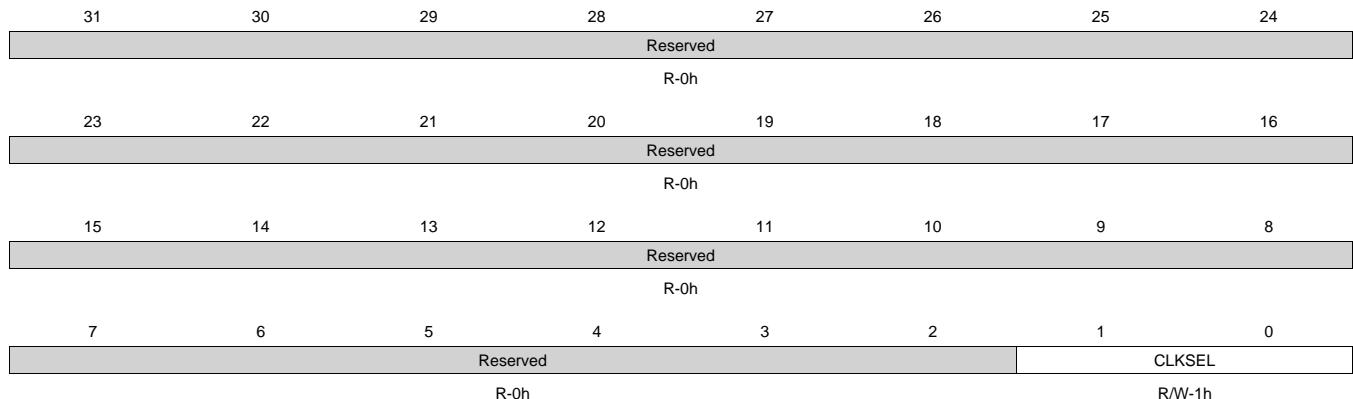
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER7 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

### 8.1.12.3.2 CLKSEL\_TIMER2\_CLK Register (offset = 8h) [reset = 1h]

CLKSEL\_TIMER2\_CLK is shown in [Figure 8-138](#) and described in [Table 8-146](#).

Selects the Mux select line for TIMER2 clock [warm reset insensitive]

**Figure 8-138. CLKSEL\_TIMER2\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-146. CLKSEL\_TIMER2\_CLK Register Field Descriptions**

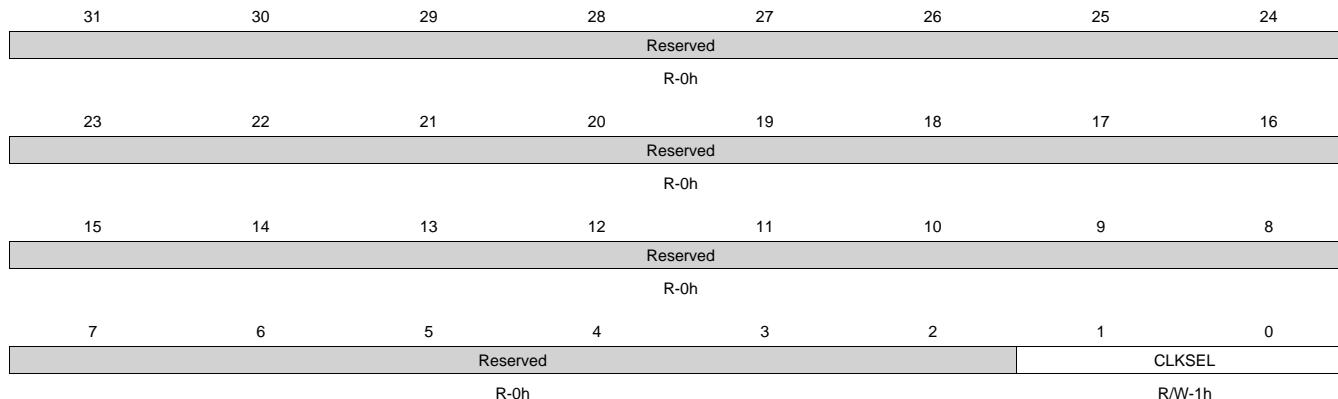
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER2 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

### 8.1.12.3.3 CLKSEL\_TIMER3\_CLK Register (offset = Ch) [reset = 1h]

CLKSEL\_TIMER3\_CLK is shown in [Figure 8-139](#) and described in [Table 8-147](#).

Selects the Mux select line for TIMER3 clock [warm reset insensitive]

**Figure 8-139. CLKSEL\_TIMER3\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-147. CLKSEL\_TIMER3\_CLK Register Field Descriptions**

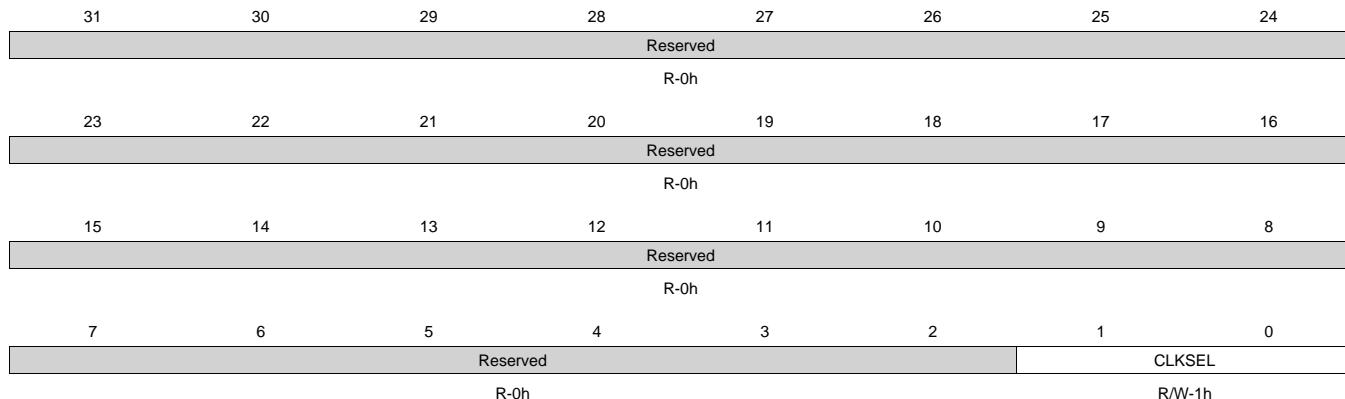
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER3 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

#### 8.1.12.3.4 CLKSEL\_TIMER4\_CLK Register (offset = 10h) [reset = 1h]

CLKSEL\_TIMER4\_CLK is shown in [Figure 8-140](#) and described in [Table 8-148](#).

Selects the Mux select line for TIMER4 clock [warm reset insensitive]

**Figure 8-140. CLKSEL\_TIMER4\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-148. CLKSEL\_TIMER4\_CLK Register Field Descriptions**

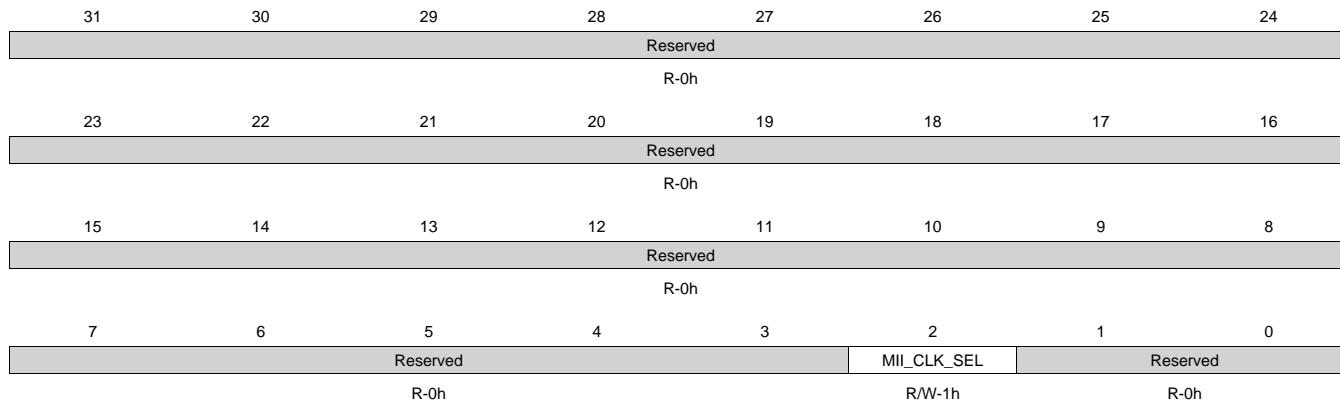
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER4 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

### 8.1.12.3.5 CM\_MAC\_CLKSEL Register (offset = 14h) [reset = 4h]

CM\_MAC\_CLKSEL is shown in [Figure 8-141](#) and described in [Table 8-149](#).

Selects the clock divide ration for MII clock [warm reset insensitive]

**Figure 8-141. CM\_MAC\_CLKSEL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-149. CM\_MAC\_CLKSEL Register Field Descriptions**

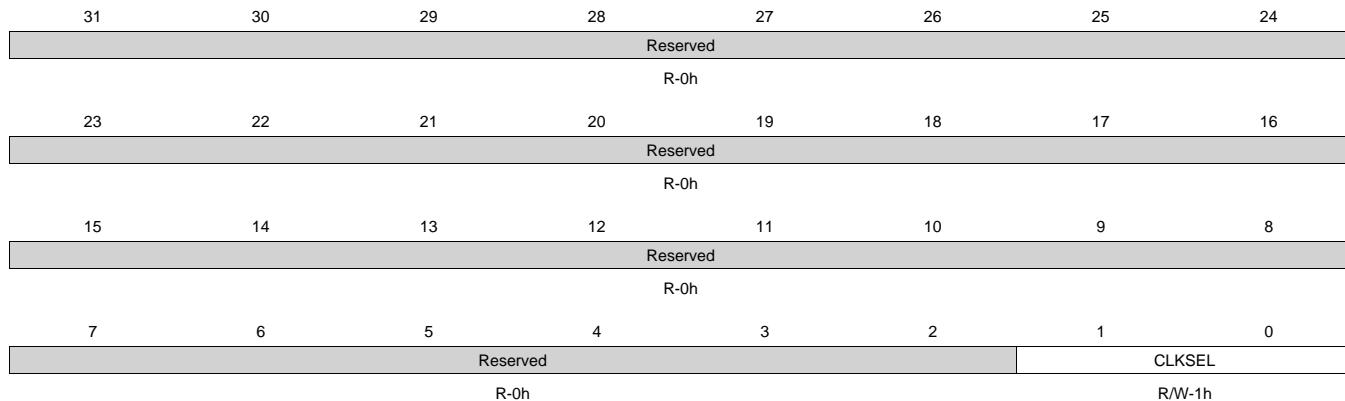
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-3 | Reserved    | R    | 0h    |  |
| 2    | MII_CLK_SEL | R/W  | 1h    | MII Clock Divider Selection.<br>This bit is warm reset insensitive when CPSW RESET_ISO is enabled<br>0x0 = SEL0 : Selects 1/2 divider of SYSCLK2<br>0x1 = SEL1 : Selects 1/5 divide ratio of SYSCLK2 |
| 1-0  | Reserved    | R    | 0h    |  |

### 8.1.12.3.6 CLKSEL\_TIMER5\_CLK Register (offset = 18h) [reset = 1h]

CLKSEL\_TIMER5\_CLK is shown in [Figure 8-142](#) and described in [Table 8-150](#).

Selects the Mux select line for TIMER5 clock [warm reset insensitive]

**Figure 8-142. CLKSEL\_TIMER5\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-150. CLKSEL\_TIMER5\_CLK Register Field Descriptions**

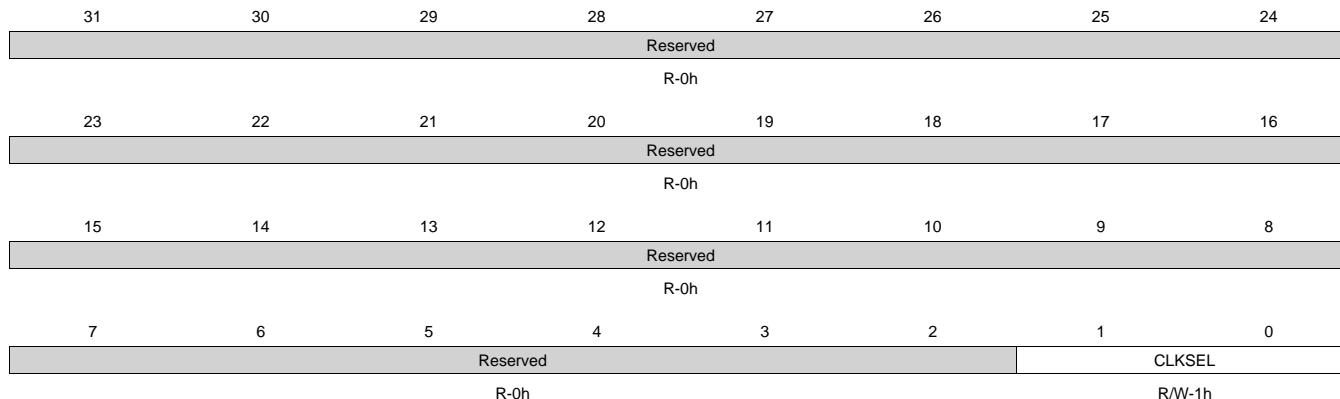
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER5 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

### 8.1.12.3.7 CLKSEL\_TIMER6\_CLK Register (offset = 1Ch) [reset = 1h]

CLKSEL\_TIMER6\_CLK is shown in [Figure 8-143](#) and described in [Table 8-151](#).

Selects the Mux select line for TIMER6 clock [warm reset insensitive]

**Figure 8-143. CLKSEL\_TIMER6\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-151. CLKSEL\_TIMER6\_CLK Register Field Descriptions**

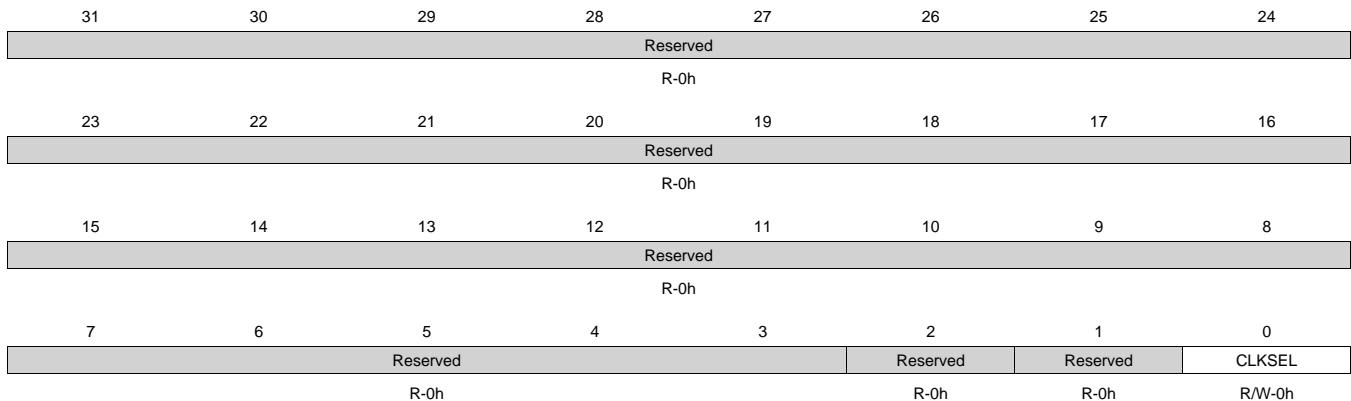
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 1h    | Selects the Mux select line for TIMER6 clock [warm reset insensitive]<br>0x0 = SEL1 : Select TCLKIN clock<br>0x1 = SEL2 : Select CLK_M_OSC clock<br>0x2 = SEL3 : Select CLK_32KHZ clock<br>0x3 = SEL4 : Reserved |

#### 8.1.12.3.8 CM\_CPTS\_RFT\_CLKSEL Register (offset = 20h) [reset = 0h]

CM\_CPTS\_RFT\_CLKSEL is shown in [Figure 8-144](#) and described in [Table 8-152](#).

Selects the Mux select line for CPTS RFT clock [warm reset insensitive]

**Figure 8-144. CM\_CPTS\_RFT\_CLKSEL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-152. CM\_CPTS\_RFT\_CLKSEL Register Field Descriptions**

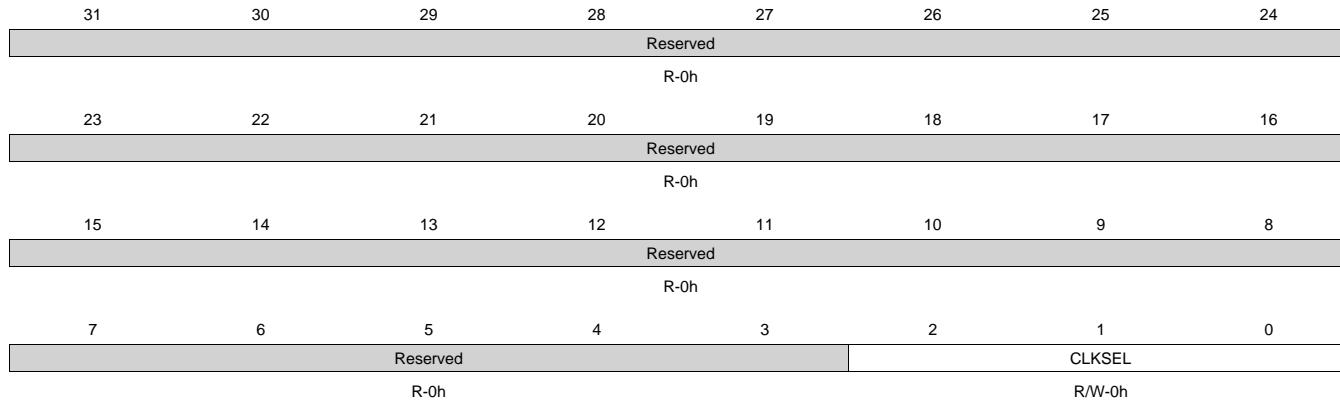
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-3 | Reserved | R    | 0h    |   |
| 2    | Reserved | R    | 0h    |   |
| 1    | Reserved | R    | 0h    |   |
| 0    | CLKSEL   | R/W  | 0h    | Selects the Mux select line for cpgmac rft clock [warm reset insensitive]<br>0x0 = SEL1 : Selects CORE_CLKOUTM5<br>0x1 = SEL2 : Selects CORE_CLKOUTM4 |

### 8.1.12.3.9 CLKSEL\_TIMER1MS\_CLK Register (offset = 28h) [reset = 0h]

CLKSEL\_TIMER1MS\_CLK is shown in [Figure 8-145](#) and described in [Table 8-153](#).

Selects the Mux select line for TIMER1 clock [warm reset insensitive]

**Figure 8-145. CLKSEL\_TIMER1MS\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-153. CLKSEL\_TIMER1MS\_CLK Register Field Descriptions**

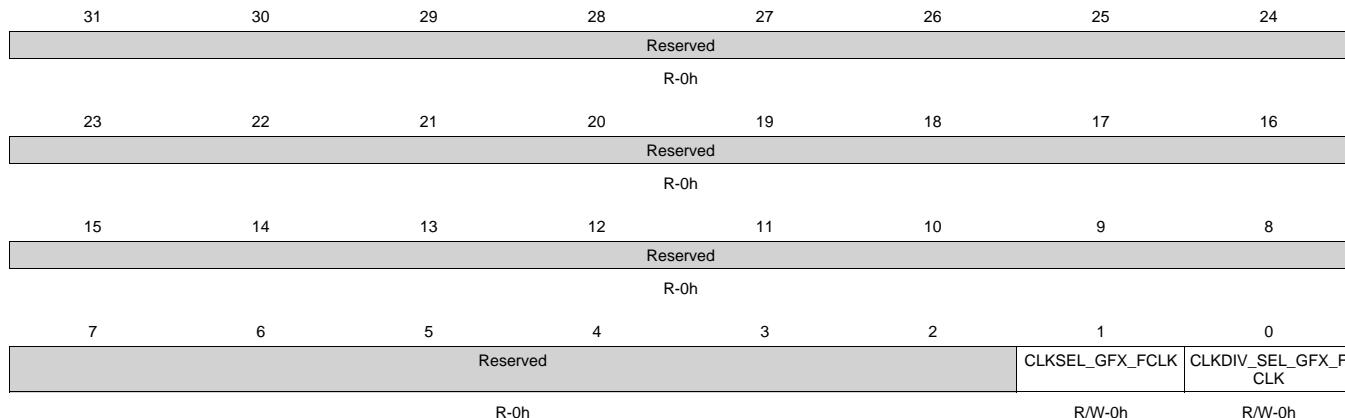
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-3 | Reserved | R    | 0h    |  |
| 2-0  | CLKSEL   | R/W  | 0h    | Selects the Mux select line for DMTIMER_1MS clock [warm reset insensitive]<br>0x0 = SEL1 : Select CLK_M_OSC clock<br>0x1 = SEL2 : Select CLK_32KHZ clock<br>0x2 = SEL3 : Select TCLKIN clock<br>0x3 = SEL4 : Select CLK_RC32K clock<br>0x4 = SEL5 : Selects the CLK_32768 from 32KHz Crystal Osc |

### 8.1.12.3.10 CLKSEL\_GFX\_FCLK Register (offset = 2Ch) [reset = 0h]

CLKSEL\_GFX\_FCLK is shown in [Figure 8-146](#) and described in [Table 8-154](#).

Selects the divider value for GFX clock [warm reset insensitive]

**Figure 8-146. CLKSEL\_GFX\_FCLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-154. CLKSEL\_GFX\_FCLK Register Field Descriptions**

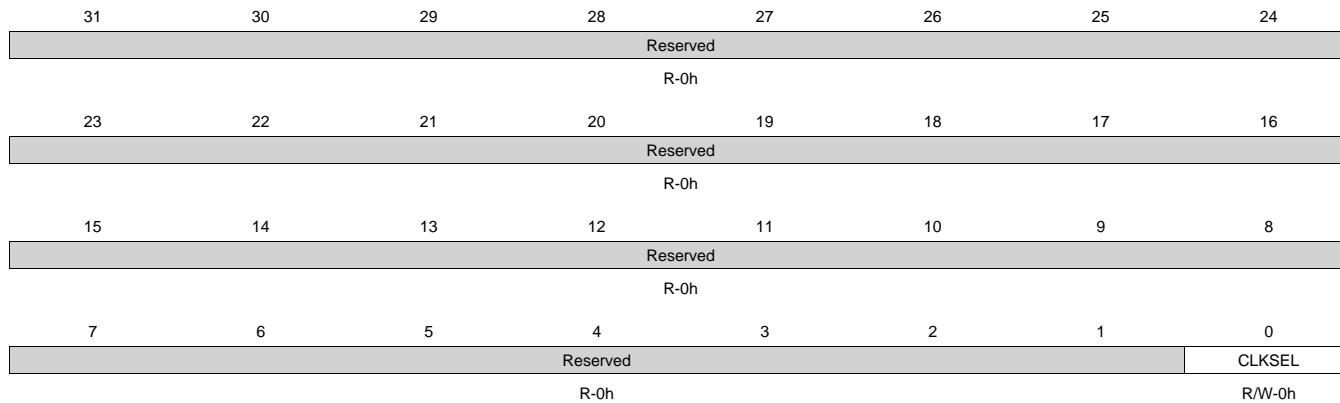
| Bit  | Field               | Type | Reset | Description  |
|------|---------------------|------|-------|--|
| 31-2 | Reserved            | R    | 0h    |  |
| 1    | CLKSEL_GFX_FCLK     | R/W  | 0h    | Selects the clock on gfx fclk [warm reset insensitive]<br>0x0 = SEL0 : SGX FCLK is from CORE PLL (same as L3 clock)<br>0x1 = SEL1 : SGX FCLK is from PER PLL (192 MHz clock)   |
| 0    | CLKDIV_SEL_GFX_FCLK | R/W  | 0h    | Selects the divider value on gfx fclk [warm reset insensitive]<br>0x0 = DIV1 : SGX FCLK is same as L3 Clock or 192MHz Clock<br>0x1 = DIV2 : SGX FCLK is L3 clock/2 or 192Mhz/2 |

### 8.1.12.3.11 CLKSEL\_PRU\_ICSS\_OCP\_CLK Register (offset = 30h) [reset = 0h]

CLKSEL\_PRU\_ICSS\_OCP\_CLK is shown in [Figure 8-147](#) and described in [Table 8-155](#).

Controls the Mux select line for PRU-ICSS OCP clock [warm reset insensitive]

**Figure 8-147. CLKSEL\_PRU\_ICSS\_OCP\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-155. CLKSEL\_PRU\_ICSS\_OCP\_CLK Register Field Descriptions**

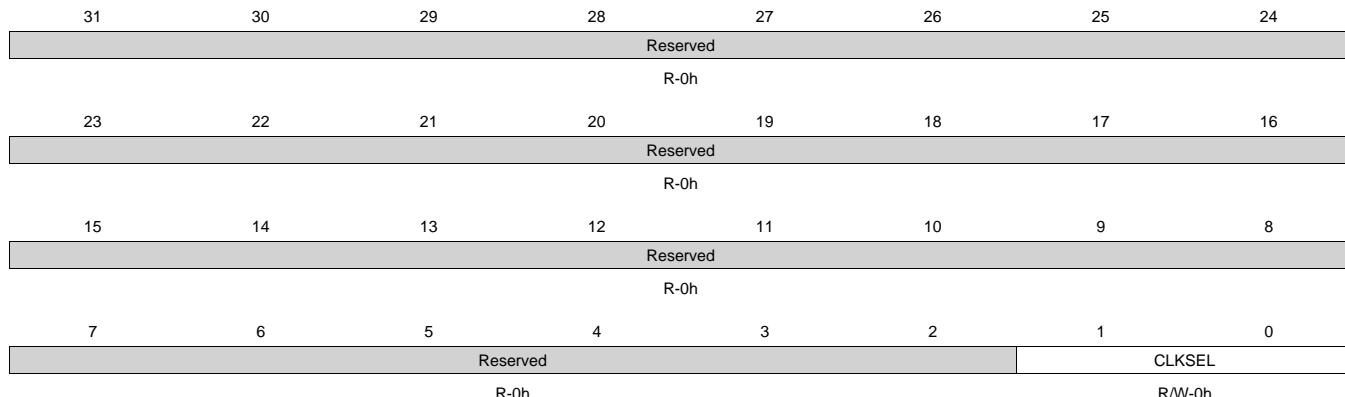
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-1 | Reserved | R    | 0h    |   |
| 0    | CLKSEL   | R/W  | 0h    | Controls Mux Select of PRU-ICSS OCP clock mux<br>0x0 = SEL1 : Select L3F clock as OCP Clock of PRU-ICSS<br>0x1 = SEL2 : Select DISP DPLL clock as OCP clock of PRU-ICSS |

### 8.1.12.3.12 CLKSEL\_LCDC\_PIXEL\_CLK Register (offset = 34h) [reset = 0h]

CLKSEL\_LCDC\_PIXEL\_CLK is shown in [Figure 8-148](#) and described in [Table 8-156](#).

Controls the Mux select line for LCDC PIXEL clock [warm reset insensitive]

**Figure 8-148. CLKSEL\_LCDC\_PIXEL\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-156. CLKSEL\_LCDC\_PIXEL\_CLK Register Field Descriptions**

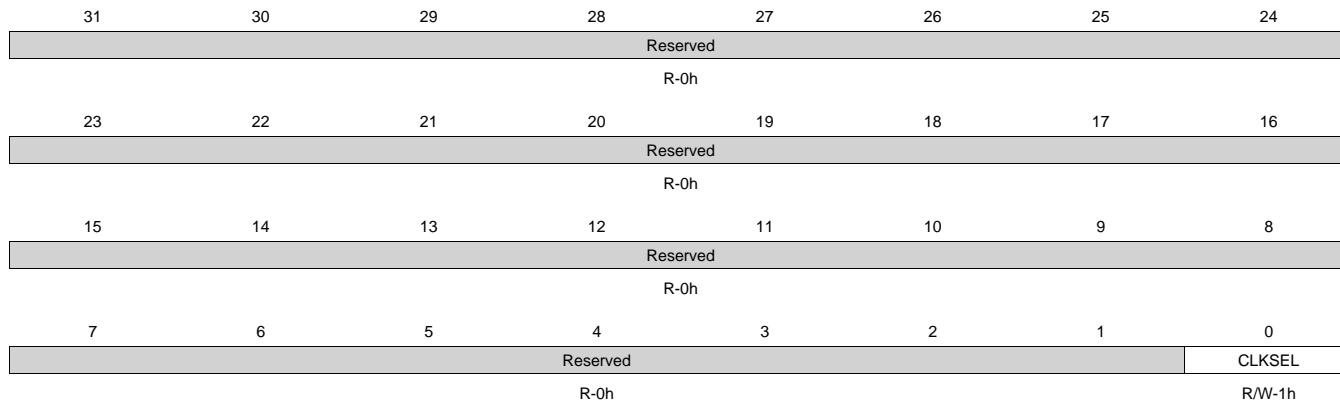
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 0h    | Controls the Mux Select of LCDC PIXEL clock<br>0x0 = SEL1 : Select DISP PLL CLKOUTM2<br>0x1 = SEL2 : Select CORE PLL CLKOUTM5<br>0x2 = SEL3 : Select PER PLL CLKOUTM2<br>0x3 = SEL4 : Reserved |

### 8.1.12.3.13 CLKSEL\_WDT1\_CLK Register (offset = 38h) [reset = 1h]

CLKSEL\_WDT1\_CLK is shown in [Figure 8-149](#) and described in [Table 8-157](#).

Selects the Mux select line for Watchdog1 clock [warm reset insensitive]

**Figure 8-149. CLKSEL\_WDT1\_CLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-157. CLKSEL\_WDT1\_CLK Register Field Descriptions**

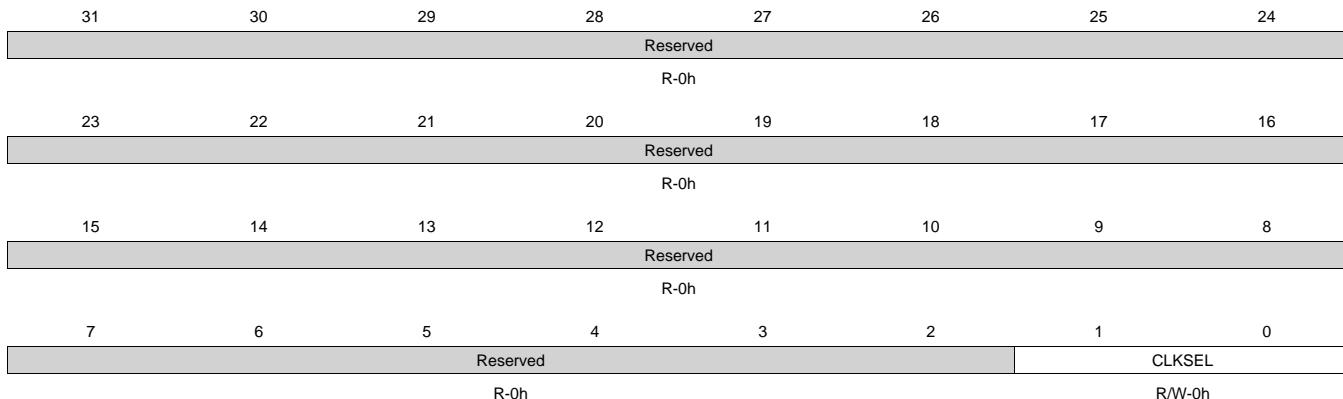
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-1 | Reserved | R    | 0h    |   |
| 0    | CLKSEL   | R/W  | 1h    | Selects the Mux select line for WDT1 clock [warm reset insensitive]<br>0x0 = SEL1 : Select 32KHZ clock from RC Oscillator<br>0x1 = SEL2 : Select 32KHZ from 32K Clock divider |

#### 8.1.12.3.14 CLKSEL\_GPIO0\_DBCLK Register (offset = 3Ch) [reset = 0h]

CLKSEL\_GPIO0\_DBCLK is shown in [Figure 8-150](#) and described in [Table 8-158](#).

Selects the Mux select line for GPIO0 debounce clock [warm reset insensitive]

**Figure 8-150. CLKSEL\_GPIO0\_DBCLK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-158. CLKSEL\_GPIO0\_DBCLK Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-2 | Reserved | R    | 0h    |  |
| 1-0  | CLKSEL   | R/W  | 0h    | Selects the Mux select line for GPIO0 debounce clock [warm reset insensitive]<br>0x0 = SEL1 : Select 32KHZ clock from RC Oscillator<br>0x1 = SEL2 : Select 32KHZ from 32K Crystal Oscillator<br>0x2 = SEL3 : Select 32KHz from Clock Divider |

#### 8.1.12.4 CM\_MPU Registers

[Table 8-159](#) lists the memory-mapped registers for the CM\_MPU. All register offset addresses not listed in [Table 8-159](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-159. CM\_MPU REGISTERS**

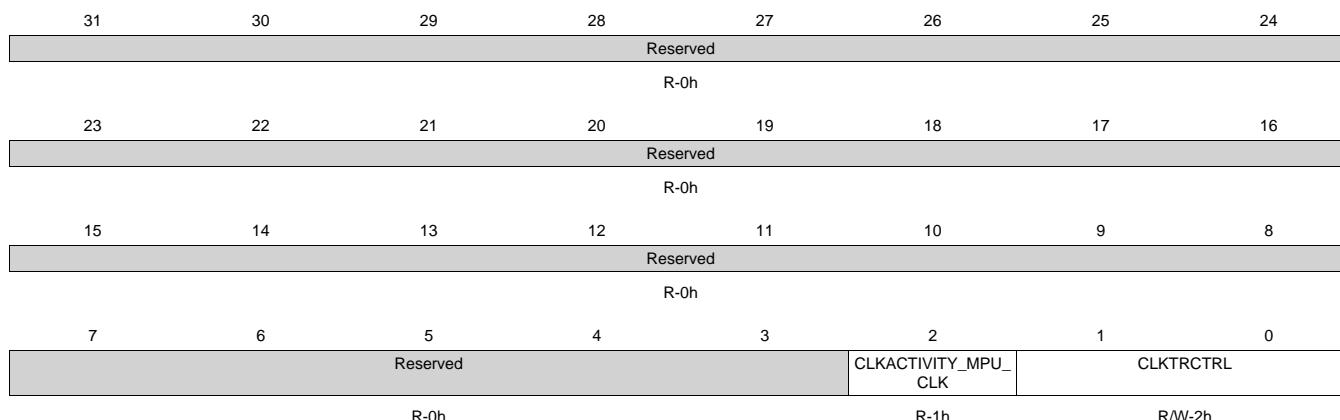
| Offset | Acronym           | Register Name  | Section                            |
|--------|-------------------|--|------------------------------------|
| 0h     | CM_MPU_CLKSTCTRL  | This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.4.1</a> |
| 4h     | CM_MPU_MPUCLKCTRL | This register manages the MPU clocks.  | <a href="#">Section 8.1.12.4.2</a> |

#### 8.1.12.4.1 CM\_MPU\_CLKSTCTRL Register (offset = 0h) [reset = 6h]

CM\_MPU\_CLKSTCTRL is shown in [Figure 8-151](#) and described in [Table 8-160](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-151. CM\_MPU\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-160. CM\_MPU\_CLKSTCTRL Register Field Descriptions**

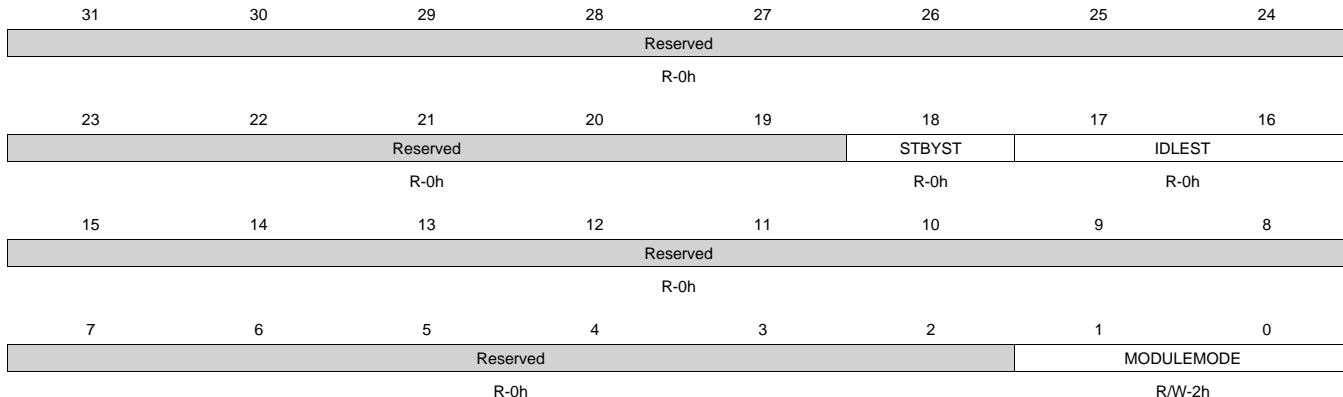
| Bit  | Field               | Type | Reset | Description   |
|------|---------------------|------|-------|---|
| 31-3 | Reserved            | R    | 0h    |   |
| 2    | CLKACTIVITY_MPU_CLK | R    | 1h    | This field indicates the state of the MPU Clock<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 1-0  | CLKTRCTRL           | R/W  | 2h    | Controls the clock state transition of the MPU clock domains.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated.<br>Wakeups may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.4.2 CM\_MPU\_MPU\_CLKCTRL Register (offset = 4h) [reset = 2h]

CM\_MPU\_MPU\_CLKCTRL is shown in [Figure 8-152](#) and described in [Table 8-161](#).

This register manages the MPU clocks.

**Figure 8-152. CM\_MPU\_MPU\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-161. CM\_MPU\_MPU\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 0h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 0h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.5 CM\_DEVICE Registers

[Table 8-162](#) lists the memory-mapped registers for the CM\_DEVICE. All register offset addresses not listed in [Table 8-162](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-162. CM\_DEVICE REGISTERS**

| Offset | Acronym        | Register Name  | Section                            |
|--------|----------------|--|------------------------------------|
| 0h     | CM_CLKOUT_CTRL | This register provides the control over CLKOUT2 output | <a href="#">Section 8.1.12.5.1</a> |

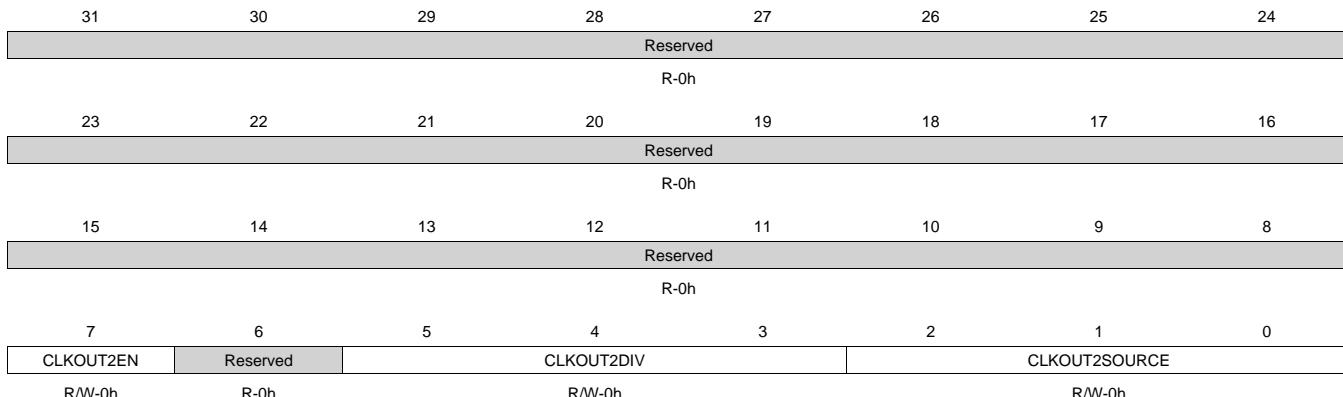


### 8.1.12.5.1 CM\_CLKOUT\_CTRL Register (offset = 0h) [reset = 0h]

CM\_CLKOUT\_CTRL is shown in [Figure 8-153](#) and described in [Table 8-163](#).

This register provides the control over CLKOUT2 output

**Figure 8-153. CM\_CLKOUT\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-163. CM\_CLKOUT\_CTRL Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-8 | Reserved      | R    | 0h    |   |
| 7    | CLKOUT2EN     | R/W  | 0h    | This bit controls the external clock activity<br>0x0 = DIS : SYS_CLKOUT2 is disabled<br>0x1 = EN : SYS_CLKOUT2 is enabled   |
| 6    | Reserved      | R    | 0h    |   |
| 5-3  | CLKOUT2DIV    | R/W  | 0h    | This field controls the external clock divisor factor<br>0x0 = DIV1 : SYS_CLKOUT2/1<br>0x1 = DIV2 : SYS_CLKOUT2/2<br>0x2 = DIV3 : SYS_CLKOUT2/3<br>0x3 = DIV4 : SYS_CLKOUT2/4<br>0x4 = DIV5 : SYS_CLKOUT2/5<br>0x5 = DIV6 : SYS_CLKOUT2/6<br>0x6 = DIV7 : SYS_CLKOUT2/7<br>0x7 = DIV8 : SYS_CLKOUT2/8 |
| 2-0  | CLKOUT2SOURCE | R/W  | 0h    | This field selects the external output clock source<br>0x0 = SEL0 : Select 32KHz Oscillator O/P<br>0x1 = SEL1 : Select L3 Clock<br>0x2 = SEL2 : Select DDR PHY Clock<br>0x3 = SEL4 : Select 192Mhz clock from PER PLL<br>0x4 = SEL5 : Select LCD Pixel Clock  |

### 8.1.12.6 CM\_RTC Registers

[Table 8-164](#) lists the memory-mapped registers for the CM\_RTC. All register offset addresses not listed in [Table 8-164](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-164. CM\_RTC REGISTERS**

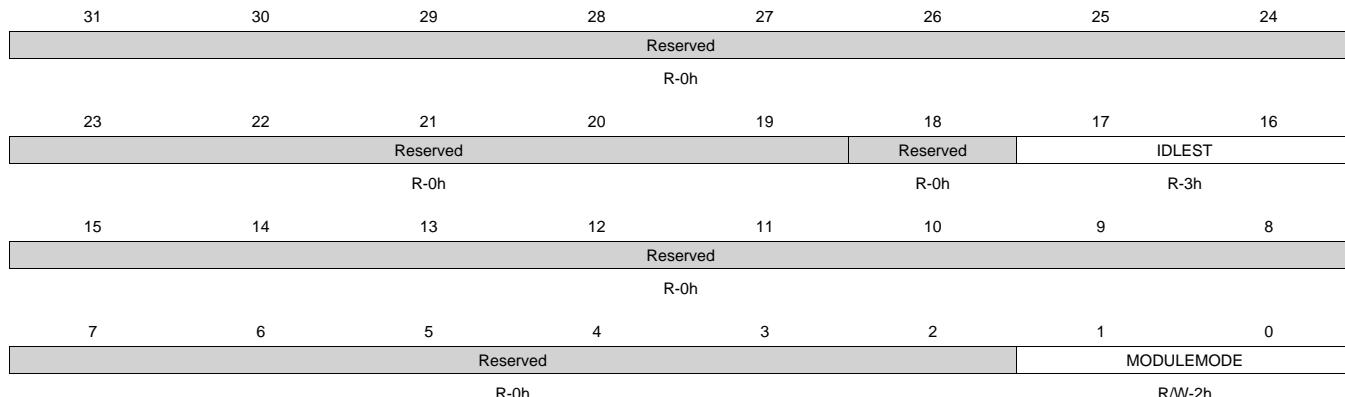
| <b>Offset</b> | <b>Acronym</b>     | <b>Register Name</b>  | <b>Section</b>                     |
|---------------|--------------------|---|------------------------------------|
| 0h            | CM_RTC_RTC_CLKCTRL | This register manages the RTC clocks.   | <a href="#">Section 8.1.12.6.1</a> |
| 4h            | CM_RTC_CLKSTCTRL   | This register enables the domain power state transition.<br>It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.6.2</a> |

### 8.1.12.6.1 CM\_RTC\_RTC\_CLKCTRL Register (offset = 0h) [reset = 30002h]

CM\_RTC\_RTC\_CLKCTRL is shown in Figure 8-154 and described in Table 8-165.

This register manages the RTC clocks.

**Figure 8-154. CM\_RTC\_RTC\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-165. CM\_RTC\_RTC\_CLKCTRL Register Field Descriptions**

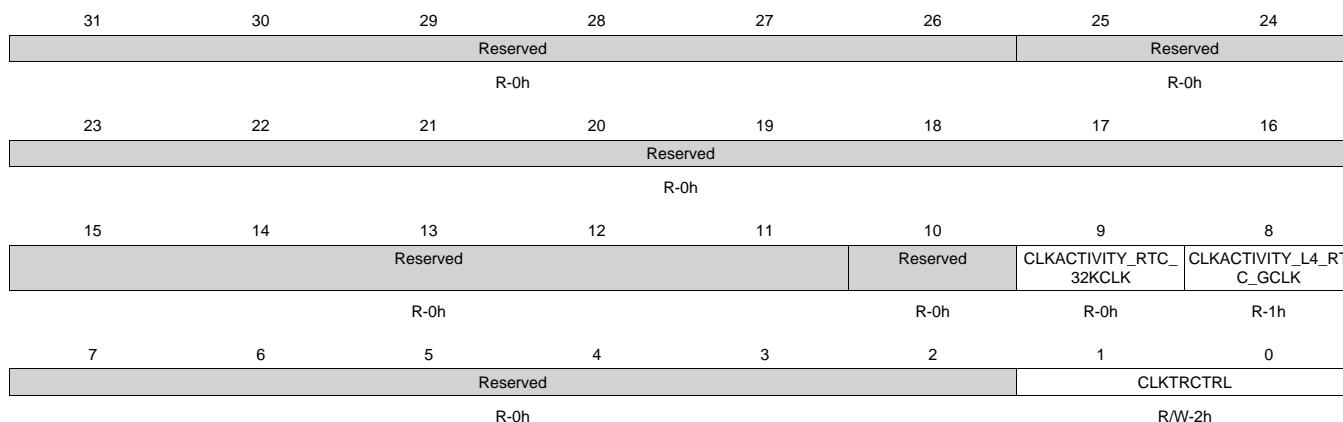
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 2h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.6.2 CM\_RTC\_CLKSTCTRL Register (offset = 4h) [reset = 102h]

CM\_RTC\_CLKSTCTRL is shown in [Figure 8-155](#) and described in [Table 8-166](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-155. CM\_RTC\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-166. CM\_RTC\_CLKSTCTRL Register Field Descriptions**

| Bit   | Field                   | Type | Reset | Description  |
|-------|-------------------------|------|-------|--|
| 31-26 | Reserved                | R    | 0h    |  |
| 25-11 | Reserved                | R    | 0h    |  |
| 10    | Reserved                | R    | 0h    |  |
| 9     | CLKACTIVITY_RTC_32KCLK  | R    | 0h    | This field indicates the state of the 32K RTC clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 8     | CLKACTIVITY_L4_RTC_GCLK | R    | 1h    | This field indicates the state of the L4 RTC clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 7-2   | Reserved                | R    | 0h    |  |
| 1-0   | CLKTRCTRL               | R/W  | 2h    | Controls the clock state transition of the RTC clock domains.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.7 CM\_GFX Registers

[Table 8-167](#) lists the memory-mapped registers for the CM\_GFX. All register offset addresses not listed in [Table 8-167](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-167. CM\_GFX REGISTERS**

| Offset | Acronym                       | Register Name   | Section                            |
|--------|-------------------------------|---|------------------------------------|
| 0h     | CM_GFX_L3_CLKSTCTRL           | This register enables the domain power state transition.<br>It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.7.1</a> |
| 4h     | CM_GFX_GFX_CLKCTRL            | This register manages the GFX clocks.   | <a href="#">Section 8.1.12.7.2</a> |
| Ch     | CM_GFX_L4LS_GFX_CLKSTCTR<br>L | This register enables the domain power state transition.<br>It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.7.3</a> |
| 10h    | CM_GFX_MMUCFG_CLKCTRL         | This register manages the MMU CFG clocks.   | <a href="#">Section 8.1.12.7.4</a> |
| 14h    | CM_GFX_MMUDATA_CLKCTRL        | This register manages the MMU clocks.   | <a href="#">Section 8.1.12.7.5</a> |

### 8.1.12.7.1 CM\_GFX\_L3\_CLKSTCTRL Register (offset = 0h) [reset = 2h]

CM\_GFX\_L3\_CLKSTCTRL is shown in [Figure 8-156](#) and described in [Table 8-168](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-156. CM\_GFX\_L3\_CLKSTCTRL Register**

|          |    |    |    |    |    |                      |                         |
|----------|----|----|----|----|----|----------------------|-------------------------|
| 31       | 30 | 29 | 28 | 27 | 26 | 25                   | 24                      |
| Reserved |    |    |    |    |    | Reserved             |                         |
| R-0h     |    |    |    |    |    | R-0h                 |                         |
| 23       | 22 | 21 | 20 | 19 | 18 | 17                   | 16                      |
| Reserved |    |    |    |    |    | R-0h                 |                         |
| R-0h     |    |    |    |    |    | R-0h                 |                         |
| 15       | 14 | 13 | 12 | 11 | 10 | 9                    | 8                       |
| Reserved |    |    |    |    |    | CLKACTIVITY_GFX_FCLK | CLKACTIVITY_GFX_L3_GCLK |
| R-0h     |    |    |    |    |    | R-0h                 |                         |
| 7        | 6  | 5  | 4  | 3  | 2  | 1                    | 0                       |
| Reserved |    |    |    |    |    | CLKTRCTRL            |                         |
| R-0h     |    |    |    |    |    | R/W-2h               |                         |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-168. CM\_GFX\_L3\_CLKSTCTRL Register Field Descriptions**

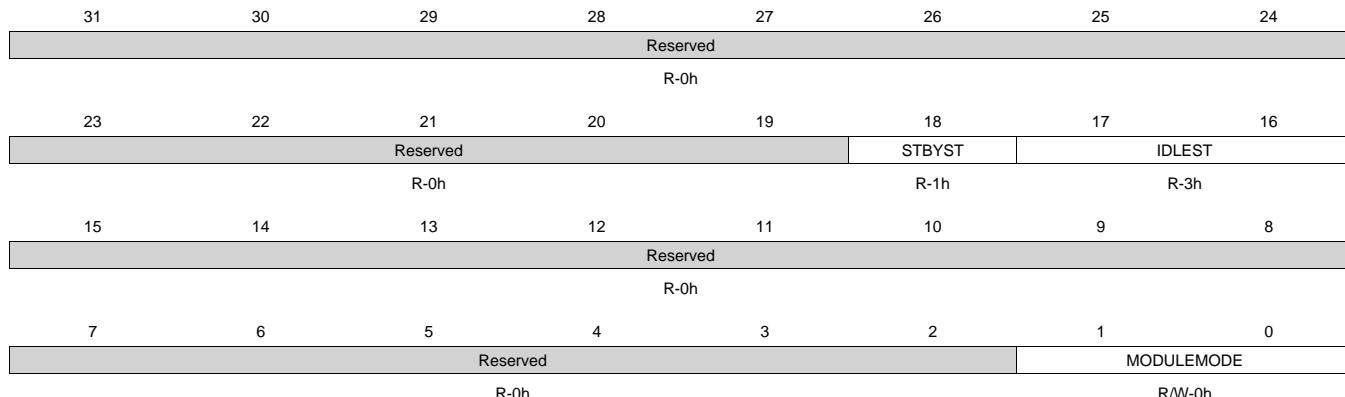
| Bit   | Field                   | Type | Reset | Description   |
|-------|-------------------------|------|-------|---|
| 31-26 | Reserved                | R    | 0h    |   |
| 25-10 | Reserved                | R    | 0h    |   |
| 9     | CLKACTIVITY_GFX_FCLK    | R    | 0h    | This field indicates the state of the GFX_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active  |
| 8     | CLKACTIVITY_GFX_L3_GCLK | R    | 0h    | This field indicates the state of the GFX_L3_GCLK clock in the domain.<br>0x0 = Inact : Corresponding clock is gated<br>0x1 = Act : Corresponding clock is active   |
| 7-2   | Reserved                | R    | 0h    |   |
| 1-0   | CLKTRCTRL               | R/W  | 2h    | Controls the clock state transition of the GFX clock domain in GFX power domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.7.2 CM\_GFX\_GFX\_CLKCTRL Register (offset = 4h) [reset = 70000h]

CM\_GFX\_GFX\_CLKCTRL is shown in [Figure 8-157](#) and described in [Table 8-169](#).

This register manages the GFX clocks.

**Figure 8-157. CM\_GFX\_GFX\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-169. CM\_GFX\_GFX\_CLKCTRL Register Field Descriptions**

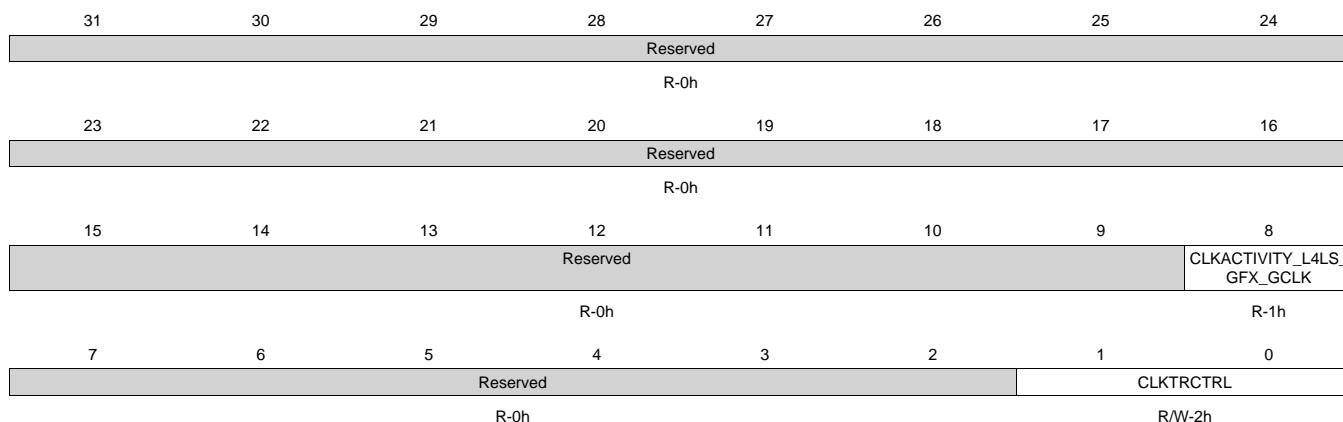
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-19 | Reserved   | R    | 0h    |   |
| 18    | STBYST     | R    | 1h    | Module standby status.<br>0x0 = Func : Module is functional (not in standby)<br>0x1 = Standby : Module is in standby  |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.7.3 CM\_GFX\_L4LS\_GFX\_CLKSTCTRL Register (offset = Ch) [reset = 102h]

CM\_GFX\_L4LS\_GFX\_CLKSTCTRL is shown in [Figure 8-158](#) and described in [Table 8-170](#).

This register enables the domain power state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-158. CM\_GFX\_L4LS\_GFX\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-170. CM\_GFX\_L4LS\_GFX\_CLKSTCTRL Register Field Descriptions**

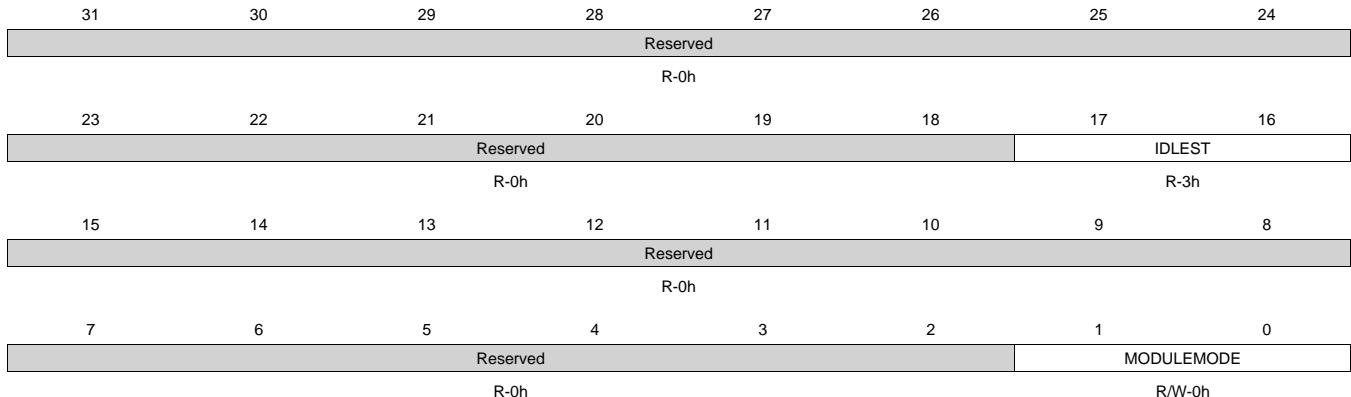
| Bit  | Field                         | Type | Reset | Description  |
|------|-------------------------------|------|-------|--|
| 31-9 | Reserved                      | R    | 0h    |  |
| 8    | CLKACTIVITY_L4LS_GF<br>X_GCLK | R    | 1h    | This field indicates the state of the L4_LS clock in the domain.<br>0x0 = Inact<br>0x1 = Act   |
| 7-2  | Reserved                      | R    | 0h    |  |
| 1-0  | CLKTRCTRL                     | R/W  | 2h    | Controls the clock state transition of the L4LS clock domain in GFX power domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

#### 8.1.12.7.4 CM\_GFX\_MMUCFG\_CLKCTRL Register (offset = 10h) [reset = 30000h]

CM\_GFX\_MMUCFG\_CLKCTRL is shown in [Figure 8-159](#) and described in [Table 8-171](#).

This register manages the MMU CFG clocks.

**Figure 8-159. CM\_GFX\_MMUCFG\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-171. CM\_GFX\_MMUCFG\_CLKCTRL Register Field Descriptions**

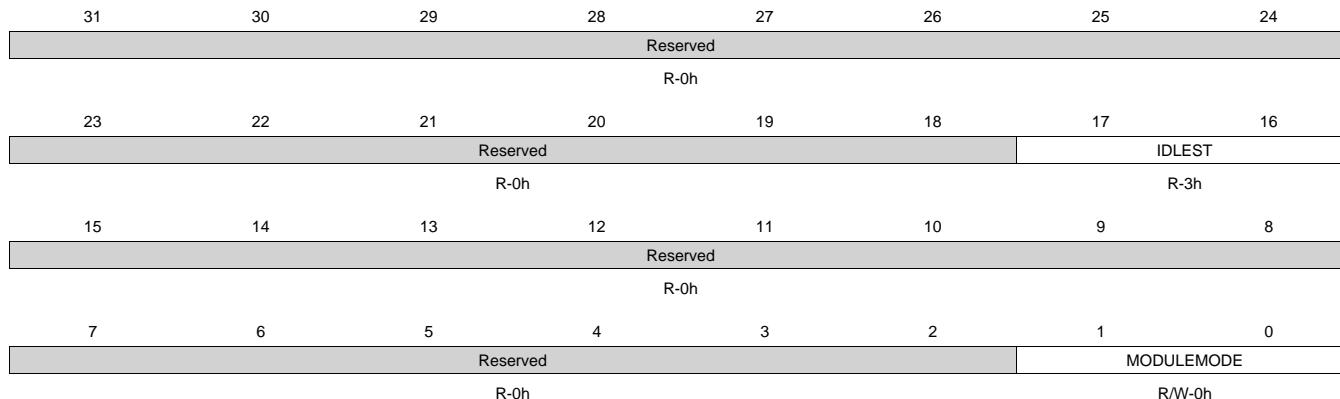
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-18 | Reserved   | R    | 0h    |  |
| 17-16 | IDLEST     | R    | 3h    | <p>Module idle status.</p> <p>0x0 = Func : Module is fully functional, including OCP</p> <p>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion</p> <p>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock</p> <p>0x3 = Disable : Module is disabled and cannot be accessed</p>  |
| 15-2  | Reserved   | R    | 0h    |  |
| 1-0   | MODULEMODE | R/W  | 0h    | <p>Control the way mandatory clocks are managed.</p> <p>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).</p> <p>0x1 = RESERVED_1 : Reserved</p> <p>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.</p> <p>0x3 = RESERVED : Reserved</p> |

### 8.1.12.7.5 CM\_GFX\_MMUDATA\_CLKCTRL Register (offset = 14h) [reset = 30000h]

CM\_GFX\_MMUDATA\_CLKCTRL is shown in [Figure 8-160](#) and described in [Table 8-172](#).

This register manages the MMU clocks.

**Figure 8-160. CM\_GFX\_MMUDATA\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-172. CM\_GFX\_MMUDATA\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

### 8.1.12.8 CM\_CEFUSE Registers

[Table 8-173](#) lists the memory-mapped registers for the CM\_CEFUSE. All register offset addresses not listed in [Table 8-173](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-173. CM\_CEFUSE REGISTERS**

| Offset | Acronym             | Register Name   | Section                            |
|--------|---------------------|---|------------------------------------|
| 0h     | CM_CEFUSE_CLKSTCTRL | This register enables the domain power state transition.<br>It controls the HW supervised domain power state transition between ON-ACTIVE and ON-INACTIVE states.<br>It also hold one status bit per clock input of the domain. | <a href="#">Section 8.1.12.8.1</a> |

**Table 8-173. CM\_CEFUSE REGISTERS (continued)**

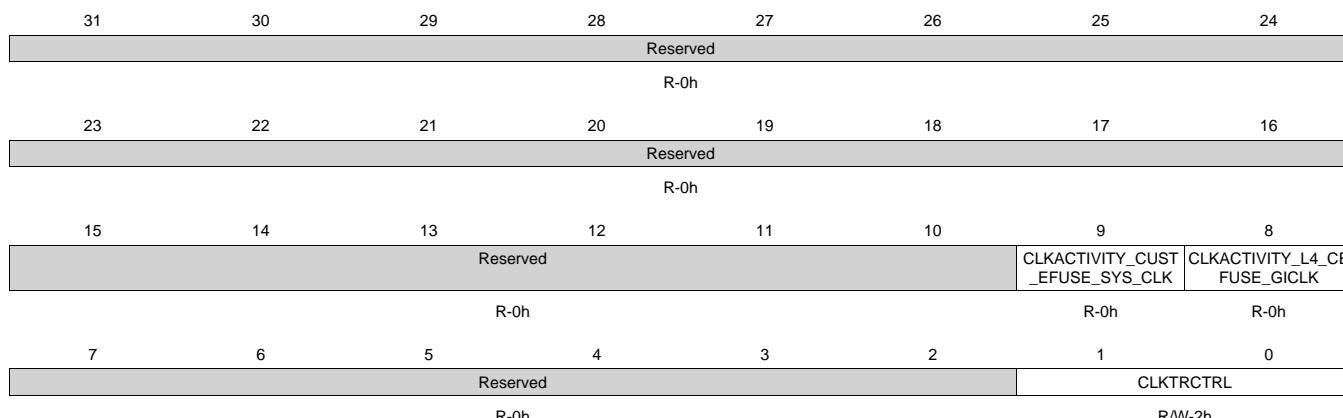
| Offset   | Acronym                 | Register Name                            | Section                            |
|----------|-------------------------|--|------------------------------------|
| 20h<br>L | CM_CEFUSE_CEFUSE_CLKCTR | This register manages the CEFUSE clocks. | <a href="#">Section 8.1.12.8.2</a> |

### 8.1.12.8.1 CM\_CEFUSE\_CLKSTCTRL Register (offset = 0h) [reset = 2h]

CM\_CEFUSE\_CLKSTCTRL is shown in [Figure 8-161](#) and described in [Table 8-174](#).

This register enables the domain power state transition. It controls the HW supervised domain power state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.

**Figure 8-161. CM\_CEFUSE\_CLKSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-174. CM\_CEFUSE\_CLKSTCTRL Register Field Descriptions**

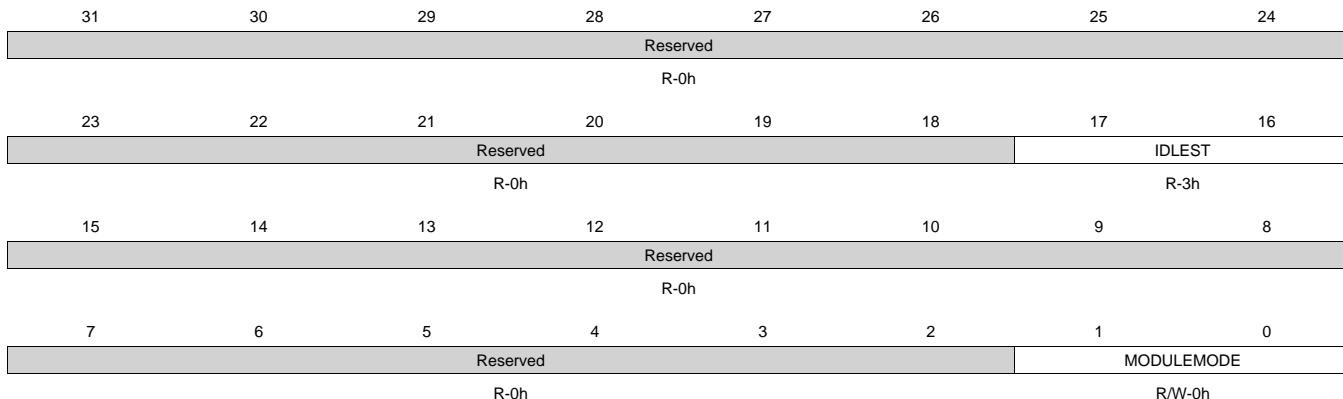
| Bit   | Field                              | Type | Reset | Description  |
|-------|------------------------------------|------|-------|--|
| 31-10 | Reserved                           | R    | 0h    |  |
| 9     | CLKACTIVITY_CUST_EF<br>USE_SYS_CLK | R    | 0h    | This field indicates the state of the Cust_Efuse_SYSCLK clock input of the domain.<br>[warm reset insensitive]<br>0x0 = Inact : Corresponding clock is definitely gated<br>0x1 = Act : Corresponding clock is running or gating/ungating transition is on-going  |
| 8     | CLKACTIVITY_L4_CEFU<br>SE_GICLK    | R    | 0h    | This field indicates the state of the L4_CEFUSE_GCLK clock input of the domain.<br>[warm reset insensitive]<br>0x0 = Inact : Corresponding clock is definitely gated<br>0x1 = Act : Corresponding clock is running or gating/ungating transition is on-going   |
| 7-2   | Reserved                           | R    | 0h    |  |
| 1-0   | CLKTRCTRL                          | R/W  | 2h    | Controls the clock state transition of the clock domain in customer efuse power domain.<br>0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur.<br>0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain.<br>0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain.<br>0x3 = Reserved : Reserved. |

### 8.1.12.8.2 CM\_CEFUSE\_CEFUSE\_CLKCTRL Register (offset = 20h) [reset = 30000h]

CM\_CEFUSE\_CEFUSE\_CLKCTRL is shown in Figure 8-162 and described in Table 8-175.

This register manages the CEFUSE clocks.

**Figure 8-162. CM\_CEFUSE\_CEFUSE\_CLKCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-175. CM\_CEFUSE\_CEFUSE\_CLKCTRL Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31-18 | Reserved   | R    | 0h    |   |
| 17-16 | IDLEST     | R    | 3h    | Module idle status.<br>[warm reset insensitive]<br>0x0 = Func : Module is fully functional, including OCP<br>0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion<br>0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock<br>0x3 = Disable : Module is disabled and cannot be accessed  |
| 15-2  | Reserved   | R    | 0h    |   |
| 1-0   | MODULEMODE | R/W  | 0h    | Control the way mandatory clocks are managed.<br>0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup).<br>0x1 = RESERVED_1 : Reserved<br>0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guaranteed to stay present. As long as in this configuration, power domain sleep transition cannot happen.<br>0x3 = RESERVED : Reserved |

## **8.1.13 Power Management Registers**

### **8.1.13.1 PRM\_IRQ Registers**

**Table 8-176** lists the memory-mapped registers for the PRM\_IRQ. All register offset addresses not listed in **Table 8-176** should be considered as reserved locations and the register contents should not be modified.

**Table 8-176. PRM\_IRQ REGISTERS**

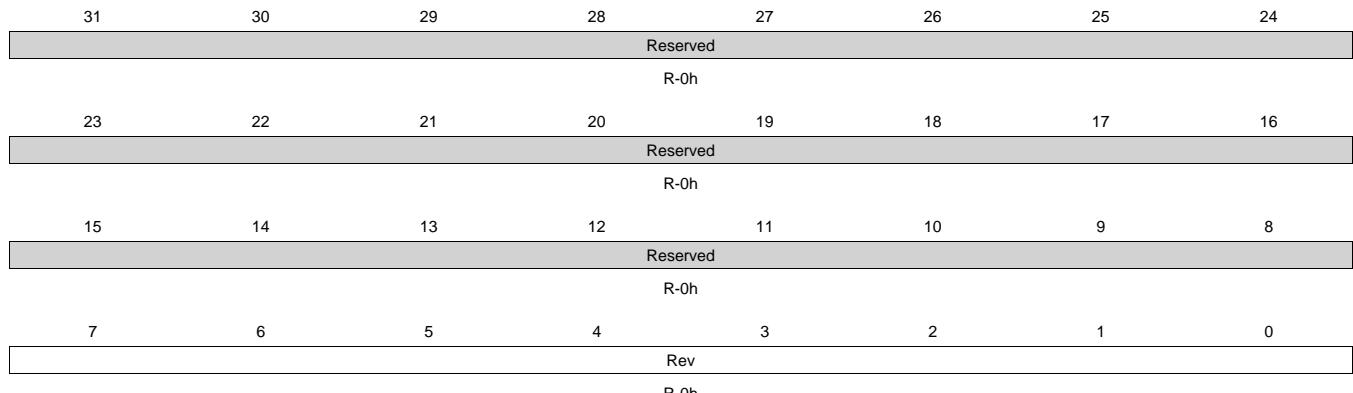
| Offset | Acronym           | Register Name   | Section                            |
|--------|-------------------|---|------------------------------------|
| 0h     | REVISION_PRM      | This register contains the IP revision code for the PRCM  | <a href="#">Section 8.1.13.1.1</a> |
| 4h     | PRM_IRQSTATUS_MPU | This register provides status on MPU interrupt events.<br>An event is logged whether interrupt generation for the event is enabled or not.<br>SW is required to clear a set bit by writing a '1' into the bit-position to be cleared. | <a href="#">Section 8.1.13.1.2</a> |
| 8h     | PRM_IRQENABLE_MPU | This register is used to enable and disable events used to trigger MPU interrupt activation.  | <a href="#">Section 8.1.13.1.3</a> |
| Ch     | PRM_IRQSTATUS_M3  | This register provides status on MPU interrupt events.<br>An event is logged whether interrupt generation for the event is enabled or not.<br>SW is required to clear a set bit by writing a '1' into the bit-position to be cleared. | <a href="#">Section 8.1.13.1.4</a> |
| 10h    | PRM_IRQENABLE_M3  | This register is used to enable and disable events used to trigger MPU interrupt activation.  | <a href="#">Section 8.1.13.1.5</a> |

### 8.1.13.1.1 REVISION\_PRM Register (offset = 0h) [reset = 0h]

REVISION\_PRM is shown in [Figure 8-163](#) and described in [Table 8-177](#).

This register contains the IP revision code for the PRCM

**Figure 8-163. REVISION\_PRM Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-177. REVISION\_PRM Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    | Reads returns 0.   |
| 7-0  | Rev      | R    | 0h    | IP revision [<br>7:4] Major revision [<br>3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1 |

### 8.1.13.1.2 PRM\_IRQSTATUS\_MPU Register (offset = 4h) [reset = 0h]

PRM\_IRQSTATUS\_MPU is shown in [Figure 8-164](#) and described in [Table 8-178](#).

This register provides status on MPU interrupt events. An event is logged whether interrupt generation for the event is enabled or not. SW is required to clear a set bit by writing a '1' into the bit-position to be cleared.

**Figure 8-164. PRM\_IRQSTATUS\_MPU Register**

|                   |                   |                    |                    |                   |              |          |               |
|-------------------|-------------------|--------------------|--------------------|-------------------|--------------|----------|---------------|
| 31                | 30                | 29                 | 28                 | 27                | 26           | 25       | 24            |
| Reserved          |                   |                    |                    |                   |              |          |               |
| R-0h              |                   |                    |                    |                   |              |          |               |
| 23                | 22                | 21                 | 20                 | 19                | 18           | 17       | 16            |
| Reserved          |                   |                    |                    |                   |              |          |               |
| R-0h              |                   |                    |                    |                   |              |          |               |
| 15                | 14                | 13                 | 12                 | 11                | 10           | 9        | 8             |
| dpll_per_recal_st | dpll_ddr_recal_st | dpll_disp_recal_st | dpll_core_recal_st | dpll_mpu_recal_st | ForceWkup_st | Reserved | Transition_st |
| R/W-0h            | R/W-0h            | R/W-0h             | R/W-0h             | R/W-0h            | R/W-0h       | R-0h     | R/W-0h        |
| 7                 | 6                 | 5                  | 4                  | 3                 | 2            | 1        | 0             |
| Reserved          |                   |                    |                    |                   |              |          |               |
| R-0h              |                   |                    |                    |                   |              |          |               |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-178. PRM\_IRQSTATUS\_MPU Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-16 | Reserved           | R    | 0h    |  |
| 15    | dpll_per_recal_st  | R/W  | 0h    | interrupt status for usb dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 14    | dpll_ddr_recal_st  | R/W  | 0h    | interrupt status for ddr dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 13    | dpll_disp_recal_st | R/W  | 0h    | interrupt status for disp dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                     |
| 12    | dpll_core_recal_st | R/W  | 0h    | interrupt status for core dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                     |
| 11    | dpll_mpu_recal_st  | R/W  | 0h    | interrupt status for mpu dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 10    | ForceWkup_st       | R/W  | 0h    | Software supervised wakeup completed event interrupt status<br>0 = IRQ_fal : No interrupt<br>1 = IRQ_tru : Interrupt is pending                  |
| 9     | Reserved           | R    | 0h    |  |
| 8     | Transition_st      | R/W  | 0h    | Software supervised transition completed event interrupt status (any domain)<br>0 = IRQ_fal : No interrupt<br>1 = IRQ_tru : Interrupt is pending |
| 7-0   | Reserved           | R    | 0h    |  |

### 8.1.13.1.3 PRM\_IRQENABLE\_MPU Register (offset = 8h) [reset = 0h]

PRM\_IRQENABLE\_MPU is shown in [Figure 8-165](#) and described in [Table 8-179](#).

This register is used to enable and disable events used to trigger MPU interrupt activation.

**Figure 8-165. PRM\_IRQENABLE\_MPU Register**

| 31                 | 30                | 29                | 28                 | 27                | 26           | 25       | 24            |
|--------------------|-------------------|-------------------|--------------------|-------------------|--------------|----------|---------------|
| Reserved           |                   |                   |                    |                   |              |          |               |
| R-0h               |                   |                   |                    |                   |              |          |               |
| 23                 | 22                | 21                | 20                 | 19                | 18           | 17       | 16            |
| Reserved           |                   |                   |                    |                   |              |          |               |
| R-0h               |                   |                   |                    |                   |              |          |               |
| 15                 | 14                | 13                | 12                 | 11                | 10           | 9        | 8             |
| dpll_disp_recal_en | dpll_ddr_recal_en | dpll_per_recal_en | dpll_core_recal_en | dpll_mpu_recal_en | ForceWkup_en | Reserved | Transition_en |
| R/W-0h             | R/W-0h            | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h       | R-0h     | R/W-0h        |
| 7                  | 6                 | 5                 | 4                  | 3                 | 2            | 1        | 0             |
| Reserved           |                   | Reserved          |                    |                   |              | Reserved |               |
| R-0h               |                   |                   |                    |                   |              |          |               |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-179. PRM\_IRQENABLE\_MPU Register Field Descriptions**

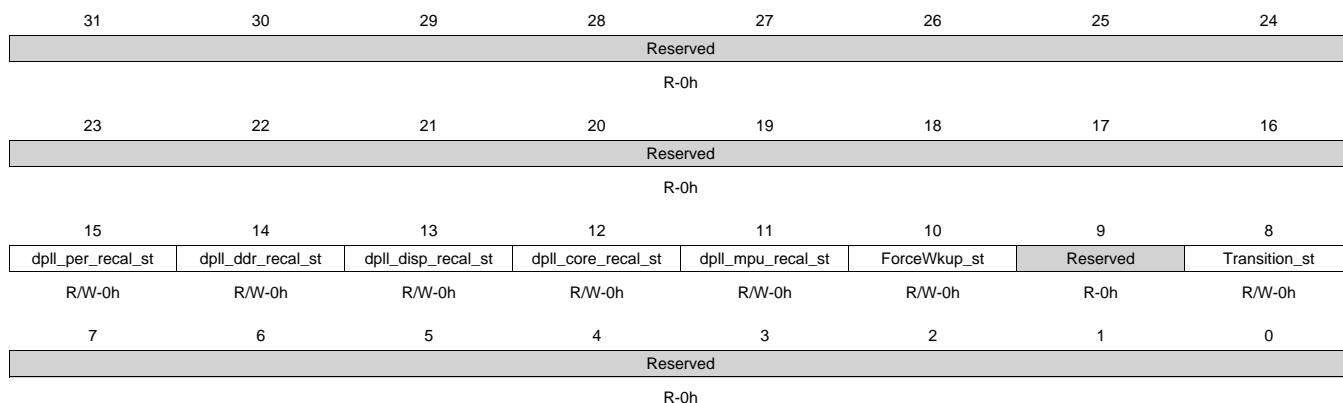
| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-16 | Reserved           | R    | 0h    |  |
| 15    | dpll_disp_recal_en | R/W  | 0h    | Interrupt enable for disp dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                           |
| 14    | dpll_ddr_recal_en  | R/W  | 0h    | Interrupt enable for ddr dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 13    | dpll_per_recal_en  | R/W  | 0h    | Interrupt enable for usb dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 12    | dpll_core_recal_en | R/W  | 0h    | Interrupt enable for core dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                           |
| 11    | dpll_mpu_recal_en  | R/W  | 0h    | Interrupt enable for mpu dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 10    | ForceWkup_en       | R/W  | 0h    | Software supervised Froce Wakeup completed event interrupt enable<br>0 = irq_msk : Interrupt is masked<br>1 = irq_en : Interrupt is enabled            |
| 9     | Reserved           | R    | 0h    |  |
| 8     | Transition_en      | R/W  | 0h    | Software supervised transition completed event interrupt enable (any domain)<br>0 = irq_msk : Interrupt is masked<br>1 = irq_en : Interrupt is enabled |
| 7-6   | Reserved           | R    | 0h    |  |
| 5-1   | Reserved           | R    | 0h    |  |
| 0     | Reserved           | R    | 0h    |  |

#### 8.1.13.1.4 PRM\_IRQSTATUS\_M3 Register (offset = Ch) [reset = 0h]

PRM\_IRQSTATUS\_M3 is shown in [Figure 8-166](#) and described in [Table 8-180](#).

This register provides status on MPU interrupt events. An event is logged whether interrupt generation for the event is enabled or not. SW is required to clear a set bit by writing a '1' into the bit-position to be cleared.

**Figure 8-166. PRM\_IRQSTATUS\_M3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-180. PRM\_IRQSTATUS\_M3 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-16 | Reserved           | R    | 0h    |  |
| 15    | dpll_per_recal_st  | R/W  | 0h    | interrupt status for usb dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 14    | dpll_ddr_recal_st  | R/W  | 0h    | interrupt status for ddr dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 13    | dpll_disp_recal_st | R/W  | 0h    | interrupt status for disp dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                     |
| 12    | dpll_core_recal_st | R/W  | 0h    | interrupt status for core dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                     |
| 11    | dpll_mpu_recal_st  | R/W  | 0h    | interrupt status for mpu dpll recalibration<br>0 = DIS : Disables dpll recalibration<br>1 = EN : ENAbles dpll recalibration                      |
| 10    | ForceWkup_st       | R/W  | 0h    | Software supervised wakeup completed event interrupt status<br>0 = IRQ_fal : No interrupt<br>1 = IRQ_tru : Interrupt is pending                  |
| 9     | Reserved           | R    | 0h    |  |
| 8     | Transition_st      | R/W  | 0h    | Software supervised transition completed event interrupt status (any domain)<br>0 = IRQ_fal : No interrupt<br>1 = IRQ_tru : Interrupt is pending |
| 7-0   | Reserved           | R    | 0h    |  |

### 8.1.13.1.5 PRM\_IRQENABLE\_M3 Register (offset = 10h) [reset = 0h]

PRM\_IRQENABLE\_M3 is shown in [Figure 8-167](#) and described in [Table 8-181](#).

This register is used to enable and disable events used to trigger MPU interrupt activation.

**Figure 8-167. PRM\_IRQENABLE\_M3 Register**

| 31                 | 30                | 29                | 28                 | 27                | 26           | 25       | 24            |
|--------------------|-------------------|-------------------|--------------------|-------------------|--------------|----------|---------------|
| Reserved           |                   |                   |                    |                   |              |          |               |
| R-0h               |                   |                   |                    |                   |              |          |               |
| 23                 | 22                | 21                | 20                 | 19                | 18           | 17       | 16            |
| Reserved           |                   |                   |                    |                   |              |          |               |
| R-0h               |                   |                   |                    |                   |              |          |               |
| 15                 | 14                | 13                | 12                 | 11                | 10           | 9        | 8             |
| dpll_disp_recal_en | dpll_ddr_recal_en | dpll_per_recal_en | dpll_core_recal_en | dpll_mpu_recal_en | ForceWkup_en | Reserved | Transition_en |
| R/W-0h             | R/W-0h            | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h       | R-0h     | R/W-0h        |
| 7                  | 6                 | 5                 | 4                  | 3                 | 2            | 1        | 0             |
| Reserved           |                   | Reserved          |                    |                   |              | Reserved |               |
| R-0h               |                   |                   |                    |                   |              |          |               |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-181. PRM\_IRQENABLE\_M3 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-16 | Reserved           | R    | 0h    |  |
| 15    | dpll_disp_recal_en | R/W  | 0h    | Interrupt enable for disp dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                           |
| 14    | dpll_ddr_recal_en  | R/W  | 0h    | Interrupt enable for ddr dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 13    | dpll_per_recal_en  | R/W  | 0h    | Interrupt enable for usb dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 12    | dpll_core_recal_en | R/W  | 0h    | Interrupt enable for core dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                           |
| 11    | dpll_mpu_recal_en  | R/W  | 0h    | Interrupt enable for mpu dppl recalibration<br>0 = DIS : Disables dppl recalibration<br>1 = EN : ENAbles dppl recalibration                            |
| 10    | ForceWkup_en       | R/W  | 0h    | Software supervised Froce Wakeup completed event interrupt enable<br>0 = irq_msk : Interrupt is masked<br>1 = irq_en : Interrupt is enabled            |
| 9     | Reserved           | R    | 0h    |  |
| 8     | Transition_en      | R/W  | 0h    | Software supervised transition completed event interrupt enable (any domain)<br>0 = irq_msk : Interrupt is masked<br>1 = irq_en : Interrupt is enabled |
| 7-6   | Reserved           | R    | 0h    |  |
| 5-1   | Reserved           | R    | 0h    |  |
| 0     | Reserved           | R    | 0h    |  |

### **8.1.13.2 PRM\_PER Registers**

**Table 8-182** lists the memory-mapped registers for the PRM\_PER. All register offset addresses not listed in [Table 8-182](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-182. PRM\_PER REGISTERS**

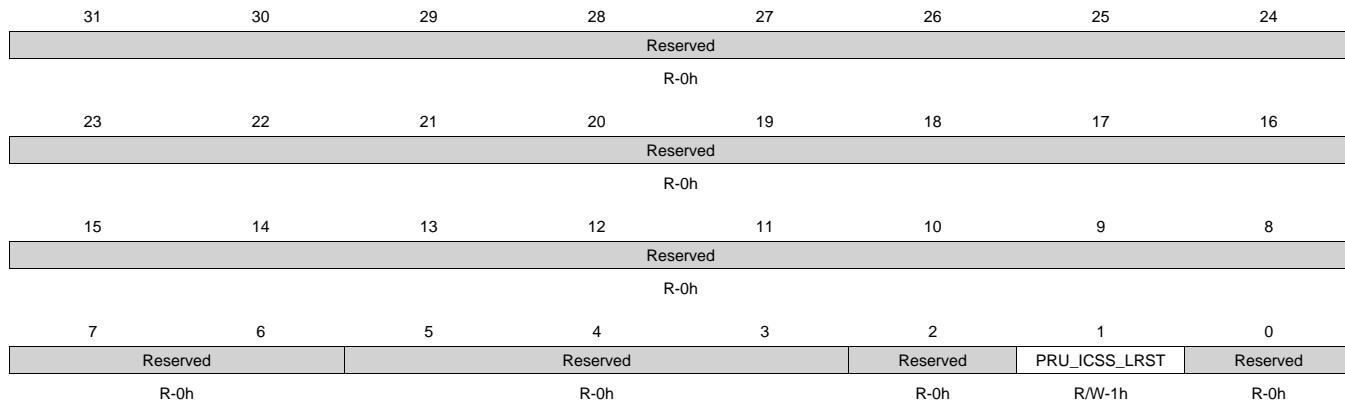
| Offset | Acronym          | Register Name  | Section                            |
|--------|------------------|--|------------------------------------|
| 0h     | RM_PER_RSTCTRL   | This register controls the release of the PER Domain resets.                                       | <a href="#">Section 8.1.13.2.1</a> |
| 8h     | PM_PER_PWRSTST   | This register provides a status on the current PER power domain state.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.2.2</a> |
| Ch     | PM_PER_PWRSTCTRL | Controls the power state of PER power domain   | <a href="#">Section 8.1.13.2.3</a> |

### 8.1.13.2.1 RM\_PER\_RSTCTRL Register (offset = 0h) [reset = 2h]

RM\_PER\_RSTCTRL is shown in [Figure 8-168](#) and described in [Table 8-183](#).

This register controls the release of the PER Domain resets.

**Figure 8-168. RM\_PER\_RSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-183. RM\_PER\_RSTCTRL Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-6 | Reserved      | R    | 0h    |   |
| 5-3  | Reserved      | R    | 0h    |   |
| 2    | Reserved      | R    | 0h    |   |
| 1    | PRU_ICSS_LRST | R/W  | 1h    | PER domain PRU-ICSS local reset control<br>0x0 = CLEAR : Reset is cleared for the PRU-ICSS<br>0x1 = ASSERT : Reset is asserted for the PRU-ICSS |
| 0    | Reserved      | R    | 0h    |   |

### 8.1.13.2.2 PM\_PER\_PWRSTST Register (offset = 8h) [reset = 1E60007h]

PM\_PER\_PWRSTST is shown in [Figure 8-169](#) and described in [Table 8-184](#).

This register provides a status on the current PER power domain state. [warm reset insensitive]

**Figure 8-169. PM\_PER\_PWRSTST Register**

| 31                   | 30              | 29           | 28       | 27              | 26           | 25 | 24                   |
|----------------------|-----------------|--------------|----------|-----------------|--------------|----|----------------------|
| Reserved             |                 |              |          |                 |              |    | pru_icss_mem_statest |
| R-0h                 |                 |              |          |                 |              |    | R-3h                 |
| 23                   | 22              | 21           | 20       | 19              | 18           | 17 | 16                   |
| pru_icss_mem_statest | ram_mem_statest | InTransition | Reserved | PER_mem_statest | Reserved     |    |                      |
| R-3h                 | R-3h            | R-0h         | R-0h     | R-3h            | R-3h         |    | R-0h                 |
| 15                   | 14              | 13           | 12       | 11              | 10           | 9  | 8                    |
| Reserved             |                 |              |          |                 |              |    | R-0h                 |
| 7                    | 6               | 5            | 4        | 3               | 2            | 1  | 0                    |
| Reserved             |                 |              |          | LogicStateSt    | PowerStateSt |    |                      |
| R-0h                 |                 |              |          | R-1h            | R-3h         |    |                      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-184. PM\_PER\_PWRSTST Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31-25 | Reserved             | R    | 0h    |  |
| 24-23 | pru_icss_mem_statest | R    | 3h    | PRU-ICSS memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                |
| 22-21 | ram_mem_statest      | R    | 3h    | OCMC RAM memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                |
| 20    | InTransition         | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress. |
| 19    | Reserved             | R    | 0h    |  |
| 18-17 | PER_mem_statest      | R    | 3h    | PER domain memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON              |
| 16-3  | Reserved             | R    | 0h    |  |
| 2     | LogicStateSt         | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON   |
| 1-0   | PowerStateSt         | R    | 3h    | Current Power State Status<br>0x0 = OFF : OFF State<br>0x1 = RET<br>0x2 = Reserved1<br>0x3 = ON : ON State                               |

### 8.1.13.2.3 PM\_PER\_PWRSTCTRL Register (offset = Ch) [reset = EE0000EBh]

PM\_PER\_PWRSTCTRL is shown in [Figure 8-170](#) and described in [Table 8-185](#).

Controls the power state of PER power domain

**Figure 8-170. PM\_PER\_PWRSTCTRL Register**

| 31                    | 30                   | 29                  | 28               | 27              | 26       | 25         | 24 |
|-----------------------|----------------------|---------------------|------------------|-----------------|----------|------------|----|
| ram_mem_ONState       | PER_mem_RETState     | Reserved            | ram_mem_RETState | PER_mem_ONState | Reserved |            |    |
| R/W-3h                | R/W-1h               | R-0h                | R/W-1h           | R/W-3h          | R-0h     |            |    |
| 23                    | 22                   | 21                  | 20               | 19              | 18       | 17         | 16 |
| Reserved              |                      |                     |                  |                 |          |            |    |
| R-0h                  |                      |                     |                  |                 |          |            |    |
| 15                    | 14                   | 13                  | 12               | 11              | 10       | 9          | 8  |
| Reserved              |                      |                     |                  |                 |          |            |    |
| R-0h                  |                      |                     |                  |                 |          |            |    |
| 7                     | 6                    | 5                   | 4                | 3               | 2        | 1          | 0  |
| pru_icss_mem_RETState | pru_icss_mem_ONState | LowPowerStateChange | LogicRETState    | Reserved        | Reserved | PowerState |    |
| R/W-1h                | R/W-3h               | R/W-0h              | R/W-1h           | R-0h            | R-0h     | R/W-3h     |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-185. PM\_PER\_PWRSTCTRL Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description  |
|-------|-----------------------|------|-------|--|
| 31-30 | ram_mem_ONState       | R/W  | 3h    | OCMC RAM memory on state<br>0x0 = OFF : Memory is OFF<br>0x1 = RET : Memory is in retention state<br>0x2 = RESERVED<br>0x3 = ON : Memory is ON |
| 29    | PER_mem_RETState      | R/W  | 1h    | Other memories in PER Domain RET state<br>0x0 = OFF<br>0x1 = RET   |
| 28    | Reserved              | R    | 0h    |  |
| 27    | ram_mem_RETState      | R/W  | 1h    | OCMC RAM memory RET state<br>0x0 = OFF : Memory is in off state<br>0x1 = RET : Memory is in retention state                                    |
| 26-25 | PER_mem_ONState       | R/W  | 3h    | Other memories in PER Domain ON state<br>0x0 = Reserved2<br>0x1 = Reserved1<br>0x2 = Reserved : Reserved<br>0x3 = ON : Memory is ON            |
| 24-8  | Reserved              | R    | 0h    |  |
| 7     | pru_icss_mem_RETState | R/W  | 1h    | PRU-ICSS memory RET state<br>0x0 = OFF : Memory is in off state<br>0x1 = RET : Memory is in retention state                                    |
| 6-5   | pru_icss_mem_ONState  | R/W  | 3h    | PRU-ICSS memory ON state<br>0x0 = Reserved2<br>0x1 = Reserved1<br>0x2 = Reserved : Reserved<br>0x3 = ON : Memory is ON                         |

**Table 8-185. PM\_PER\_PWRSTCTRL Register Field Descriptions (continued)**

| Bit | Field               | Type | Reset | Description   |
|-----|---------------------|------|-------|---|
| 4   | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3   | LogicRETState       | R/W  | 1h    | Logic state when power domain is RETENTION<br>0x0 = logic_off : Only retention registers are retained and remaining logic is off when the domain is in RETENTION state.<br>0x1 = logic_ret : Whole logic is retained when domain is in RETENTION state.   |
| 2   | Reserved            | R    | 0h    |   |
| 1-0 | PowerState          | R/W  | 3h    | PER domain power state control<br>0x0 = OFF<br>0x1 = RET<br>0x2 = Reserved<br>0x3 = ON  |

### 8.1.13.3 PRM\_WKUP Registers

**Table 8-186** lists the memory-mapped registers for the PRM\_WKUP. All register offset addresses not listed in **Table 8-186** should be considered as reserved locations and the register contents should not be modified.

**Table 8-186. PRM\_WKUP REGISTERS**

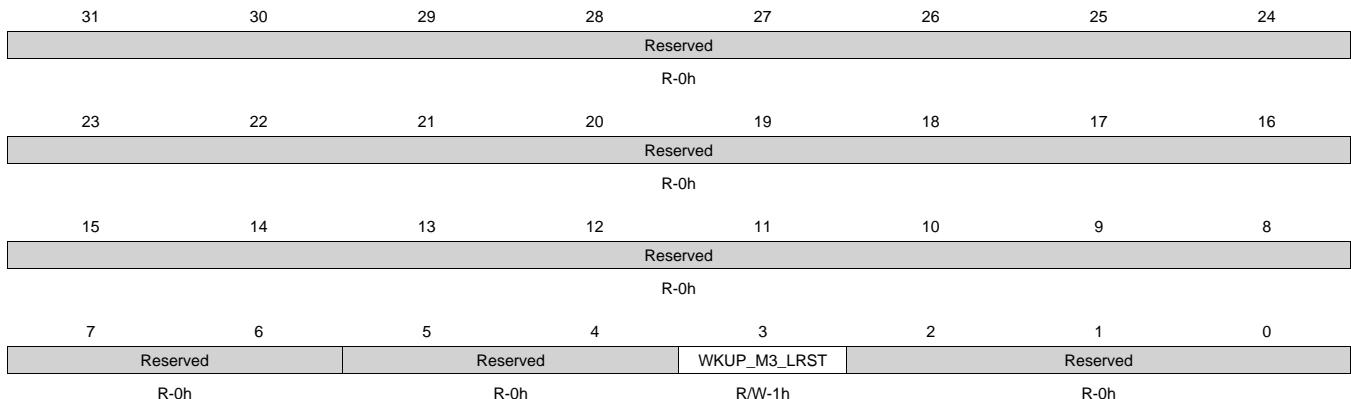
| Offset | Acronym           | Register Name   | Section                            |
|--------|-------------------|---|------------------------------------|
| 0h     | RM_WKUP_RSTCTRL   | This register controls the release of the ALWAYS ON Domain resets.  | <a href="#">Section 8.1.13.3.1</a> |
| 4h     | PM_WKUP_PWRSTCTRL | Controls power state of WKUP power domain   | <a href="#">Section 8.1.13.3.2</a> |
| 8h     | PM_WKUP_PWRSTST   | This register provides a status on the current WKUP power domain state.<br>[warm reset insensitive]   | <a href="#">Section 8.1.13.3.3</a> |
| Ch     | RM_WKUP_RSTST     | This register logs the different reset sources of the ALWON domain.<br>Each bit is set upon release of the domain reset signal.<br>Must be cleared by software.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.3.4</a> |

### 8.1.13.3.1 RM\_WKUP\_RSTCTRL Register (offset = 0h) [reset = 8h]

RM\_WKUP\_RSTCTRL is shown in [Figure 8-171](#) and described in [Table 8-187](#).

This register controls the release of the ALWAYS ON Domain resets.

**Figure 8-171. RM\_WKUP\_RSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-187. RM\_WKUP\_RSTCTRL Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-6 | Reserved     | R    | 0h    |   |
| 5-4  | Reserved     | R    | 0h    |   |
| 3    | WKUP_M3_LRST | R/W  | 1h    | Assert Reset to WKUP_M3<br>0x0 = CLEAR : Reset is cleared for the M3<br>0x1 = ASSERT : Reset is asserted for the M3 by A8 |
| 2-0  | Reserved     | R    | 0h    |   |

### 8.1.13.3.2 PM\_WKUP\_PWRSTCTRL Register (offset = 4h) [reset = 8h]

PM\_WKUP\_PWRSTCTRL is shown in [Figure 8-172](#) and described in [Table 8-188](#).

Controls power state of WKUP power domain

**Figure 8-172. PM\_WKUP\_PWRSTCTRL Register**

| 31       | 30       | 29                      | 28            | 27       | 26       | 25       | 24       |
|----------|----------|-------------------------|---------------|----------|----------|----------|----------|
| Reserved | Reserved | Reserved                | Reserved      | Reserved | Reserved | Reserved | Reserved |
| R-0h     | R-0h     | R-0h                    | R-0h          | R-0h     | R-0h     | R-0h     | R-0h     |
| 23       | 22       | 21                      | 20            | 19       | 18       | 17       | 16       |
| Reserved |          |                         |               |          |          |          |          |
| R-0h     |          |                         |               |          |          |          |          |
| 15       | 14       | 13                      | 12            | 11       | 10       | 9        | 8        |
| Reserved |          |                         |               |          |          |          |          |
| R-0h     |          |                         |               |          |          |          |          |
| 7        | 6        | 5                       | 4             | 3        | 2        | 1        | 0        |
| Reserved |          | LowPowerStateChang<br>e | LogicRETState | Reserved | Reserved | Reserved | Reserved |
| R-0h     |          | R/W-0h                  | R/W-1h        | R-0h     | R-0h     | R-0h     | R-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-188. PM\_WKUP\_PWRSTCTRL Register Field Descriptions**

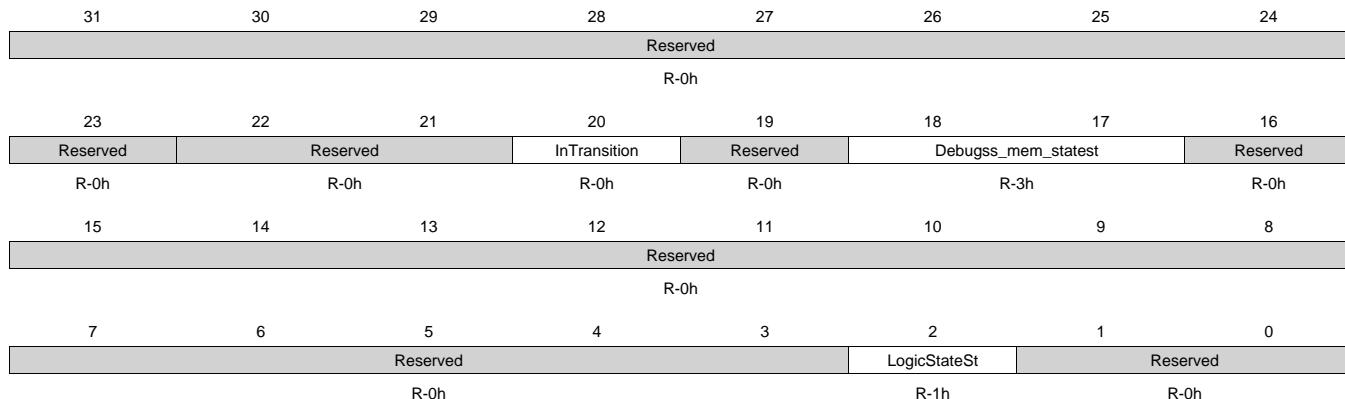
| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-30 | Reserved            | R    | 0h    |   |
| 29    | Reserved            | R    | 0h    |   |
| 28    | Reserved            | R    | 0h    |   |
| 27    | Reserved            | R    | 0h    |   |
| 26-25 | Reserved            | R    | 0h    |   |
| 24-5  | Reserved            | R    | 0h    |   |
| 4     | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3     | LogicRETState       | R/W  | 1h    | Logic state when power domain is RETENTION<br>0x0 = logic_off : Only retention registers are retained and remaining logic is off when the domain is in RETENTION state.<br>0x1 = logic_ret : Whole logic is retained when domain is in RETENTION state.   |
| 2     | Reserved            | R    | 0h    |   |
| 1-0   | Reserved            | R    | 0h    |   |

### 8.1.13.3.3 PM\_WKUP\_PWRSTST Register (offset = 8h) [reset = 60004h]

PM\_WKUP\_PWRSTST is shown in [Figure 8-173](#) and described in [Table 8-189](#).

This register provides a status on the current WKUP power domain state. [warm reset insensitive]

**Figure 8-173. PM\_WKUP\_PWRSTST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-189. PM\_WKUP\_PWRSTST Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description  |
|-------|---------------------|------|-------|--|
| 31-23 | Reserved            | R    | 0h    |  |
| 22-21 | Reserved            | R    | 0h    |  |
| 20    | InTransition        | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress. |
| 19    | Reserved            | R    | 0h    |  |
| 18-17 | Debugss_mem_statest | R    | 3h    | WKUP domain memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON             |
| 16-3  | Reserved            | R    | 0h    |  |
| 2     | LogicStateSt        | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON   |
| 1-0   | Reserved            | R    | 0h    |  |

#### 8.1.13.3.4 RM\_WKUP\_RSTST Register (offset = Ch) [reset = 0h]

RM\_WKUP\_RSTST is shown in [Figure 8-174](#) and described in [Table 8-190](#).

This register logs the different reset sources of the ALWON domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]

**Figure 8-174. RM\_WKUP\_RSTST Register**

|                   |                  |              |    |          |    |    |    |
|-------------------|------------------|--------------|----|----------|----|----|----|
| 31                | 30               | 29           | 28 | 27       | 26 | 25 | 24 |
| Reserved          |                  |              |    |          |    |    |    |
| R-0h              |                  |              |    |          |    |    |    |
| 23                | 22               | 21           | 20 | 19       | 18 | 17 | 16 |
| Reserved          |                  |              |    |          |    |    |    |
| R-0h              |                  |              |    |          |    |    |    |
| 15                | 14               | 13           | 12 | 11       | 10 | 9  | 8  |
| Reserved          |                  |              |    |          |    |    |    |
| R-0h              |                  |              |    |          |    |    |    |
| 7                 | 6                | 5            | 4  | 3        | 2  | 1  | 0  |
| ICECRUSHER_M3_RST | EMULATION_M3_RST | WKUP_M3_LRST |    | Reserved |    |    |    |
| R/W-0h            | R/W-0h           | R/W-0h       |    | R-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-190. RM\_WKUP\_RSTST Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 31-8 | Reserved          | R    | 0h    |  |
| 7    | ICECRUSHER_M3_RST | R/W  | 0h    | M3 Processor has been reset due to M3 ICECRUSHER1 reset event<br>0x0 = RESET_NO : No reset<br>0x1 = RESET_YES : M3 Processor has been reset  |
| 6    | EMULATION_M3_RST  | R/W  | 0h    | M3 Processor has been reset due to emulation reset source e.g.<br>assert reset command initiated by the icepick module<br>0x0 = RESET_NO : No reset<br>0x1 = RESET_YES : M3 Processor has been reset |
| 5    | WKUP_M3_LRST      | R/W  | 0h    | M3 Processor has been reset<br>0x0 = RESET_NO : No reset<br>0x1 = RESET_YES : M3 Processor has been reset  |
| 4-0  | Reserved          | R    | 0h    |  |

#### 8.1.13.4 PRM\_MPUs Registers

[Table 8-191](#) lists the memory-mapped registers for the PRM\_MPUs. All register offset addresses not listed in [Table 8-191](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-191. PRM\_MPUs REGISTERS**

| Offset | Acronym           | Register Name   | Section                            |
|--------|-------------------|---|------------------------------------|
| 0h     | PM_MPUs_PWRSTCTRL | This register controls the MPUs power state to reach upon mpu domain sleep transition   | <a href="#">Section 8.1.13.4.1</a> |
| 4h     | PM_MPUs_PWRSTST   | This register provides a status on the current MPUs power domain state0.<br>[warm reset insensitive]  | <a href="#">Section 8.1.13.4.2</a> |
| 8h     | RM_MPUs_RSTST     | This register logs the different reset sources of the ALWON domain.<br>Each bit is set upon release of the domain reset signal.<br>Must be cleared by software.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.4.3</a> |



#### 8.1.13.4.1 PM\_MPU\_PWRSTCTRL Register (offset = 0h) [reset = 1FF0007h]

PM\_MPU\_PWRSTCTRL is shown in [Figure 8-175](#) and described in [Table 8-192](#).

This register controls the MPU power state to reach upon mpu domain sleep transition

**Figure 8-175. PM\_MPU\_PWRSTCTRL Register**

| 31              | 30              | 29                  | 28             | 27              | 26            | 25         | 24               |
|-----------------|-----------------|---------------------|----------------|-----------------|---------------|------------|------------------|
| Reserved        |                 |                     |                |                 |               | Reserved   | mpu_ram_RETState |
| R-0h            |                 |                     |                |                 |               | R-0h       | R/W-1h           |
| 23              | 22              | 21                  | 20             | 19              | 18            | 17         | 16               |
| mpu_I2_RETState | mpu_I1_RETState | MPU_L2_ONState      | MPU_L1_ONState | MPU_RAM_ONState |               |            |                  |
| R/W-1h          | R/W-1h          | R-3h                |                | R-3h            |               |            | R/W-3h           |
| 15              | 14              | 13                  | 12             | 11              | 10            | 9          | 8                |
| Reserved        |                 |                     |                |                 |               | R-0h       |                  |
| 7               | 6               | 5                   | 4              | 3               | 2             | 1          | 0                |
| Reserved        |                 | LowPowerStateChange | Reserved       |                 | LogicRETState | PowerState |                  |
| R-0h            |                 | R/W-0h              | R-0h           |                 | R/W-1h        | R/W-3h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-192. PM\_MPU\_PWRSTCTRL Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-26 | Reserved            | R    | 0h    |   |
| 25    | Reserved            | R    | 0h    |   |
| 24    | mpu_ram_RETState    | R/W  | 1h    | Default power domain memory(ram) retention state when power domain is in retention  |
| 23    | mpu_I2_RETState     | R/W  | 1h    | Default power domain memory(L2) retention state when power domain is in retention   |
| 22    | mpu_I1_RETState     | R/W  | 1h    | Default power domain memory(L1) retention state when power domain is in retention   |
| 21-20 | MPU_L2_ONState      | R    | 3h    | Default power domain memory state when domain is ON.  |
| 19-18 | MPU_L1_ONState      | R    | 3h    | Default power domain memory state when domain is ON.  |
| 17-16 | MPU_RAM_ONState     | R/W  | 3h    | Default power domain memory state when domain is ON.<br>0x0 = Mem_off<br>0x2 = Reserved<br>0x3 = Mem_on : Memory bank is on when the domain is ON.  |
| 15-5  | Reserved            | R    | 0h    |   |
| 4     | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3     | Reserved            | R    | 0h    |   |
| 2     | LogicRETState       | R/W  | 1h    | Logic state when power domain is RETENTION<br>0x0 = logic_off : Only retention registers are retained and remaining logic is off when the domain is in RETENTION state.<br>0x1 = logic_ret : Whole logic is retained when domain is in RETENTION state.   |

**Table 8-192. PM\_MPU\_PWRSTCTRL Register Field Descriptions (continued)**

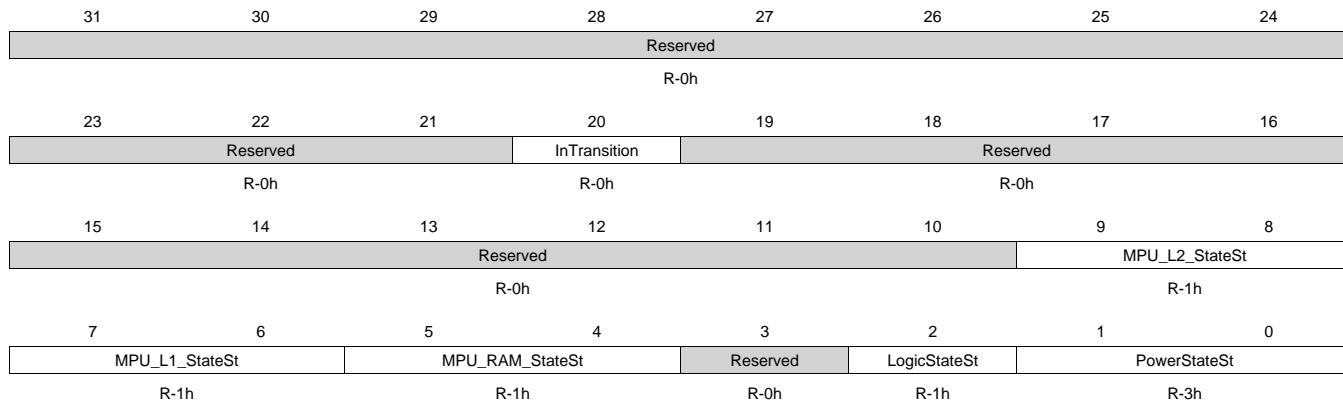
| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 1-0 | PowerState | R/W  | 3h    | Power state control<br>0x0 = OFF : OFF State<br>0x1 = RET<br>0x2 = Reserved<br>0x3 = ON : ON State |

#### 8.1.13.4.2 PM\_MPU\_PWRSTST Register (offset = 4h) [reset = 157h]

PM\_MPU\_PWRSTST is shown in [Figure 8-176](#) and described in [Table 8-193](#).

This register provides a status on the current MPU power domain state0. [warm reset insensitive]

**Figure 8-176. PM\_MPU\_PWRSTST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-193. PM\_MPU\_PWRSTST Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-21 | Reserved        | R    | 0h    |   |
| 20    | InTransition    | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress.              |
| 19-10 | Reserved        | R    | 0h    |   |
| 9-8   | MPU_L2_StateSt  | R    | 1h    | MPU L2 memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                               |
| 7-6   | MPU_L1_StateSt  | R    | 1h    | MPU L1 memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                               |
| 5-4   | MPU_RAM_StateSt | R    | 1h    | MPU_RAM memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                              |
| 3     | Reserved        | R    | 0h    |   |
| 2     | LogicStateSt    | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON  |
| 1-0   | PowerStateSt    | R    | 3h    | Current Power State Status<br>0x0 = OFF : OFF State [warm reset insensitive]<br>0x1 = RET : RET State<br>0x3 = ON : ON State [warm reset insensitive] |

### 8.1.13.4.3 RM\_MPU\_RSTST Register (offset = 8h) [reset = 0h]

RM\_MPU\_RSTST is shown in [Figure 8-177](#) and described in [Table 8-194](#).

This register logs the different reset sources of the ALWON domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]

**Figure 8-177. RM\_MPU\_RSTST Register**

|          |                    |                   |          |          |          |          |          |
|----------|--------------------|-------------------|----------|----------|----------|----------|----------|
| 31       | 30                 | 29                | 28       | 27       | 26       | 25       | 24       |
| Reserved |                    |                   |          |          |          |          |          |
| R-0h     |                    |                   |          |          |          |          |          |
| 23       | 22                 | 21                | 20       | 19       | 18       | 17       | 16       |
| Reserved |                    |                   |          |          |          |          |          |
| R-0h     |                    |                   |          |          |          |          |          |
| 15       | 14                 | 13                | 12       | 11       | 10       | 9        | 8        |
| Reserved |                    | Reserved          |          |          |          |          |          |
| R-0h     |                    |                   |          |          |          |          |          |
| 7        | 6                  | 5                 | 4        | 3        | 2        | 1        | 0        |
| Reserved | ICECRUSHER_MPU_RST | EMULATION_MPU_RST | Reserved | Reserved | Reserved | Reserved | Reserved |
| R-0h     | R/W-0h             | R/W-0h            | R-0h     | R-0h     | R-0h     | R-0h     | R-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-194. RM\_MPU\_RSTST Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-15 | Reserved           | R    | 0h    |  |
| 14-8  | Reserved           | R    | 0h    |  |
| 7     | Reserved           | R    | 0h    |  |
| 6     | ICECRUSHER_MPU_RST | R/W  | 0h    | MPU Processor has been reset due to MPU ICECRUSHER1 reset event<br>0x0 = RESET_NO : No icecrusher reset<br>0x1 = RESET_YES : MPU Processor has been reset upon icecursher reset  |
| 5     | EMULATION_MPU_RST  | R/W  | 0h    | MPU Processor has been reset due to emulation reset source e.g. assert reset command initiated by the icepick module<br>0x0 = RESET_NO : No emulation reset<br>0x1 = RESET_YES : MPU Processor has been reset upon emulation reset |
| 4     | Reserved           | R    | 0h    |  |
| 3     | Reserved           | R    | 0h    |  |
| 2     | Reserved           | R    | 0h    |  |
| 1-0   | Reserved           | R    | 0h    |  |

### 8.1.13.5 PRM\_DEVICE Registers

[Table 8-195](#) lists the memory-mapped registers for the PRM\_DEVICE. All register offset addresses not listed in [Table 8-195](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-195. PRM\_DEVICE REGISTERS**

| Offset | Acronym     | Register Name | Section                            |
|--------|-------------|---------------|------------------------------------|
| 0h     | PRM_RSTCTRL |               | <a href="#">Section 8.1.13.5.1</a> |
| 4h     | PRM_RSTTIME |               | <a href="#">Section 8.1.13.5.2</a> |
| 8h     | PRM_RSTST   |               | <a href="#">Section 8.1.13.5.3</a> |

**Table 8-195. PRM\_DEVICE REGISTERS (continued)**

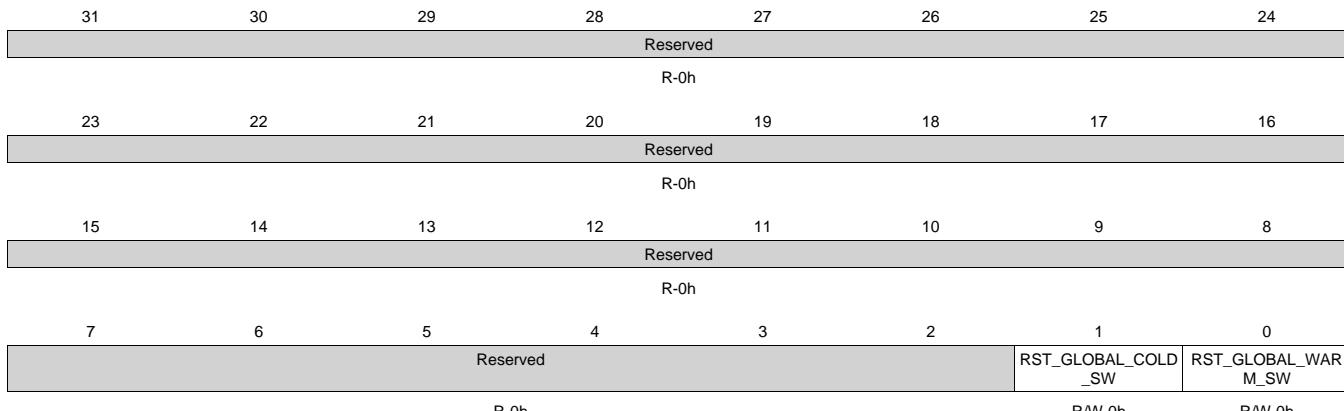
| Offset | Acronym                 | Register Name | Section                            |
|--------|-------------------------|---------------|------------------------------------|
| Ch     | PRM_SRAM_COUNT          |               | <a href="#">Section 8.1.13.5.4</a> |
| 10h    | PRM_LDO_SRAM_CORE_SETUP |               | <a href="#">Section 8.1.13.5.5</a> |
| 14h    | PRM_LDO_SRAM_CORE_CTRL  |               | <a href="#">Section 8.1.13.5.6</a> |
| 18h    | PRM_LDO_SRAM_MPU_SETUP  |               | <a href="#">Section 8.1.13.5.7</a> |
| 1Ch    | PRM_LDO_SRAM_MPU_CTRL   |               | <a href="#">Section 8.1.13.5.8</a> |

### 8.1.13.5.1 PRM\_RSTCTRL Register (offset = 0h) [reset = 0h]

PRM\_RSTCTRL is shown in [Figure 8-178](#) and described in [Table 8-196](#).

Global software cold and warm reset control. This register is auto-cleared. Only write 1 is possible. A read returns 0 only.

**Figure 8-178. PRM\_RSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-196. PRM\_RSTCTRL Register Field Descriptions**

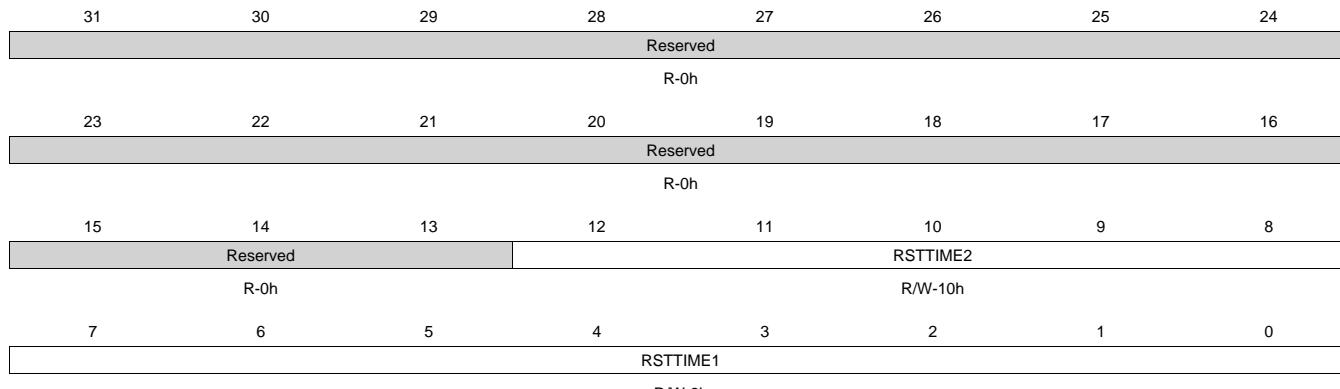
| Bit  | Field              | Type | Reset | Description   |
|------|--------------------|------|-------|---|
| 31-2 | Reserved           | R    | 0h    |   |
| 1    | RST_GLOBAL_COLD_SW | R/W  | 0h    | Global COLD software reset control.<br>This bit is reset only upon a global cold source of reset.<br>Read returns 0.<br>0x0 = 0x0 : Global COLD software reset is cleared.<br>0x1 = 0x1 : Asserts a global COLD software reset. The software must ensure the SDRAM is properly put in self-refresh mode before applying this reset. |
| 0    | RST_GLOBAL_WARM_SW | R/W  | 0h    | Global WARM software reset control.<br>This bit is reset upon any global source of reset (warm and cold).<br>Read returns 0.<br>0x0 = 0x0 : Global warm software reset is cleared.<br>0x1 = 0x1 : Asserts a global warm software reset.   |

### 8.1.13.5.2 PRM\_RSTTIME Register (offset = 4h) [reset = 1006h]

PRM\_RSTTIME is shown in [Figure 8-179](#) and described in [Table 8-197](#).

Reset duration control. [warm reset insensitive]

**Figure 8-179. PRM\_RSTTIME Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-197. PRM\_RSTTIME Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-13 | Reserved | R    | 0h    |  |
| 12-8  | RSTTIME2 | R/W  | 10h   | (Power domain) reset duration 2 (number of CLK_M_OSC clock cycles) |
| 7-0   | RSTTIME1 | R/W  | 6h    | (Global) reset duration 1 (number of CLK_M_OSC clock cycles)       |

### 8.1.13.5.3 PRM\_RSTST Register (offset = 8h) [reset = 1h]

PRM\_RSTST is shown in [Figure 8-180](#) and described in [Table 8-198](#).

This register logs the global reset sources. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]

**Figure 8-180. PRM\_RSTST Register**

|          |                   |          |          |          |                    |                 |        |
|----------|-------------------|----------|----------|----------|--------------------|-----------------|--------|
| 31       | 30                | 29       | 28       | 27       | 26                 | 25              | 24     |
| Reserved |                   |          |          |          |                    |                 |        |
| R-0h     |                   |          |          |          |                    |                 |        |
| 23       | 22                | 21       | 20       | 19       | 18                 | 17              | 16     |
| Reserved |                   |          |          |          |                    |                 |        |
| R-0h     |                   |          |          |          |                    |                 |        |
| 15       | 14                | 13       | 12       | 11       | 10                 | 9               | 8      |
| Reserved |                   |          |          |          |                    |                 |        |
| R-0h     |                   |          |          |          |                    |                 |        |
| 7        | 6                 | 5        | 4        | 3        | 2                  | 1               | 0      |
| Reserved | EXTERNAL_WARM_RST | WDT1_RST | Reserved | Reserved | GLOBAL_WARM_SW_RST | GLOBAL_COLD_RST |        |
| R-0h     | R/W-0h            | R/W-0h   | R-0h     | R/W-0h   | R/W-0h             | R/W-0h          | R/W-1h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-198. PRM\_RSTST Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-10 | Reserved           | R    | 0h    |   |
| 9     | ICEPICK_RST        | R/W  | 0h    | IcePick reset event.<br>This is a source of global warm reset initiated by the emulation.<br>[warm reset insensitive]<br>0x0 = 0x0 : No ICEPICK reset.<br>0x1 = 0x1 : IcePick reset has occurred. |
| 8-6   | Reserved           | R    | 0h    |   |
| 5     | EXTERNAL_WARM_RST  | R/W  | 0h    | External warm reset event [warm reset insensitive]<br>0x0 = 0x0 : No global warm reset.<br>0x1 = 0x1 : Global external warm reset has occurred.   |
| 4     | WDT1_RST           | R/W  | 0h    | Watchdog1 timer reset event.<br>This is a source of global WARM reset.<br>[warm reset insensitive]<br>0x0 = 0x0 : No watchdog reset.<br>0x1 = 0x1 : watchdog reset has occurred.                  |
| 3     | Reserved           | R    | 0h    | Reserved.   |
| 2     | Reserved           | R/W  | 0h    | Reserved.   |
| 1     | GLOBAL_WARM_SW_RST | R/W  | 0h    | Global warm software reset event [warm reset insensitive]<br>0x0 = 0x0 : No global warm SW reset<br>0x1 = 0x1 : Global warm SW reset has occurred.  |
| 0     | GLOBAL_COLD_RST    | R/W  | 1h    | Power-on (cold) reset event [warm reset insensitive]<br>0x0 = 0x0 : No power-on reset.<br>0x1 = 0x1 : Power-on reset has occurred.  |

#### **8.1.13.5.4 PRM\_SRAM\_COUNT Register (offset = Ch) [reset = 78000017h]**

PRM\_SRAM\_COUNT is shown in [Figure 8-181](#) and described in [Table 8-199](#).

Common setup for SRAM LDO transition counters. Applies to all voltage domains. [warm reset insensitive]

**Figure 8-181. PRM\_SRAM\_COUNT Register**

|                 |    |                  |    |    |    |    |    |  |  |  |  |  |  |
|-----------------|----|------------------|----|----|----|----|----|--|--|--|--|--|--|
| 31              | 30 | 29               | 28 | 27 | 26 | 25 | 24 |  |  |  |  |  |  |
| StartUp_Count   |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| R/W-78h         |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| 23              | 22 | 21               | 20 | 19 | 18 | 17 | 16 |  |  |  |  |  |  |
| SLPCNT_VALUE    |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| R/W-0h          |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| 15              | 14 | 13               | 12 | 11 | 10 | 9  | 8  |  |  |  |  |  |  |
| VSETUPCNT_VALUE |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| R/W-0h          |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| 7               | 6  | 5                | 4  | 3  | 2  | 1  | 0  |  |  |  |  |  |  |
| Reserved        |    | PCHARGEcnt_VALUE |    |    |    |    |    |  |  |  |  |  |  |
| R-0h            |    |                  |    |    |    |    |    |  |  |  |  |  |  |
| R/W-17h         |    |                  |    |    |    |    |    |  |  |  |  |  |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-199. PRM\_SRAM\_COUNT Register Field Descriptions**

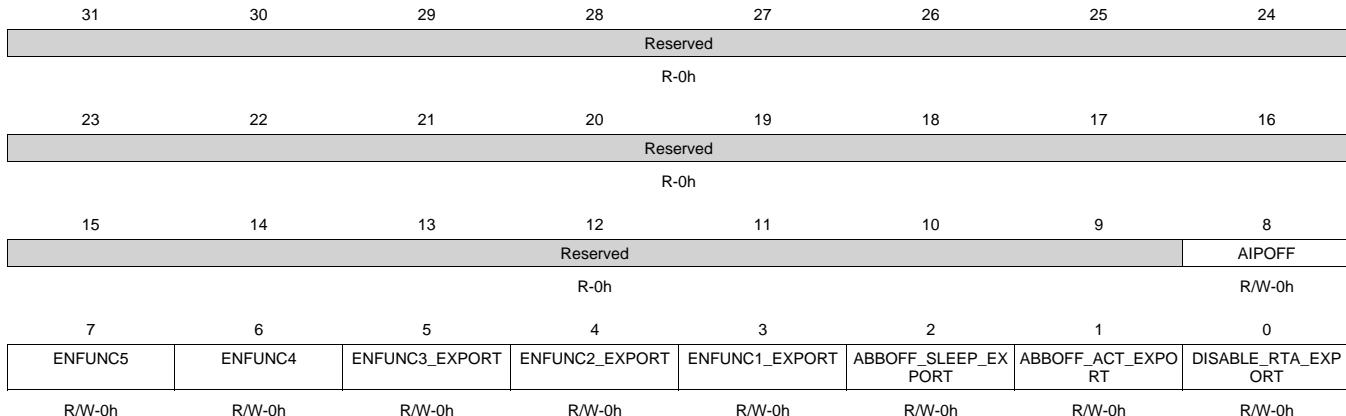
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | StartUp_Count    | R/W  | 78h   | Determines the start-up duration of SRAM and ABB LDO. The duration is computed as 16 x NbCycles of system clock cycles. Target is 50us.       |
| 23-16 | SLPCNT_VALUE     | R/W  | 0h    | Delay between retention/off assertion of last SRAM bank and SRAMALLRET signal to LDO is driven high. Counting on system clock. Target is 2us. |
| 15-8  | VSETUPCNT_VALUE  | R/W  | 0h    | SRAM LDO rampup time from retention to active mode. The duration is computed as 8 x NbCycles of system clock cycles. Target is 30us.          |
| 7-6   | Reserved         | R    | 0h    |   |
| 5-0   | PCHARGEcnt_VALUE | R/W  | 17h   | Delay between de-assertion of standby_rta_ret_on and standby_rta_ret_good. Counting on system clock. Target is 600ns.                         |

### 8.1.13.5.5 PRM\_LDO\_SRAM\_CORE\_SETUP Register (offset = 10h) [reset = 0h]

PRM\_LDO\_SRAM\_CORE\_SETUP is shown in [Figure 8-182](#) and described in [Table 8-200](#).

Setup of the SRAM LDO for CORE voltage domain. [warm reset insensitive]

**Figure 8-182. PRM\_LDO\_SRAM\_CORE\_SETUP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-200. PRM\_LDO\_SRAM\_CORE\_SETUP Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description  |
|------|----------------|------|-------|--|
| 31-9 | Reserved       | R    | 0h    |  |
| 8    | AIPOFF         | R/W  | 0h    | Override on AIPOFF input of SRAM LDO.<br>0x0 = No_Override : AIPOFF signal is not overridden<br>0x1 = Override : AIPOFF signal is overridden to '1'. Corresponding SRAM LDO is disabled and in HZ mode.  |
| 7    | ENFUNC5        | R/W  | 0h    | ENFUNC5 input of SRAM LDO.<br>0x0 = One_step : Active to retention is a one step transfer<br>0x1 = Two_step : Active to retention is a two steps transfer  |
| 6    | ENFUNC4        | R/W  | 0h    | ENFUNC4 input of SRAM LDO.<br>0x0 = Ext_clock : One external clock is supplied<br>0x1 = No_ext_clock : No external clock is supplied   |
| 5    | ENFUNC3_EXPORT | R/W  | 0h    | ENFUNC3 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Sub_regul_disabled : Sub regulation is disabled<br>0x1 = Sub_regul_enabled : Sub regulation is enabled                       |
| 4    | ENFUNC2_EXPORT | R/W  | 0h    | ENFUNC2 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Ext_cap : External cap is used<br>0x1 = No_ext_cap : External cap is not used  |
| 3    | ENFUNC1_EXPORT | R/W  | 0h    | ENFUNC1 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Short_prot_disabled : Short circuit protection is disabled<br>0x1 = Short_prot_enabled : Short circuit protection is enabled |

**Table 8-200. PRM\_LDO\_SRAM\_CORE\_SETUP Register Field Descriptions (continued)**

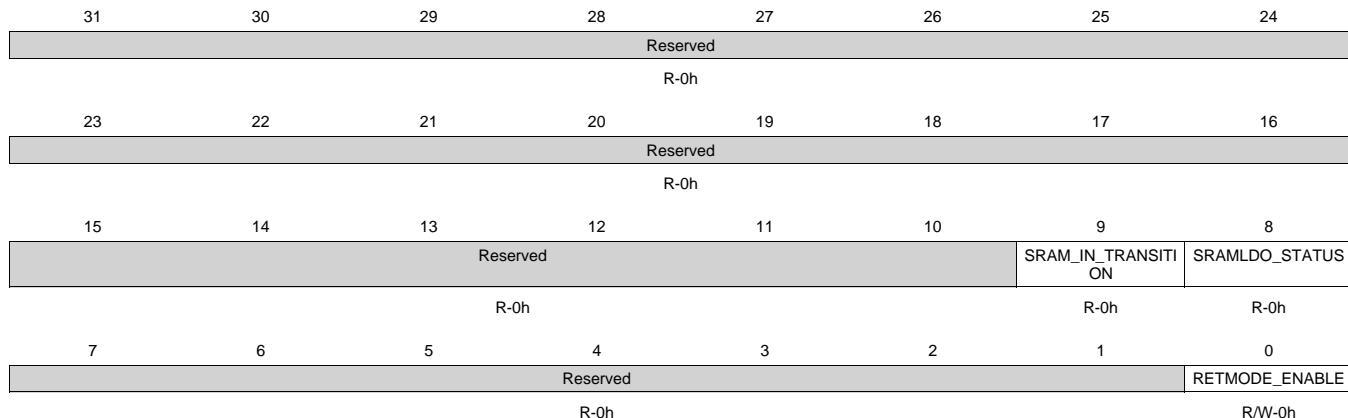
| Bit | Field               | Type | Reset | Description   |
|-----|---------------------|------|-------|---|
| 2   | ABBOFF_SLEEP_EXPORT | R/W  | 0h    | Determines whether SRAMNWA is supplied by VDDS or VDDAR during deep-sleep.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = SRAMNW_SLP_VDDS : SRAMNWA supplied with VDDS<br>0x1(Read) = SRAMNW_SLP_VDDASRAMNWA supplied with VDDAR  |
| 1   | ABBOFF_ACT_EXPORT   | R/W  | 0h    | Determines whether SRAMNWA is supplied by VDDS or VDDAR during active mode.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = SRAMNW_ACT_VDDS : SRAMNWA supplied with VDDS<br>0x1(Read) = SRAMNW_ACT_VDDASRAMNWA supplied with VDDAR |
| 0   | DISABLE_RTA_EXPORT  | R/W  | 0h    | Control for HD memory RTA feature.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = RTA_ENABLED : HD memory RTA feature is enabled<br>0x1 = RTA_DISABLED : HD memory RTA feature is disabled  |

### 8.1.13.5.6 PRM\_LDO\_SRAM\_CORE\_CTRL Register (offset = 14h) [reset = 0h]

PRM\_LDO\_SRAM\_CORE\_CTRL is shown in [Figure 8-183](#) and described in [Table 8-201](#).

Control and status of the SRAM LDO for CORE voltage domain. [warm reset insensitive]

**Figure 8-183. PRM\_LDO\_SRAM\_CORE\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-201. PRM\_LDO\_SRAM\_CORE\_CTRL Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-10 | Reserved           | R    | 0h    |  |
| 9     | SRAM_IN_TRANSITION | R    | 0h    | Status indicating SRAM LDO state machine state.<br>0x0 = IDLE : SRAM LDO state machine is stable<br>0x1 = IN_TRANSITION : SRAM LDO state machine is in transition state  |
| 8     | SRAMLDO_STATUS     | R    | 0h    | SRAMLDO status<br>0x0 = ACTIVE : SRAMLDO is in ACTIVE mode.<br>0x1 = RETENTION : SRAMLDO is on RETENTION mode.   |
| 7-1   | Reserved           | R    | 0h    |  |
| 0     | RETMODE_ENABLE     | R/W  | 0h    | Control if the SRAM LDO retention mode is used or not.<br>0x0 = Disabled : SRAM LDO is not allowed to go to RET mode<br>0x1 = Enabled : SRAM LDO go to RET mode when all memory of voltage domain are OFF or RET |

### 8.1.13.5.7 PRM\_LDO\_SRAM\_MPUMPU\_SETUP Register (offset = 18h) [reset = 0h]

PRM\_LDO\_SRAM\_MPUMPU\_SETUP is shown in [Figure 8-184](#) and described in [Table 8-202](#).

Setup of the SRAM LDO for MPU voltage domain. [warm reset insensitive]

**Figure 8-184. PRM\_LDO\_SRAM\_MPUMPU\_SETUP Register**

|          |         |                |                |                |                     |                  |                   |
|----------|---------|----------------|----------------|----------------|---------------------|------------------|-------------------|
| 31       | 30      | 29             | 28             | 27             | 26                  | 25               | 24                |
| Reserved |         |                |                |                |                     |                  |                   |
| R-0h     |         |                |                |                |                     |                  |                   |
| 23       | 22      | 21             | 20             | 19             | 18                  | 17               | 16                |
| Reserved |         |                |                |                |                     |                  |                   |
| R-0h     |         |                |                |                |                     |                  |                   |
| 15       | 14      | 13             | 12             | 11             | 10                  | 9                | 8                 |
| Reserved |         |                |                |                |                     |                  |                   |
| R-0h     |         |                |                |                |                     |                  |                   |
| 7        | 6       | 5              | 4              | 3              | 2                   | 1                | 0                 |
| ENFUNC5  | ENFUNC4 | ENFUNC3_EXPORT | ENFUNC2_EXPORT | ENFUNC1_EXPORT | ABBOFF_SLEEP_EXPORT | ABBOFF_ACT_EXPOR | DISABLE_RTA_EXPOR |
| R/W-0h   | R/W-0h  | R/W-0h         | R/W-0h         | R/W-0h         | R/W-0h              | R/W-0h           | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-202. PRM\_LDO\_SRAM\_MPUMPU\_SETUP Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description  |
|------|----------------|------|-------|--|
| 31-9 | Reserved       | R    | 0h    |  |
| 8    | AIPOFF         | R/W  | 0h    | Override on AIPOFF input of SRAM LDO.<br>0x0 = No_Override : AIPOFF signal is not overridden<br>0x1 = Override : AIPOFF signal is overridden to '1'. Corresponding SRAM LDO is disabled and in HZ mode.  |
| 7    | ENFUNC5        | R/W  | 0h    | ENFUNC5 input of SRAM LDO.<br>0x0 = One_step : Active to retention is a one step transfer<br>0x1 = Two_step : Active to retention is a two steps transfer  |
| 6    | ENFUNC4        | R/W  | 0h    | ENFUNC4 input of SRAM LDO.<br>0x0 = Ext_clock : One external clock is supplied<br>0x1 = No_ext_clock : No external clock is supplied   |
| 5    | ENFUNC3_EXPORT | R/W  | 0h    | ENFUNC3 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Sub_regul_disabled : Sub regulation is disabled<br>0x1 = Sub_regul_enabled : Sub regulation is enabled                       |
| 4    | ENFUNC2_EXPORT | R/W  | 0h    | ENFUNC2 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Ext_cap : External cap is used<br>0x1 = No_ext_cap : External cap is not used  |
| 3    | ENFUNC1_EXPORT | R/W  | 0h    | ENFUNC1 input of SRAM LDO.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = Short_prot_disabled : Short circuit protection is disabled<br>0x1 = Short_prot_enabled : Short circuit protection is enabled |

**Table 8-202. PRM\_LDO\_SRAM\_MPUMPU\_SETUP Register Field Descriptions (continued)**

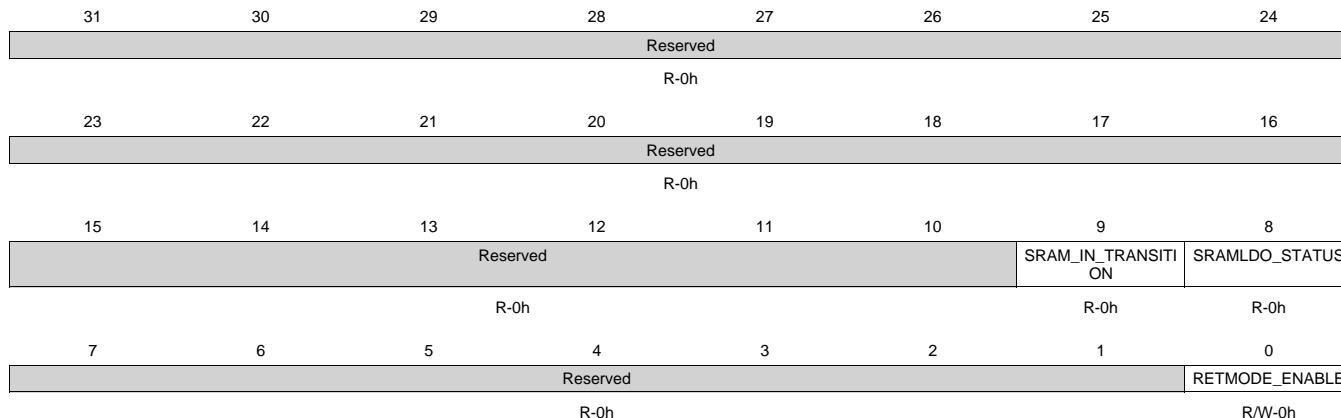
| Bit | Field               | Type | Reset | Description   |
|-----|---------------------|------|-------|---|
| 2   | ABBOFF_SLEEP_EXPORT | R/W  | 0h    | Determines whether SRAMNWA is supplied by VDDS or VDDAR during deep-sleep.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = SRAMNW_SLP_VDDS : SRAMNWA supplied with VDDS<br>0x1(Read) = SRAMNW_SLP_VDDASRAMNWA supplied with VDDAR  |
| 1   | ABBOFF_ACT_EXPORT   | R/W  | 0h    | Determines whether SRAMNWA is supplied by VDDS or VDDAR during active mode.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = SRAMNW_ACT_VDDS : SRAMNWA supplied with VDDS<br>0x1(Read) = SRAMNW_ACT_VDDASRAMNWA supplied with VDDAR |
| 0   | DISABLE_RTA_EXPORT  | R/W  | 0h    | Control for HD memory RTA feature.<br>After PowerOn reset and Efuse sensing, this bitfield is automatically loaded with an Efuse value from control module.<br>Bitfield remains writable after this.<br>0x0 = RTA_ENABLED : HD memory RTA feature is enabled<br>0x1 = RTA_DISABLED : HD memory RTA feature is disabled  |

### 8.1.13.5.8 PRM\_LDO\_SRAM\_MPUMPU\_CTRL Register (offset = 1Ch) [reset = 0h]

PRM\_LDO\_SRAM\_MPUMPU\_CTRL is shown in [Figure 8-185](#) and described in [Table 8-203](#).

Control and status of the SRAM LDO for MPU voltage domain. [warm reset insensitive]

**Figure 8-185. PRM\_LDO\_SRAM\_MPUMPU\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-203. PRM\_LDO\_SRAM\_MPUMPU\_CTRL Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-10 | Reserved           | R    | 0h    |  |
| 9     | SRAM_IN_TRANSITION | R    | 0h    | Status indicating SRAM LDO state machine state.<br>0x0 = IDLE : SRAM LDO state machine is stable<br>0x1 = IN_TRANSITION : SRAM LDO state machine is in transition state  |
| 8     | SRAMLDO_STATUS     | R    | 0h    | SRAMLDO status<br>0x0 = ACTIVE : SRAMLDO is in ACTIVE mode.<br>0x1 = RETENTION : SRAMLDO is on RETENTION mode.   |
| 7-1   | Reserved           | R    | 0h    |  |
| 0     | RETMODE_ENABLE     | R/W  | 0h    | Control if the SRAM LDO retention mode is used or not.<br>0x0 = Disabled : SRAM LDO is not allowed to go to RET mode<br>0x1 = Enabled : SRAM LDO go to RET mode when all memory of voltage domain are OFF or RET |

### 8.1.13.6 PRM\_RTC Registers

[Table 8-204](#) lists the memory-mapped registers for the PRM\_RTC. All register offset addresses not listed in [Table 8-204](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-204. PRM\_RTC REGISTERS**

| Offset | Acronym          | Register Name   | Section                            |
|--------|------------------|---|------------------------------------|
| 0h     | PM_RTC_PWRSTCTRL | This register controls the RTC power state to reach upon mpu domain sleep transition                | <a href="#">Section 8.1.13.6.1</a> |
| 4h     | PM_RTC_PWRSTST   | This register provides a status on the current RTC power domain state0.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.6.2</a> |

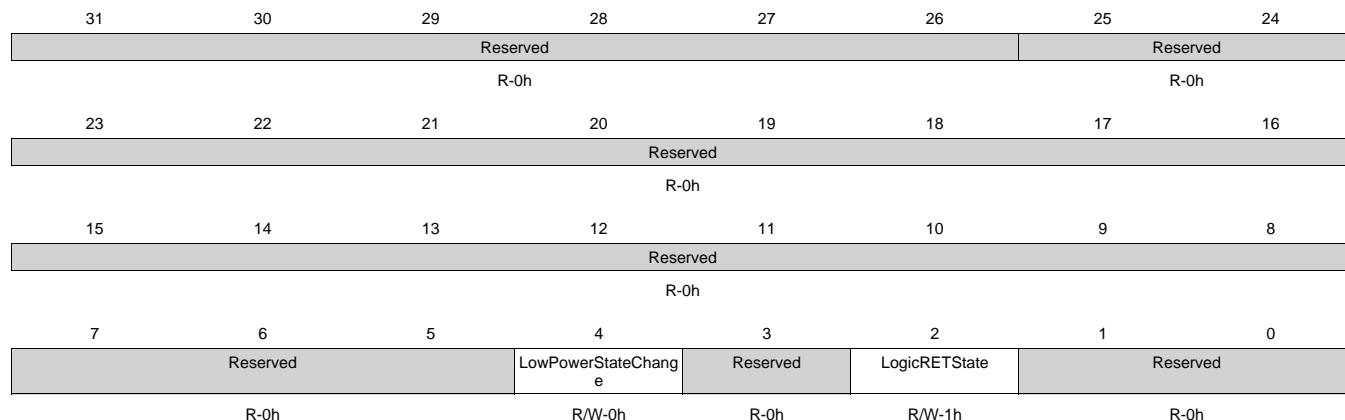


### 8.1.13.6.1 PM\_RTC\_PWRSTCTRL Register (offset = 0h) [reset = 4h]

PM\_RTC\_PWRSTCTRL is shown in [Figure 8-186](#) and described in [Table 8-205](#).

This register controls the RTC power state to reach upon mpu domain sleep transition

**Figure 8-186. PM\_RTC\_PWRSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-205. PM\_RTC\_PWRSTCTRL Register Field Descriptions**

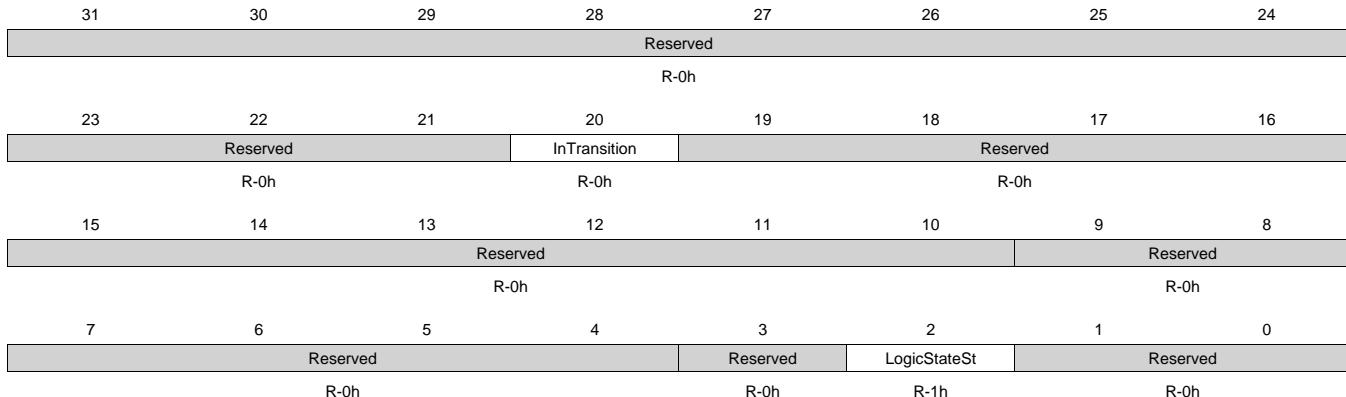
| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-26 | Reserved            | R    | 0h    |   |
| 25-16 | Reserved            | R    | 0h    |   |
| 15-5  | Reserved            | R    | 0h    |   |
| 4     | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3     | Reserved            | R    | 0h    |   |
| 2     | LogicRETState       | R/W  | 1h    | Logic state when power domain is RETENTION<br>0x0 = logic_off : Only retention registers are retained and remaining logic is off when the domain is in RETENTION state.<br>0x1 = logic_ret : Whole logic is retained when domain is in RETENTION state.   |
| 1-0   | Reserved            | R    | 0h    |   |

### 8.1.13.6.2 PM\_RTC\_PWRSTST Register (offset = 4h) [reset = 4h]

PM\_RTC\_PWRSTST is shown in [Figure 8-187](#) and described in [Table 8-206](#).

This register provides a status on the current RTC power domain state0. [warm reset insensitive]

**Figure 8-187. PM\_RTC\_PWRSTST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-206. PM\_RTC\_PWRSTST Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description  |
|-------|--------------|------|-------|--|
| 31-21 | Reserved     | R    | 0h    |  |
| 20    | InTransition | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress. |
| 19-10 | Reserved     | R    | 0h    |  |
| 9-4   | Reserved     | R    | 0h    |  |
| 3     | Reserved     | R    | 0h    |  |
| 2     | LogicStateSt | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON   |
| 1-0   | Reserved     | R    | 0h    |  |

### 8.1.13.7 PRM\_GFX Registers

[Table 8-207](#) lists the memory-mapped registers for the PRM\_GFX. All register offset addresses not listed in [Table 8-207](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-207. PRM\_GFX REGISTERS**

| Offset | Acronym          | Register Name   | Section                            |
|--------|------------------|---|------------------------------------|
| 0h     | PM_GFX_PWRSTCTRL | This register controls the GFX power state to reach upon a domain sleep transition.   | <a href="#">Section 8.1.13.7.1</a> |
| 4h     | RM_GFX_RSTCTRL   | This register controls the release of the GFX Domain resets.  | <a href="#">Section 8.1.13.7.2</a> |
| 10h    | PM_GFX_PWRSTST   | This register provides a status on the current GFX power domain state.<br>[warm reset insensitive]  | <a href="#">Section 8.1.13.7.3</a> |
| 14h    | RM_GFX_RSTST     | This register logs the different reset sources of the GFX domain.<br>Each bit is set upon release of the domain reset signal.<br>Must be cleared by software.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.7.4</a> |



### 8.1.13.7.1 PM\_GFX\_PWRSTCTRL Register (offset = 0h) [reset = 60044h]

PM\_GFX\_PWRSTCTRL is shown in [Figure 8-188](#) and described in [Table 8-208](#).

This register controls the GFX power state to reach upon a domain sleep transition.

**Figure 8-188. PM\_GFX\_PWRSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-208. PM\_GFX\_PWRSTCTRL Register Field Descriptions**

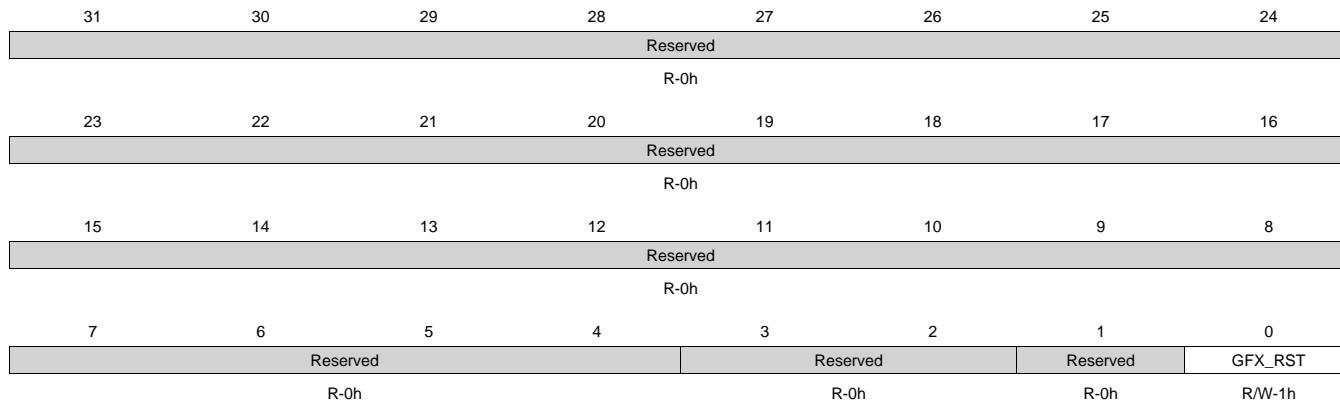
| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-19 | Reserved            | R    | 0h    |   |
| 18-17 | GFX_MEM_ONState     | R    | 3h    | GFX memory state when domain is ON.   |
| 16-7  | Reserved            | R    | 0h    |   |
| 6     | GFX_MEM_RETState    | R/W  | 1h    |   |
| 5     | Reserved            | R    | 0h    |   |
| 4     | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3     | Reserved            | R    | 0h    |   |
| 2     | LogicRETState       | R/W  | 1h    | Logic state when power domain is RETENTION<br>0x0 = logic_off : Only retention registers are retained and remaining logic is off when the domain is in RETENTION state.<br>0x1 = logic_ret : Whole logic is retained when domain is in RETENTION state.   |
| 1-0   | PowerState          | R/W  | 0h    | Power state control<br>0x0 = OFF : OFF State<br>0x1 = RET<br>0x2 = reserved_1<br>0x3 = ON : ON State  |

### 8.1.13.7.2 RM\_GFX\_RSTCTRL Register (offset = 4h) [reset = 1h]

RM\_GFX\_RSTCTRL is shown in [Figure 8-189](#) and described in [Table 8-209](#).

This register controls the release of the GFX Domain resets.

**Figure 8-189. RM\_GFX\_RSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-209. RM\_GFX\_RSTCTRL Register Field Descriptions**

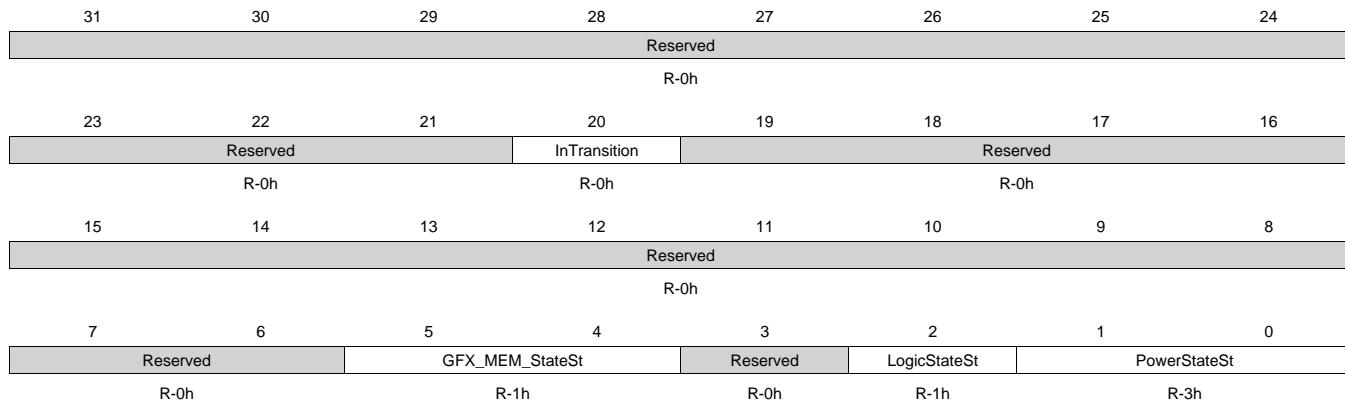
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-4 | Reserved | R    | 0h    |   |
| 3-2  | Reserved | R    | 0h    |   |
| 1    | Reserved | R    | 0h    |   |
| 0    | GFX_RST  | R/W  | 1h    | GFX domain local reset control<br>0x0 = CLEAR : Reset is cleared for the GFX Domain (SGX530)<br>0x1 = ASSERT : Reset is asserted for the GFX Domain (SGX 530) |

### 8.1.13.7.3 PM\_GFX\_PWRSTST Register (offset = 10h) [reset = 17h]

PM\_GFX\_PWRSTST is shown in [Figure 8-190](#) and described in [Table 8-210](#).

This register provides a status on the current GFX power domain state. [warm reset insensitive]

**Figure 8-190. PM\_GFX\_PWRSTST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-210. PM\_GFX\_PWRSTST Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-21 | Reserved        | R    | 0h    |   |
| 20    | InTransition    | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress.  |
| 19-6  | Reserved        | R    | 0h    |   |
| 5-4   | GFX_MEM_StateSt | R    | 1h    | GFX memory state status<br>0x0 = Mem_off : Memory is OFF<br>0x2 = Reserved : Reserved<br>0x3 = Mem_on : Memory is ON                      |
| 3     | Reserved        | R    | 0h    |   |
| 2     | LogicStateSt    | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON  |
| 1-0   | PowerStateSt    | R    | 3h    | Current Power State Status<br>0x0 = OFF : OFF State [warm reset insensitive]<br>0x1 = RET<br>0x3 = ON : ON State [warm reset insensitive] |

#### 8.1.13.7.4 RM\_GFX\_RSTST Register (offset = 14h) [reset = 0h]

RM\_GFX\_RSTST is shown in [Figure 8-191](#) and described in [Table 8-211](#).

This register logs the different reset sources of the GFX domain. Each bit is set upon release of the domain reset signal. Must be cleared by software. [warm reset insensitive]

**Figure 8-191. RM\_GFX\_RSTST Register**

|          |    |    |    |          |      |          |                |
|----------|----|----|----|----------|------|----------|----------------|
| 31       | 30 | 29 | 28 | 27       | 26   | 25       | 24             |
| Reserved |    |    |    |          |      |          |                |
| R-0h     |    |    |    |          |      |          |                |
| 23       | 22 | 21 | 20 | 19       | 18   | 17       | 16             |
| Reserved |    |    |    |          |      |          |                |
| R-0h     |    |    |    |          |      |          |                |
| 15       | 14 | 13 | 12 | 11       | 10   | 9        | 8              |
| Reserved |    |    |    | Reserved |      |          |                |
| R-0h     |    |    |    | R-0h     |      |          |                |
| 7        | 6  | 5  | 4  | 3        | 2    | 1        | 0              |
| Reserved |    |    |    |          | R-0h | Reserved | GFX_RST R/W-0h |
| R-0h     |    |    |    |          |      |          |                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-211. RM\_GFX\_RSTST Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-12 | Reserved | R    | 0h    |  |
| 11-2  | Reserved | R    | 0h    |  |
| 1     | Reserved | R    | 0h    |  |
| 0     | GFX_RST  | R/W  | 0h    | GFX Domain Logic Reset<br>0x0 = RESET_NO : No SW reset occurred<br>0x1 = RESET_YES : GFX Domain Logic has been reset upon SW reset |

#### 8.1.13.8 PRM\_CEFUSE Registers

[Table 8-212](#) lists the memory-mapped registers for the PRM\_CEFUSE. All register offset addresses not listed in [Table 8-212](#) should be considered as reserved locations and the register contents should not be modified.

**Table 8-212. PRM\_CEFUSE REGISTERS**

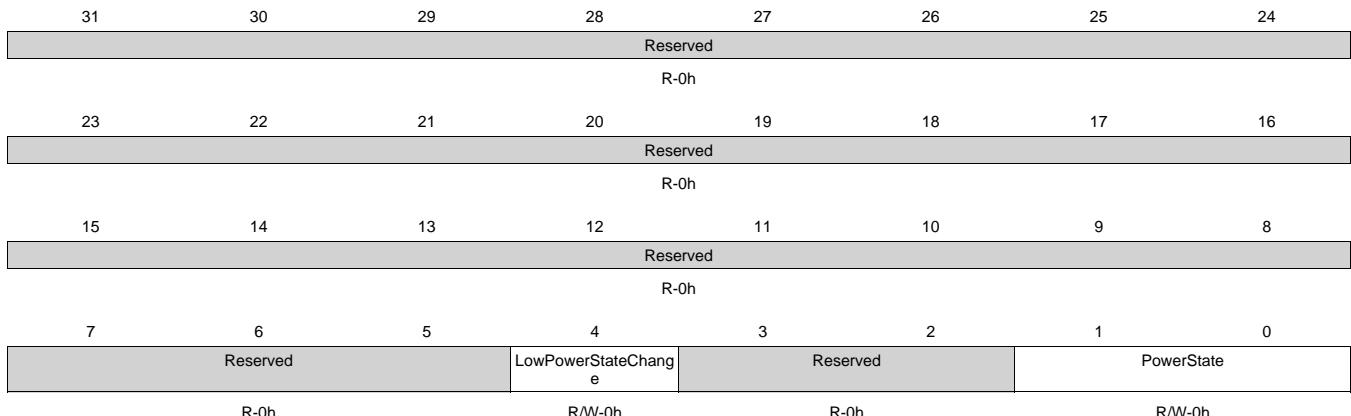
| Offset | Acronym             | Register Name   | Section                            |
|--------|---------------------|---|------------------------------------|
| 0h     | PM_CEFUSE_PWRSTCTRL | This register controls the CEFUSE power state to reach upon a domain sleep transition                 | <a href="#">Section 8.1.13.8.1</a> |
| 4h     | PM_CEFUSE_PWRSTST   | This register provides a status on the current CEFUSE power domain state.<br>[warm reset insensitive] | <a href="#">Section 8.1.13.8.2</a> |

### 8.1.13.8.1 PM\_CEFUSE\_PWRSTCTRL Register (offset = 0h) [reset = 0h]

PM\_CEFUSE\_PWRSTCTRL is shown in [Figure 8-192](#) and described in [Table 8-213](#).

This register controls the CEFUSE power state to reach upon a domain sleep transition

**Figure 8-192. PM\_CEFUSE\_PWRSTCTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-213. PM\_CEFUSE\_PWRSTCTRL Register Field Descriptions**

| Bit  | Field               | Type | Reset | Description   |
|------|---------------------|------|-------|---|
| 31-5 | Reserved            | R    | 0h    |   |
| 4    | LowPowerStateChange | R/W  | 0h    | Power state change request when domain has already performed a sleep transition.<br>Allows going into deeper low power state without waking up the power domain.<br>0x0 = DIS : Do not request a low power state change.<br>0x1 = EN : Request a low power state change. This bit is automatically cleared when the power state is effectively changed or when power state is ON. |
| 3-2  | Reserved            | R    | 0h    |   |
| 1-0  | PowerState          | R/W  | 0h    | Power state control<br>0x0 = OFF : OFF state<br>0x1 = Reserved : Reserved<br>0x2 = INACT : INACTIVE state<br>0x3 = ON : ON State  |

### 8.1.13.8.2 PM\_CEFUSE\_PWRSTST Register (offset = 4h) [reset = 7h]

PM\_CEFUSE\_PWRSTST is shown in [Figure 8-193](#) and described in [Table 8-214](#).

This register provides a status on the current CEFUSE power domain state. [warm reset insensitive]

**Figure 8-193. PM\_CEFUSE\_PWRSTST Register**

|          |    |              |    |              |    |                       |    |
|----------|----|--------------|----|--------------|----|-----------------------|----|
| 31       | 30 | 29           | 28 | 27           | 26 | 25                    | 24 |
| Reserved |    |              |    |              |    | LastPowerStateEntered |    |
| R-0h     |    |              |    |              |    | R/W-0h                |    |
| 23       | 22 | 21           | 20 | 19           | 18 | 17                    | 16 |
| Reserved |    | InTransition |    | Reserved     |    |                       |    |
| R-0h     |    | R-0h         |    | R-0h         |    |                       |    |
| 15       | 14 | 13           | 12 | 11           | 10 | 9                     | 8  |
| Reserved |    |              |    |              |    | R-0h                  |    |
| 7        | 6  | 5            | 4  | 3            | 2  | 1                     | 0  |
| Reserved |    |              |    | LogicStateSt |    | PowerStateSt          |    |
| R-0h     |    |              |    | R-1h         |    | R-3h                  |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 8-214. PM\_CEFUSE\_PWRSTST Register Field Descriptions**

| Bit   | Field                 | Type | Reset | Description  |
|-------|-----------------------|------|-------|--|
| 31-26 | Reserved              | R    | 0h    |  |
| 25-24 | LastPowerStateEntered | R/W  | 0h    | Last low power state entered.<br>Set to 0x3 upon write of the same only.<br>This register is intended for debug purpose only.<br>0x0 = OFF : Power domain was previously OFF<br>0x1 = ON : Power domain was previously ON-ACTIVE |
| 23-21 | Reserved              | R    | 0h    |  |
| 20    | InTransition          | R    | 0h    | Domain transition status<br>0x0 = No : No on-going transition on power domain<br>0x1 = Ongoing : Power domain transition is in progress.   |
| 19-3  | Reserved              | R    | 0h    |  |
| 2     | LogicStateSt          | R    | 1h    | Logic state status<br>0x0 = OFF : Logic in domain is OFF<br>0x1 = ON : Logic in domain is ON   |
| 1-0   | PowerStateSt          | R    | 3h    | Current power state status<br>0x0 = OFF : Power domain is OFF<br>0x1 = RET : Power domain is in RETENTION<br>0x2 = INACTIVE : Power domain is ON-INACTIVE<br>0x3 = ON : Power domain is ON-ACTIVE                                |

**Control Module**

This chapter describes the control module of the device.

| Topic                                     | Page       |
|---|------------|
| 9.1 <b>Introduction</b> .....             | <b>743</b> |
| 9.2 <b>Functional Description</b> .....   | <b>743</b> |
| 9.3 <b>CONTROL_MODULE Registers</b> ..... | <b>753</b> |

## 9.1 Introduction

The control module includes status and control logic not addressed within the peripherals or the rest of the device infrastructure. This module provides interface to control the following areas of the device:

- Functional I/O multiplexing
- Emulation controls
- Device control and status
- DDR PHY control and IO control registers
- EDMA event multiplexing control registers

**Note:** For writing to the control module registers, the Cortex A8 MPU will need to be in privileged mode of operation and writes will not work from user mode.

## 9.2 Functional Description

### 9.2.1 Control Module Initialization

The control module responds only to the internal POR and device type. At power on, reset values for the registers define the safe state for the device. In the initialization mode, only modules to be used at boot time are associated with the pads. Other module inputs are internally tied and output pads are turned off. After POR, software sets the pad functional multiplexing and configuration registers to the desired values according to the requested device configuration.

General-purpose (GP) devices include features that are inaccessible or unavailable. These inaccessible registers define the default or fixed device configuration or behavior.

The CONTROL\_STATUS[7:0] SYS\_BOOT bit field reflects the state of the sys\_boot pins captured at POR in the PRCM module.

### 9.2.2 Pad Control Registers

The Pad Control Registers are 32-bit registers to control the signal muxing and other aspects of each I/O pad. After POR, software must set the pad functional multiplexing and configuration registers to the desired values according to the requested device configuration. The configuration is controlled by pads or by a group of pads. Each configurable pin has its own configuration register for pullup/down control and for the assignment to a given module.

The following table shows the generic Pad Control Register Description.

**Table 9-1. Pad Control Register Field Descriptions**

| Bit  | Field       | Value  | Description  |
|------|-------------|--------|--|
| 31-7 | Reserved    |        | Reserved. Read returns 0.  |
| 6    | SLEWCTRL    | 0<br>1 | Select between faster or slower slew rate.<br>Fast<br>Slow <sup>(1)</sup>  |
| 5    | RXACTIVE    | 0<br>1 | Input enable value for the Pad. Set to 0 for output only. Set to 1 for input or output.<br>Receiver disabled<br>Receiver enabled |
| 4    | PULLTYPESEL | 0<br>1 | Pad pullup/pulldown type selection<br>Pulldown selected<br>Pullup selected   |
| 3    | PULLUDEN    | 0<br>1 | Pad Pullup/pulldown enable<br>Pullup/pulldown enabled.<br>Pullup/pulldown disabled.  |
| 2-0  | MUXMODE     |        | Pad functional signal mux select   |

<sup>(1)</sup> Some peripherals do not support slow slew rate. To determine which interfaces support each slew rate, see AM335x ARM Cortex-A8 Microprocessors (MPUs) (literature number [SPRS717](#)).

### 9.2.2.1 Mode Selection

The MUXMODE field in the pad control registers defines the multiplexing mode applied to the pad. Modes are referred to by their decimal (from 0 to 7) or binary (from 0b000 to 0b111) representation. For most pads, the reset value for the MUXMODE field in the registers is 0b111. The exceptions are pads to be used at boot time to transfer data from selected peripherals to the external flash memory.

**Table 9-2. Mode Selection**

| MUXMODE | Selected Mode         |
|---------|-----------------------|
| 000b    | Primary Mode = Mode 0 |
| 001b    | Mode 1                |
| 010b    | Mode 2                |
| 011b    | Mode 3                |
| 100b    | Mode 4                |
| 101b    | Mode 5                |
| 110b    | Mode 6                |
| 111b    | Mode 7                |

Mode 0 is the primary mode. When mode 0 is set, the function mapped to the pin corresponds to the name of the pin. Mode 1 to mode 7 are possible modes for alternate functions. On each pin, some modes are used effectively for alternate functions, while other modes are unused and correspond to no functional configuration.

### 9.2.2.2 Pull Selection

There is no automatic gating control to ensure that internal weak pull-down/pull up resistors on a pad are disconnected whenever the pad is configured as output. If a pad is always configured in output mode, it is recommended for user software to disable any internal pull resistor tied to it, to avoid unnecessary consumption. The following table summarizes the various possible combinations of PULLTYPESEL and PULLUDEN fields of PAD control register.

**Table 9-3. Pull Selection**

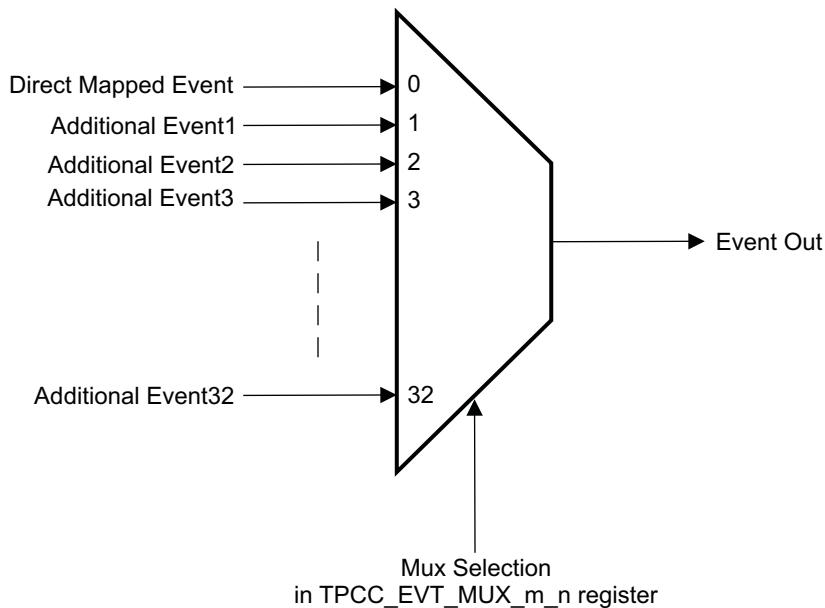
| PULL TYPE   |              | Pin Behavior                        |
|-------------|--------------|-------------------------------------|
| PULLTYPESEL | PULLUDENABLE |                                     |
| 0b          | 0b           | Pulldown selected and activated     |
| 0b          | 1b           | Pulldown selected but not activated |
| 1b          | 0b           | Pullup selected and activated       |
| 1b          | 1b           | Pullup selected but not activated   |

### 9.2.2.3 RX Active

The RXACTIVE bit is used to enable and disable the input buffer. This control can be used to help with power leakage or device isolation through the I/O. The characteristic of the signal is ultimately dictated by the mux mode the pad is put into.

### 9.2.3 EDMA Event Multiplexing

The device has more DMA events than can be accommodated by the TPCC's maximum number of events, which is 64. To overcome the device has one crossbar at the top level. This module will multiplex the extra events with all of the direct mapped events. Mux control registers are defined in the Control Module to select the event to be routed to the TPCC. Direct mapped event is the default (mux selection set to '0').

**Figure 9-1. Event Crossbar**


For every EDMA event there is a cross bar implemented in the design as shown in the figure. The direct mapped event/interrupt will be always connected to Mux input[0], The additional events will be connected to Mux input[1], Mux input[2].etc as defined in EDMA event table. The Mux selection value is programmed into the corresponding TPCC\_EVT\_MUX\_n register. The EVT\_MUX value can take a value from 1 to 32. Other values are reserved. By default the MUX\_selection value is written to 0, which means the direct mapped event is connected to the Event output.

When the additional event is selected through the Cross bar programming the direct mapped event cannot be used.

For example, when TINT0 (Timer Interrupt 0) event, which is not directly mapped to the DMA event source needs to be connected to EDMA channel no 24 (which is directly mapped to SDTXEVT0 event). The user has to program the EVT\_MUX\_24 field in TPCC\_EVT\_MUX\_24\_27 register to 22 (value corresponding to TINT0 interrupt in crossbar mapping). When this is set, TINT0 interrupt event will trigger the channel 24.

Please note that once this is set. The SDTXEVT0 can no longer be handled by EDMA. The user has to allocate the correct DMA event number for crossbar mapped events so that there is no compromise on the channel allocation for the used event numbers.

## 9.2.4 Device Control and Status

### 9.2.4.1 Control and Boot Status

The device configuration is set during power on or hardware reset (PORz sequence) by the configuration input pins (SYSBOOT[15:0]).The CONTROL\_STATUS register reflects the system boot and the device type configuration values as sampled when the power-on reset (PORz) signal is asserted. The Configuration input pins are sampled continuously during the PORz active period and the final sampled value prior to the last rising edge is latched in the register. The CONTROL\_STATUS register gives the status of the device boot process.

### 9.2.4.2 Interprocessor Communication

The control module has the IPC\_MSG\_REG (7:0) registers which is for sharing messages between Cortex M3 and the Cortex A8 MPU. The M3 TX end of event (M3\_TXEV\_EOI) register provides the mechanism to clear/enable the TX Event from Cortex M3 to Cortex A8 MPU Subsystem. See the M3\_TXEV\_EOI register description for further detail.

### 9.2.4.3 Initiator Priority Control

The control module provides the registers to control the bus interconnect priority and the EMIF priority.

#### 9.2.4.3.1 Initiator Priority Control for Interconnect

The INIT\_PRIORITY\_n register controls the infrastructure priority at the bus interconnects. This can be used for dynamic priority escalation. There are bit fields that control the interconnect priority for each bus initiator. By default all the initiators are given equal priority and the allocation is done on a round robin basis.

The priority can take a value from 0 to 3. The following table gives the valid set of priority values.

**Table 9-4. Interconnect Priority Values**

| Interconnect Priority Value | Remarks         |
|-----------------------------|-----------------|
| 00                          | Low priority    |
| 01                          | Medium priority |
| 10                          | Reserved        |
| 11                          | High priority   |

#### 9.2.4.3.2 Initiator Priority at EMIF

The MREQPRI0 register provides an interface to change the access priorities for the various masters accessing the EMIF(DDR). Software can make use of this register to set the requestor priorities for required EMIF arbitration. The EMIF priority can take a value from 000b to 111b where 000b will be the highest priority and 111b will be lowest priority.

### 9.2.4.4 Peripheral Control and Status

#### 9.2.4.4.1 USB Control and Status

The USB\_CTRLn and USB\_STSn registers reflect the Control and Status of the USB instances. The USB IO lines can be used as UART TX and RX lines the USB Control register bit field GPIO MODE has settings that configures the USB lines as GPIO lines. The other USB PHY control settings for controlling the OTG settings and PHY are part of the USB\_CTRLn register.

The USB\_STSn register gives the status of the USB PHY module. See the USB\_STSn register description for further details.

See [Section 16.2.4, USB GPIO Details](#), for more information.

#### 9.2.4.4.2 USB Charger Detect

Each USB PHY contains circuitry which can automatically detect the presence of a charger attached to the USB port. The charger detection circuitry is compliant to the Battery Charging Specification Revision 1.1 from the USB Implementers Forum, which can be found at [www.usb.org](http://www.usb.org). See this document for more details on USB charger implementation.

#### 9.2.4.4.2.1 Features

The charger detection circuitry of each PHY has the following features:

- Contains a state machine which can automatically detect the presence of a Charging Downstream Port or a Dedicated Charging Port (see the Battery Charging Specification for the definition of these terms)
- Outputs a charger enable signal (3.3 V level active high CMOS driver) when a charger is present.
- Allows you to enable/disable the circuitry to save power
- The detection circuitry requires only a 3.3-V supply to be present to operate.
- The charger detection also has a manual mode which allows the user to implement the battery charging specification in software.

### 9.2.4.4.2.2 Operation

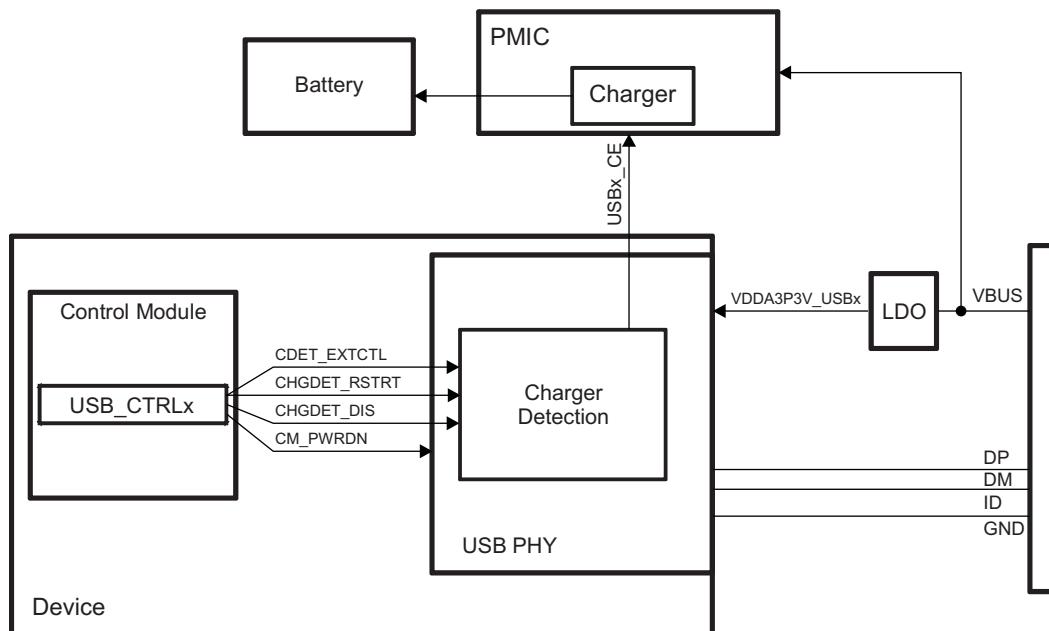
The control module gives the following interface to control the automatic charger detection circuitry:

- **USB\_CTRLx.CDET\_EXTCTL**: Turns the automatic detection on/off. Keep this bit 0 to keep the automatic detection on. Changing this to 1 enables the manual mode.
- **USB\_CTRLx.CHGDET\_RSTRT**: Restarts the charger detection state machine. To initiate the charger detection, change this bit from 1 to 0. If this bit is 1, the charger enable output (CE) is disabled.
- **USB\_CTRLx.CHGDET\_DIS**: Enables/disables the charger detection circuitry. Keep this bit 0 to keep this charger detection enabled. Setting this bit to 1 will power down the charger detection circuitry.
- **USB\_CTRLx.CM\_PWRDN**: Powers up/down the PHY which contains the charger detection circuitry. Clear this bit to 0 to enable power to the PHY.

To start the charger detection during normal operation, ensure that the PHY and charger are enabled and the automatic detection is turned on. Then, initiate a charger detection cycle by transitioning CHGDET\_RSTRT from 1 to 0. If a Charging Downstream Port or a Dedicated Charging Port is detected, the charger enable signal (USBx\_CE) will be driven high and remain high until the charger is disabled by either CHGDET\_DIS = 1 or CHGDET\_RSTRT=1. If the port remains unconnected after initiating the charger detect cycle, it will continue the detection until a charger is detected or an error condition occurs. Note that USBx\_CE is not an open drain output.

To disable the charger after successful detection, you must disable the charger detect circuitry with CHGDET\_DIS or CHGDET\_RSTRT, even if the charger is physically disconnected.

**Figure 9-2. USB Charger Detection**



Charger detection can be automatically started with no power to the rest of AM335x. If VDDA3P3V\_USBx is present, via an LDO powered by VBUS connected to a host, the charger detection state machine will automatically start and perform detection. If a charger is detected, USBx\_CE will be driven high, otherwise it will be driven low.

The charger detection circuitry performs the following steps of the Battery Charging specification v1.1:

1. VBUS Detect
2. Data Contact Detect
3. Primary Detection

Secondary Detection (to distinguish between a Charging Downstream Port and a Dedicated Charging Port) is a newly added feature of the v1.2 spec and is not implemented in the charger detection state machine.

#### 9.2.4.4.3 Ethernet MII Mode Selection

The control module provides a mechanism to select the Mode of operation of Ethernet MII interface. The GMII\_SEL register has register bit fields to select the MII/RMII/RGMII modes, clock sources, and delay mode.

#### 9.2.4.4.4 Ethernet Module Reset Isolation Control

This feature allows the device to undergo a warm reset without disrupting the switch or traffic being routed through the switch during the reset condition. The CPSW Reset Isolation register (RESET\_ISO) has an ISO\_CONTROL field which controls the reset isolation feature.

If the reset isolation is enabled, any warm reset source will be blocked to the EMAC switch. If the EMAC reset isolation is NOT active (default state), then the warm reset sources are allowed to propagate as normal including to the EMAC Switch module (both reset inputs to the IP). All cold or POR resets will always propagate to the EMAC switch module as normal.

When RESET\_ISO is enabled, the following registers will not be disturbed by a warm reset:

- GMII\_SEL
- CONF\_GPMC\_A[11:0]
- CONF\_GPMC\_WAIT0
- CONF\_GPMC\_WPN
- CONF\_GPMC\_BEN1
- CONF\_MII1\_COL
- CONF\_MII1\_CRS
- CONF\_MII1\_RX\_ER
- CONF\_MII1\_TX\_EN
- CONF\_MII1\_RX\_DV
- CONF\_MII1\_TXD[3:0]
- CONF\_MII1\_TX\_CLK
- CONF\_MII1\_RX\_CLK
- CONF\_MII1\_RXD[3:0]
- CONF\_RMII1\_REF\_CLK
- CONF\_MDIO\_DATA
- CONF\_MDIO\_CLK

#### 9.2.4.4.5 Timer/eCAP Event Capture Control

The Timer 5, 6, 7 events and the eCAP0, 1, 2 events can be selected using the TIMER\_EVT\_CAPTURE and ECAP\_EVT\_CAPTURE registers. The following table lists the available sources for those events.

**Table 9-5. Available Sources for Timer[5–7] and eCAP[0–2] Events**

| Event No. | Source module                                  | Interrupt Name/Pin |
|-----------|--|--------------------|
| 0         | For Timer 5 MUX input from IO signal<br>TIMER5 | TIMER5 IO pin      |
|           | For Timer 6 MUX input from IO signal<br>TIMER6 | TIMER6 IO pin      |
|           | For Timer 7 MUX input from IO signal<br>TIMER7 | TIMER7 IO pin      |
|           | For eCAP 0 MUX input from IO signal<br>eCAP0   | eCAP0 IO pin       |
|           | For eCAP 1 MUX input from IO signal<br>eCAP1   | eCAP1 IO pin       |
|           | For eCAP 2 MUX input from IO signal<br>eCAP2   | eCAP2 IO pin       |
| 1         | UART0  | UART0INT           |
| 2         | UART1  | UART1INT           |
| 3         | UART2  | UART2INT           |
| 4         | UART3  | UART3INT           |
| 5         | UART4  | UART4INT           |
| 6         | UART5  | UART5INT           |
| 7         | 3PGSW  | 3PGSWRXTTHR0       |
| 8         | 3PGSW  | 3PGSWRXINT0        |
| 9         | 3PGSW  | 3PGSWTXINT0        |
| 10        | 3PGSW  | 3PGSWMISC0         |
| 11        | McASP0   | MCATXINT0          |
| 12        | McASP0   | MCARXINT0          |
| 13        | McASP1   | MCATXINT1          |
| 14        | McASP1   | MCARXINT1          |
| 15        | Reserved                                       | Reserved           |
| 16        | Reserved                                       | Reserved           |
| 17        | GPIO 0   | GPIOINT0A          |
| 18        | GPIO 0   | GPIOINT0B          |
| 19        | GPIO 1   | GPIOINT1A          |
| 20        | GPIO 1   | GPIOINT1B          |
| 21        | GPIO 2   | GPIOINT2A          |
| 22        | GPIO 2   | GPIOINT2B          |
| 23        | GPIO 3   | GPIOINT3A          |
| 24        | GPIO 3   | GPIOINT3B          |
| 25        | DCAN0  | DCAN0_INT0         |
| 26        | DCAN0  | DCAN0_INT1         |
| 27        | DCAN0  | DCAN0_PARITY       |
| 28        | DCAN1  | DCAN1_INT0         |
| 29        | DCAN1  | DCAN1_INT1         |
| 30        | DCAN1  | DCAN1_PARITY       |



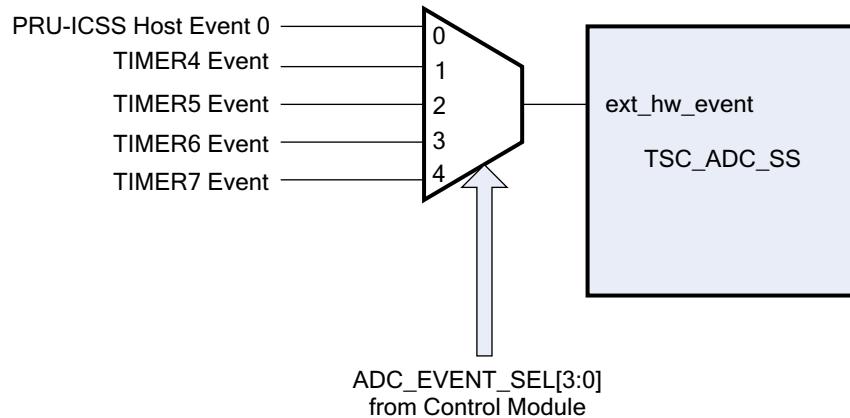
#### 9.2.4.4.6 ADC Capture Control

The following chip level events can be connected through the software-controlled multiplexer to the TSC\_ADC module.

1. PRU-ICSS Host Event 0
2. Timer 4 Event
3. Timer 5 Event
4. Timer 6 Event
5. Timer 7 Event

This pin is the external hardware trigger to start the ADC channel conversion. The ADC\_EVT\_CAPT register needs to be programmed to select the proper source for this conversion.

**Figure 9-3. Timer Events**



The following table contains the value to be programmed in the selection mux.

**Table 9-6. Selection Mux Values**

| ADC_EVENT_SEL Value | ADC External event selected |
|---------------------|-----------------------------|
| 000                 | PRU-ICSS Host Event 0       |
| 001                 | Timer 4 Event               |
| 010                 | Timer 5 Event               |
| 011                 | Timer 6 Event               |
| 100                 | Timer 7 Event               |
| 101-111             | Reserved                    |

## 9.2.5 DDR PHY

**Table 9-7. DDR Slew Rate Control Settings<sup>(1)(2)</sup>**

| sr0 | sr1 | Turn-on Time Level | Interface | Turn-on Time for Drv8 Setting (ps) <sup>(3)</sup> | Max Noise on IO Supply for Drv8 Setting <sup>(4)</sup> |
|-----|-----|--------------------|-----------|---|--|
| 0   | 0   | Fastest            | 1.5 V     | 117   | 0.43   |
|     |     |                    | 1.8 V     | 116   | 0.6  |
| 0   | 1   | Fast               | 1.5 V     | 430   | 0.17   |
|     |     |                    | 1.8 V     | 443   | 0.24   |
| 1   | 0   | Slow               | 1.5 V     | 530   | 0.15   |
|     |     |                    | 1.8 V     | 567   | 0.21   |
| 1   | 1   | Slowest            | 1.5 V     | 790   | 0.13   |
|     |     |                    | 1.8 V     | 815   | 0.18   |

<sup>(1)</sup> These values are programmed in the following registers: ddr\_cmd0\_ioctrl, ddr\_cmd1\_ioctrl, ddr\_cmd2\_ioctrl, ddr\_data0\_ioctrl, ddr\_data1\_ioctrl.

<sup>(2)</sup> Values for DDR\_CMDx\_IOCCTRL.io\_config\_sr\_clk should be programmed to the same value.

<sup>(3)</sup> Value is obtained at nominal PTV.

<sup>(4)</sup> Value is the average across PTVs.

**Table 9-8. DDR Impedance Control Settings<sup>(1)(2)</sup>**

| I2 | I1 | I0 | Drive Setting Name | Output Impedance (R <sub>on</sub> ) | Drive Strength  I <sub>OHL</sub>  ,  I <sub>OL</sub> | Example: R <sub>on</sub> for R <sub>ext</sub> = 50 ohms | Example:  I <sub>OHL</sub>  ,  I <sub>OL</sub>   for R <sub>ext</sub> = 50 ohms |
|----|----|----|--------------------|-------------------------------------|--|---|---|
| 0  | 0  | 0  | Drv5               | 1.6*R <sub>ext</sub>                | 0.625*I <sub>out</sub>                               | 80 ohms   | 5 mA  |
| 0  | 0  | 1  | Drv6               | 1.33*R <sub>ext</sub>               | 0.75*I <sub>out</sub>                                | 67 ohms   | 6 mA  |
| 0  | 1  | 0  | Drv7               | 1.14*R <sub>ext</sub>               | 0.875*I <sub>out</sub>                               | 57 ohms   | 7 mA  |
| 0  | 1  | 1  | Drv8               | R <sub>ext</sub>                    | I <sub>out</sub>                                     | 50 ohms   | 8 mA  |
| 1  | 0  | 0  | Drv9               | 0.88*R <sub>ext</sub>               | 1.125*I <sub>out</sub>                               | 44 ohms   | 9 mA  |
| 1  | 0  | 1  | Drv10              | 0.8*R <sub>ext</sub>                | 1.250*I <sub>out</sub>                               | 40 ohms   | 10 mA   |
| 1  | 1  | 0  | Drv11              | 0.73*R <sub>ext</sub>               | 1.375*I <sub>out</sub>                               | 36 ohms   | 11 mA   |
| 1  | 1  | 1  | Drv12              | 0.67*R <sub>ext</sub>               | 1.5*I <sub>out</sub>                                 | 33 ohms   | 12 mA   |

<sup>(1)</sup> These values are programmed in the following registers: ddr\_cmd0\_ioctrl, ddr\_cmd1\_ioctrl, ddr\_cmd2\_ioctrl, ddr\_data0\_ioctrl, ddr\_data1\_ioctrl.

<sup>(2)</sup> Values for DDR\_CMDx\_IOCCTRL.io\_config\_i\_clk should be programmed to the same value.

### 9.2.5.1 DDR PHY to IO Pin Mapping

The following table describes the DDR PHY to IO pin mapping.

**Table 9-9. DDR PHY to IO Pin Mapping**

| Macro Pin | CMD0    | CMD1     | CMD2       | DATA0    | DATA1    |
|-----------|---------|----------|------------|----------|----------|
| 0         | ddr_ba2 | Unconn   | ddr_cke    | ddr_d8   | ddr_d0   |
| 1         | ddr_wen | ddr_a15  | ddr_resetn | ddr_d9   | ddr_d1   |
| 2         | ddr_ba0 | ddr_a2   | ddr_odt    | ddr_d10  | ddr_d2   |
| 3         | ddr_a5  | ddr_a12  | Unconn     | ddr_d11  | ddr_d3   |
| 4         | ddr_ck  | ddr_a7   | ddr_a14    | ddr_d12  | ddr_d4   |
| 5         | ddr_ckn | ddr_ba1  | ddr_a13    | ddr_d13  | ddr_d5   |
| 6         | ddr_a3  | ddr_a10  | ddr_csn0   | ddr_d14  | ddr_d6   |
| 7         | ddr_a4  | ddr_a0   | Unconn     | ddr_d15  | ddr_d7   |
| 8         | ddr_a8  | ddr_a11  | ddr_a1     | ddr_dqm1 | ddr_dqm0 |
| 9         | ddr_a9  | ddr_casn | Unconn     | ddr_dqs1 | ddr_dqs0 |

**Table 9-9. DDR PHY to IO Pin Mapping (continued)**

| Macro Pin | CMD0   | CMD1     | CMD2   | DATA0     | DATA1     |
|-----------|--------|----------|--------|-----------|-----------|
| 10        | ddr_a6 | ddr_rasn | Unconn | ddr_dqsn1 | ddr_dqsn0 |

### 9.3 CONTROL\_MODULE Registers

**Table 9-10** lists the memory-mapped registers for the CONTROL\_MODULE. All other register offset addresses not listed in **Table 9-10** should be considered as reserved locations and the register contents should not be modified.

**Table 9-10. CONTROL\_MODULE REGISTERS**

| Offset | Acronym                   | Register Description | Section                        |
|--------|---------------------------|----------------------|--------------------------------|
| 0h     | control_revision          |                      | <a href="#">Section 9.3.1</a>  |
| 4h     | control_hwinfo            |                      | <a href="#">Section 9.3.3</a>  |
| 10h    | control_sysconfig         |                      | <a href="#">Section 9.3.4</a>  |
| 40h    | control_status            |                      | <a href="#">Section 9.3.5</a>  |
| 110h   | control_emif_sdram_config |                      | <a href="#">Section 9.3.6</a>  |
| 41Ch   | cortex_vbbldo_ctrl        |                      | <a href="#">Section 9.3.6</a>  |
| 428h   | core_sldo_ctrl            |                      | <a href="#">Section 9.3.7</a>  |
| 42Ch   | mpu_sldo_ctrl             |                      | <a href="#">Section 9.3.8</a>  |
| 444h   | clk32kdivratio_ctrl       |                      | <a href="#">Section 9.3.9</a>  |
| 448h   | bandgap_ctrl              |                      | <a href="#">Section 9.3.10</a> |
| 44Ch   | bandgap_trim              |                      | <a href="#">Section 9.3.11</a> |
| 458h   | pll_clkinpulow_ctrl       |                      | <a href="#">Section 9.3.12</a> |
| 468h   | mosc_ctrl                 |                      | <a href="#">Section 9.3.13</a> |
| 46Ch   | rcosc_ctrl                |                      | <a href="#">Section 9.3.14</a> |
| 470h   | deepsleep_ctrl            |                      | <a href="#">Section 9.3.15</a> |
| 600h   | device_id                 |                      | <a href="#">Section 9.3.2</a>  |
| 604h   | dev_feature               |                      | <a href="#">Section 9.3.16</a> |
| 608h   | init_priority_0           |                      | <a href="#">Section 9.3.17</a> |
| 60Ch   | init_priority_1           |                      | <a href="#">Section 9.3.18</a> |
| 610h   | mmu_cfg                   |                      | <a href="#">Section 9.3.19</a> |
| 614h   | tptc_cfg                  |                      | <a href="#">Section 9.3.20</a> |
| 620h   | usb_ctrl0                 |                      | <a href="#">Section 9.3.21</a> |
| 624h   | usb_sts0                  |                      | <a href="#">Section 9.3.22</a> |
| 628h   | usb_ctrl1                 |                      | <a href="#">Section 9.3.23</a> |
| 62Ch   | usb_sts1                  |                      | <a href="#">Section 9.3.24</a> |
| 630h   | mac_id0_lo                |                      | <a href="#">Section 9.3.25</a> |
| 634h   | mac_id0_hi                |                      | <a href="#">Section 9.3.26</a> |
| 638h   | mac_id1_lo                |                      | <a href="#">Section 9.3.27</a> |
| 63Ch   | mac_id1_hi                |                      | <a href="#">Section 9.3.28</a> |
| 644h   | dcan_raminit              |                      | <a href="#">Section 9.3.29</a> |
| 648h   | usb_wkup_ctrl             |                      | <a href="#">Section 9.3.30</a> |
| 650h   | gmii_sel                  |                      | <a href="#">Section 9.3.31</a> |
| 664h   | pwmss_ctrl                |                      | <a href="#">Section 9.3.32</a> |
| 670h   | mreqprio_0                |                      | <a href="#">Section 9.3.33</a> |
| 674h   | mreqprio_1                |                      | <a href="#">Section 9.3.34</a> |
| 690h   | hw_event_sel_grp1         |                      | <a href="#">Section 9.3.35</a> |
| 694h   | hw_event_sel_grp2         |                      | <a href="#">Section 9.3.36</a> |

**Table 9-10. CONTROL\_MODULE REGISTERS (continued)**

| Offset | Acronym            | Register Description   | Section        |
|--------|--------------------|--|----------------|
| 698h   | hw_event_sel_grp3  |  | Section 9.3.37 |
| 69Ch   | hw_event_sel_grp4  |  | Section 9.3.38 |
| 6A0h   | smrt_ctrl          |  | Section 9.3.39 |
| 6A4h   | mpuss_hw_debug_sel |  | Section 9.3.40 |
| 6A8h   | mpuss_hw_dbg_info  |  | Section 9.3.41 |
| 770h   | vdd_mpu_opp_050    |  | Section 9.3.42 |
| 774h   | vdd_mpu_opp_100    |  | Section 9.3.43 |
| 778h   | vdd_mpu_opp_120    |  | Section 9.3.44 |
| 77Ch   | vdd_mpu_opp_turbo  |  | Section 9.3.45 |
| 7B8h   | vdd_core_opp_050   |  | Section 9.3.46 |
| 7BCh   | vdd_core_opp_100   |  | Section 9.3.47 |
| 7D0h   | bb_scale           |  | Section 9.3.48 |
| 7F4h   | usb_vid_pid        |  | Section 9.3.49 |
| 800h   | conf_gpmc_ad0      | See the device datasheet for information on default pin mux configurations. Note that the device ROM may change the default pin mux for certain pins based on the SYSBOOT mode settings. | Section 9.3.50 |
| 804h   | conf_gpmc_ad1      |  | Section 9.3.50 |
| 808h   | conf_gpmc_ad2      |  | Section 9.3.50 |
| 80Ch   | conf_gpmc_ad3      |  | Section 9.3.50 |
| 810h   | conf_gpmc_ad4      |  | Section 9.3.50 |
| 814h   | conf_gpmc_ad5      |  | Section 9.3.50 |
| 818h   | conf_gpmc_ad6      |  | Section 9.3.50 |
| 81Ch   | conf_gpmc_ad7      |  | Section 9.3.50 |
| 820h   | conf_gpmc_ad8      |  | Section 9.3.50 |
| 824h   | conf_gpmc_ad9      |  | Section 9.3.50 |
| 828h   | conf_gpmc_ad10     |  | Section 9.3.50 |
| 82Ch   | conf_gpmc_ad11     |  | Section 9.3.50 |
| 830h   | conf_gpmc_ad12     |  | Section 9.3.50 |
| 834h   | conf_gpmc_ad13     |  | Section 9.3.50 |
| 838h   | conf_gpmc_ad14     |  | Section 9.3.50 |
| 83Ch   | conf_gpmc_ad15     |  | Section 9.3.50 |
| 840h   | conf_gpmc_a0       |  | Section 9.3.50 |
| 844h   | conf_gpmc_a1       |  | Section 9.3.50 |
| 848h   | conf_gpmc_a2       |  | Section 9.3.50 |
| 84Ch   | conf_gpmc_a3       |  | Section 9.3.50 |
| 850h   | conf_gpmc_a4       |  | Section 9.3.50 |
| 854h   | conf_gpmc_a5       |  | Section 9.3.50 |
| 858h   | conf_gpmc_a6       |  | Section 9.3.50 |
| 85Ch   | conf_gpmc_a7       |  | Section 9.3.50 |
| 860h   | conf_gpmc_a8       |  | Section 9.3.50 |
| 864h   | conf_gpmc_a9       |  | Section 9.3.50 |
| 868h   | conf_gpmc_a10      |  | Section 9.3.50 |
| 86Ch   | conf_gpmc_a11      |  | Section 9.3.50 |
| 870h   | conf_gpmc_wait0    |  | Section 9.3.50 |
| 874h   | conf_gpmc_wpn      |  | Section 9.3.50 |
| 878h   | conf_gpmc_ben1     |  | Section 9.3.50 |
| 87Ch   | conf_gpmc_csn0     |  | Section 9.3.50 |

**Table 9-10. CONTROL\_MODULE REGISTERS (continued)**

| Offset | Acronym             | Register Description | Section        |
|--------|---------------------|----------------------|----------------|
| 880h   | conf_gpmc_csn1      |                      | Section 9.3.50 |
| 884h   | conf_gpmc_csn2      |                      | Section 9.3.50 |
| 888h   | conf_gpmc_csn3      |                      | Section 9.3.50 |
| 88Ch   | conf_gpmc_clk       |                      | Section 9.3.50 |
| 890h   | conf_gpmc_advn_ale  |                      | Section 9.3.50 |
| 894h   | conf_gpmc_oen_ren   |                      | Section 9.3.50 |
| 898h   | conf_gpmc_wen       |                      | Section 9.3.50 |
| 89Ch   | conf_gpmc_ben0_cle  |                      | Section 9.3.50 |
| 8A0h   | conf_lcd_data0      |                      | Section 9.3.50 |
| 8A4h   | conf_lcd_data1      |                      | Section 9.3.50 |
| 8A8h   | conf_lcd_data2      |                      | Section 9.3.50 |
| 8ACh   | conf_lcd_data3      |                      | Section 9.3.50 |
| 8B0h   | conf_lcd_data4      |                      | Section 9.3.50 |
| 8B4h   | conf_lcd_data5      |                      | Section 9.3.50 |
| 8B8h   | conf_lcd_data6      |                      | Section 9.3.50 |
| 8BCh   | conf_lcd_data7      |                      | Section 9.3.50 |
| 8C0h   | conf_lcd_data8      |                      | Section 9.3.50 |
| 8C4h   | conf_lcd_data9      |                      | Section 9.3.50 |
| 8C8h   | conf_lcd_data10     |                      | Section 9.3.50 |
| 8CCh   | conf_lcd_data11     |                      | Section 9.3.50 |
| 8D0h   | conf_lcd_data12     |                      | Section 9.3.50 |
| 8D4h   | conf_lcd_data13     |                      | Section 9.3.50 |
| 8D8h   | conf_lcd_data14     |                      | Section 9.3.50 |
| 8DCh   | conf_lcd_data15     |                      | Section 9.3.50 |
| 8E0h   | conf_lcd_vsync      |                      | Section 9.3.50 |
| 8E4h   | conf_lcd_hsync      |                      | Section 9.3.50 |
| 8E8h   | conf_lcd_pclk       |                      | Section 9.3.50 |
| 8EcH   | conf_lcd_ac_bias_en |                      | Section 9.3.50 |
| 8F0h   | conf_mmc0_dat3      |                      | Section 9.3.50 |
| 8F4h   | conf_mmc0_dat2      |                      | Section 9.3.50 |
| 8F8h   | conf_mmc0_dat1      |                      | Section 9.3.50 |
| 8FCh   | conf_mmc0_dat0      |                      | Section 9.3.50 |
| 900h   | conf_mmc0_clk       |                      | Section 9.3.50 |
| 904h   | conf_mmc0_cmd       |                      | Section 9.3.50 |
| 908h   | conf_mii1_col       |                      | Section 9.3.50 |
| 90Ch   | conf_mii1_crs       |                      | Section 9.3.50 |
| 910h   | conf_mii1_rx_er     |                      | Section 9.3.50 |
| 914h   | conf_mii1_tx_en     |                      | Section 9.3.50 |
| 918h   | conf_mii1_rx_dv     |                      | Section 9.3.50 |
| 91Ch   | conf_mii1_txd3      |                      | Section 9.3.50 |
| 920h   | conf_mii1_txd2      |                      | Section 9.3.50 |
| 924h   | conf_mii1_txd1      |                      | Section 9.3.50 |
| 928h   | conf_mii1_txd0      |                      | Section 9.3.50 |
| 92Ch   | conf_mii1_tx_clk    |                      | Section 9.3.50 |
| 930h   | conf_mii1_rx_clk    |                      | Section 9.3.50 |
| 934h   | conf_mii1_rxd3      |                      | Section 9.3.50 |
| 938h   | conf_mii1_rxd2      |                      | Section 9.3.50 |

**Table 9-10. CONTROL\_MODULE REGISTERS (continued)**

| Offset | Acronym                | Register Description | Section        |
|--------|------------------------|----------------------|----------------|
| 93Ch   | conf_mii1_rx0d1        |                      | Section 9.3.50 |
| 940h   | conf_mii1_rx0d0        |                      | Section 9.3.50 |
| 944h   | conf_rmi1_ref_clk      |                      | Section 9.3.50 |
| 948h   | conf_mdio              |                      | Section 9.3.50 |
| 94Ch   | conf_mdc               |                      | Section 9.3.50 |
| 950h   | conf_spi0_sclk         |                      | Section 9.3.50 |
| 954h   | conf_spi0_d0           |                      | Section 9.3.50 |
| 958h   | conf_spi0_d1           |                      | Section 9.3.50 |
| 95Ch   | conf_spi0_cs0          |                      | Section 9.3.50 |
| 960h   | conf_spi0_cs1          |                      | Section 9.3.50 |
| 964h   | conf_ecap0_in_pwm0_out |                      | Section 9.3.50 |
| 968h   | conf_uart0_ctsn        |                      | Section 9.3.50 |
| 96Ch   | conf_uart0_rtsn        |                      | Section 9.3.50 |
| 970h   | conf_uart0_rx0         |                      | Section 9.3.50 |
| 974h   | conf_uart0_tx0         |                      | Section 9.3.50 |
| 978h   | conf_uart1_ctsn        |                      | Section 9.3.50 |
| 97Ch   | conf_uart1_rtsn        |                      | Section 9.3.50 |
| 980h   | conf_uart1_rx0         |                      | Section 9.3.50 |
| 984h   | conf_uart1_tx0         |                      | Section 9.3.50 |
| 988h   | conf_i2c0_sda          |                      | Section 9.3.50 |
| 98Ch   | conf_i2c0_scl          |                      | Section 9.3.50 |
| 990h   | conf_mcasp0_aclkx      |                      | Section 9.3.50 |
| 994h   | conf_mcasp0_fsx        |                      | Section 9.3.50 |
| 998h   | conf_mcasp0_axr0       |                      | Section 9.3.50 |
| 99Ch   | conf_mcasp0_ahclkx     |                      | Section 9.3.50 |
| 9A0h   | conf_mcasp0_aclkx      |                      | Section 9.3.50 |
| 9A4h   | conf_mcasp0_fsr        |                      | Section 9.3.50 |
| 9A8h   | conf_mcasp0_axr1       |                      | Section 9.3.50 |
| 9ACh   | conf_mcasp0_ahclkx     |                      | Section 9.3.50 |
| 9B0h   | conf_xdma_event_intr0  |                      | Section 9.3.50 |
| 9B4h   | conf_xdma_event_intr1  |                      | Section 9.3.50 |
| 9B8h   | conf_warmrstn          |                      | Section 9.3.50 |
| 9BCh   | conf_pwronrstn         |                      | Section 9.3.50 |
| 9C0h   | conf_nnm               |                      | Section 9.3.50 |
| 9C4h   | conf_xtalin            |                      | Section 9.3.50 |
| 9C8h   | conf_xtalout           |                      | Section 9.3.50 |
| 9D0h   | conf_tms               |                      | Section 9.3.50 |
| 9D4h   | conf_tdi               |                      | Section 9.3.50 |
| 9D8h   | conf_tdo               |                      | Section 9.3.50 |
| 9DCh   | conf_tck               |                      | Section 9.3.50 |
| 9E0h   | conf_trstn             |                      | Section 9.3.50 |
| 9E4h   | conf_emu0              |                      | Section 9.3.50 |
| 9E8h   | conf_emu1              |                      | Section 9.3.50 |
| 9ECh   | conf_rtc_xtalin        |                      | Section 9.3.50 |
| 9F0h   | conf_rtc_xtalout       |                      | Section 9.3.50 |
| 9F8h   | conf_rtc_pwronrstn     |                      | Section 9.3.50 |
| 9FCh   | conf_pmic_power_en     |                      | Section 9.3.50 |

**Table 9-10. CONTROL\_MODULE REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b>     | <b>Register Description</b> | <b>Section</b> |
|---------------|--------------------|-----------------------------|----------------|
| A00h          | conf_ext_wakeup    |                             | Section 9.3.50 |
| A04h          | conf_rtc_kaldo_enn |                             | Section 9.3.50 |
| A08h          | conf_usb0_dm       |                             | Section 9.3.50 |
| A0Ch          | conf_usb0_dp       |                             | Section 9.3.50 |
| A10h          | conf_usb0_ce       |                             | Section 9.3.50 |
| A14h          | conf_usb0_id       |                             | Section 9.3.50 |
| A18h          | conf_usb0_vbus     |                             | Section 9.3.50 |
| A1Ch          | conf_usb0_drvvbus  |                             | Section 9.3.50 |
| A20h          | conf_usb1_dm       |                             | Section 9.3.50 |
| A24h          | conf_usb1_dp       |                             | Section 9.3.50 |
| A28h          | conf_usb1_ce       |                             | Section 9.3.50 |
| A2Ch          | conf_usb1_id       |                             | Section 9.3.50 |
| A30h          | conf_usb1_vbus     |                             | Section 9.3.50 |
| A34h          | conf_usb1_drvvbus  |                             | Section 9.3.50 |
| E00h          | cqdetect_status    |                             | Section 9.3.51 |
| E04h          | ddr_io_ctrl        |                             | Section 9.3.52 |
| E0Ch          | vtp_ctrl           |                             | Section 9.3.53 |
| E14h          | vref_ctrl          |                             | Section 9.3.54 |
| F90h          | tpcc_evt_mux_0_3   |                             | Section 9.3.55 |
| F94h          | tpcc_evt_mux_4_7   |                             | Section 9.3.56 |
| F98h          | tpcc_evt_mux_8_11  |                             | Section 9.3.57 |
| F9Ch          | tpcc_evt_mux_12_15 |                             | Section 9.3.58 |
| FA0h          | tpcc_evt_mux_16_19 |                             | Section 9.3.59 |
| FA4h          | tpcc_evt_mux_20_23 |                             | Section 9.3.60 |
| FA8h          | tpcc_evt_mux_24_27 |                             | Section 9.3.61 |
| FACh          | tpcc_evt_mux_28_31 |                             | Section 9.3.62 |
| FB0h          | tpcc_evt_mux_32_35 |                             | Section 9.3.63 |
| FB4h          | tpcc_evt_mux_36_39 |                             | Section 9.3.64 |
| FB8h          | tpcc_evt_mux_40_43 |                             | Section 9.3.65 |
| FBCh          | tpcc_evt_mux_44_47 |                             | Section 9.3.66 |
| FC0h          | tpcc_evt_mux_48_51 |                             | Section 9.3.67 |
| FC4h          | tpcc_evt_mux_52_55 |                             | Section 9.3.68 |
| FC8h          | tpcc_evt_mux_56_59 |                             | Section 9.3.69 |
| FCCh          | tpcc_evt_mux_60_63 |                             | Section 9.3.70 |
| FD0h          | timer_evt_capt     |                             | Section 9.3.71 |
| FD4h          | ecap_evt_capt      |                             | Section 9.3.72 |
| FD8h          | adc_evt_capt       |                             | Section 9.3.73 |
| 1000h         | reset_iso          |                             | Section 9.3.74 |
| 131Ch         | ddr_cke_ctrl       |                             | Section 9.3.75 |
| 1320h         | sma2               |                             | Section 9.3.76 |
| 1324h         | m3_txev_eoi        |                             | Section 9.3.77 |
| 1328h         | ipc_msg_reg0       |                             | Section 9.3.78 |
| 132Ch         | ipc_msg_reg1       |                             | Section 9.3.79 |
| 1330h         | ipc_msg_reg2       |                             | Section 9.3.80 |
| 1334h         | ipc_msg_reg3       |                             | Section 9.3.81 |
| 1338h         | ipc_msg_reg4       |                             | Section 9.3.82 |
| 133Ch         | ipc_msg_reg5       |                             | Section 9.3.83 |

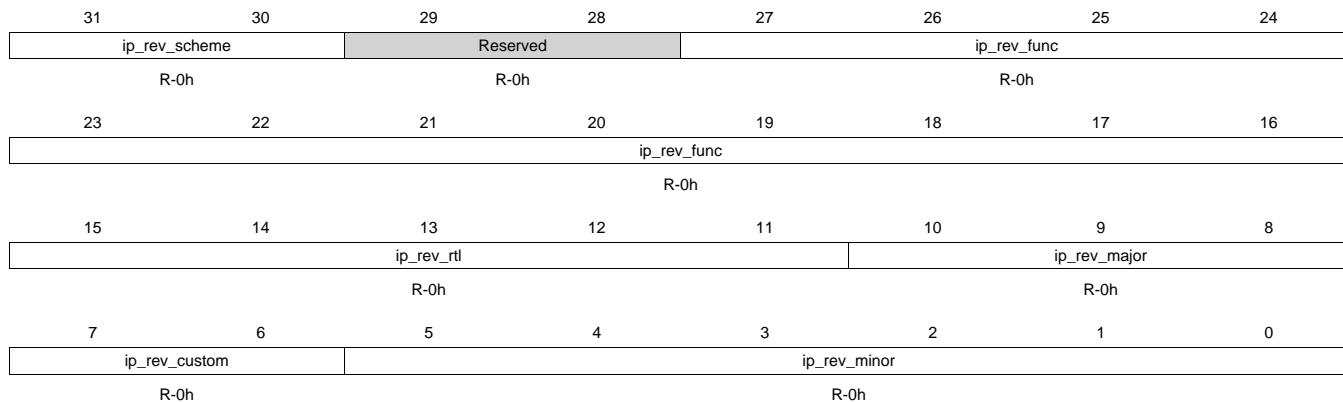
**Table 9-10. CONTROL\_MODULE REGISTERS (continued)**

| Offset | Acronym         | Register Description | Section                        |
|--------|-----------------|----------------------|--------------------------------|
| 1340h  | ipc_msg_reg6    |                      | <a href="#">Section 9.3.84</a> |
| 1344h  | ipc_msg_reg7    |                      | <a href="#">Section 9.3.85</a> |
| 1404h  | ddr_cmd0_ioctl  |                      | <a href="#">Section 9.3.86</a> |
| 1408h  | ddr_cmd1_ioctl  |                      | <a href="#">Section 9.3.87</a> |
| 140Ch  | ddr_cmd2_ioctl  |                      | <a href="#">Section 9.3.88</a> |
| 1440h  | ddr_data0_ioctl |                      | <a href="#">Section 9.3.89</a> |
| 1444h  | ddr_data1_ioctl |                      | <a href="#">Section 9.3.90</a> |

### 9.3.1 control\_revision Register (offset = 0h) [reset = 0h]

control\_revision is shown in [Figure 9-4](#) and described in [Table 9-11](#).

**Figure 9-4. control\_revision Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

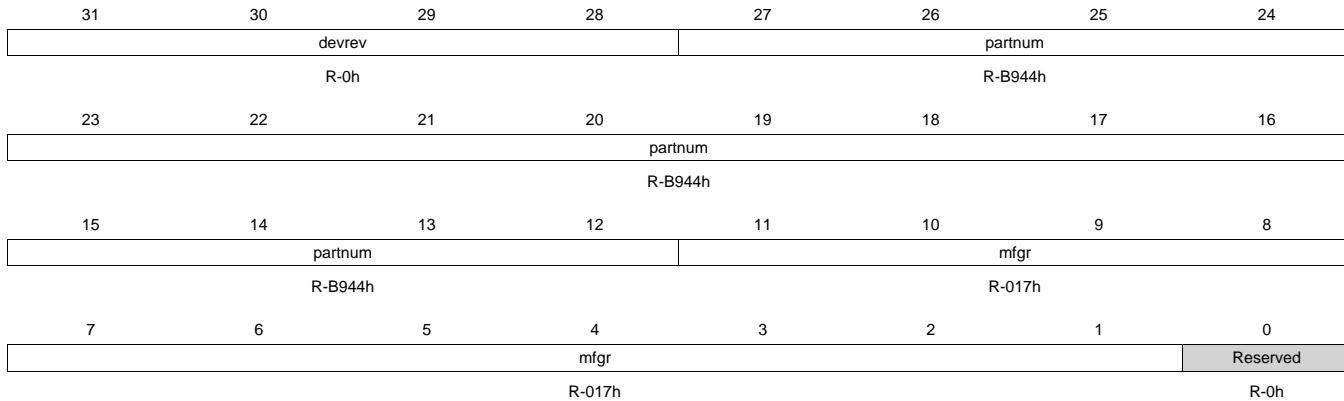
**Table 9-11. control\_revision Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31-30 | ip_rev_scheme | R    | 0h    | 01 - New Scheme  |
| 29-28 | Reserved      | R    | 0h    |  |
| 27-16 | ip_rev_func   | R    | 0h    | Function indicates a software compatible module family.<br>If there is no level of software compatibility a new Func number (and hence REVISION) should be assigned.           |
| 15-11 | ip_rev_rtl    | R    | 0h    | RTL Version (R).   |
| 10-8  | ip_rev_major  | R    | 0h    | Major Revision (X).  |
| 7-6   | ip_rev_custom | R    | 0h    | Indicates a special version for a particular device. Consequence of use may avoid use of standard Chip Support Library (CSL) / Drivers<br>- 00: Non custom (standard) revision |
| 5-0   | ip_rev_minor  | R    | 0h    | Minor Revision (Y).  |

### 9.3.2 device\_id Register (offset = 600h) [reset = 0x]

device\_id is shown in [Figure 9-5](#) and described in [Table 9-12](#).

**Figure 9-5. device\_id Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-12. device\_id Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-28 | devrev   | R    | 0h    | Device revision (See device errata for more information on device revision.<br>Reset value is revision-dependent. |
| 27-12 | partnum  | R    | B944h | Device part number (unique JTAG ID)   |
| 11-1  | mfgr     | R    | 017h  | Manufacturer's JTAG ID  |
| 0     | Reserved | R    | 0h    |   |

### **9.3.3 control\_hwinfo Register (offset = 4h) [reset = 0h]**

control\_hwinfo is shown in [Figure 9-6](#) and described in [Table 9-13](#).

**Figure 9-6. control\_hwinfo Register**

|           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31        | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ip_hwinfo |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

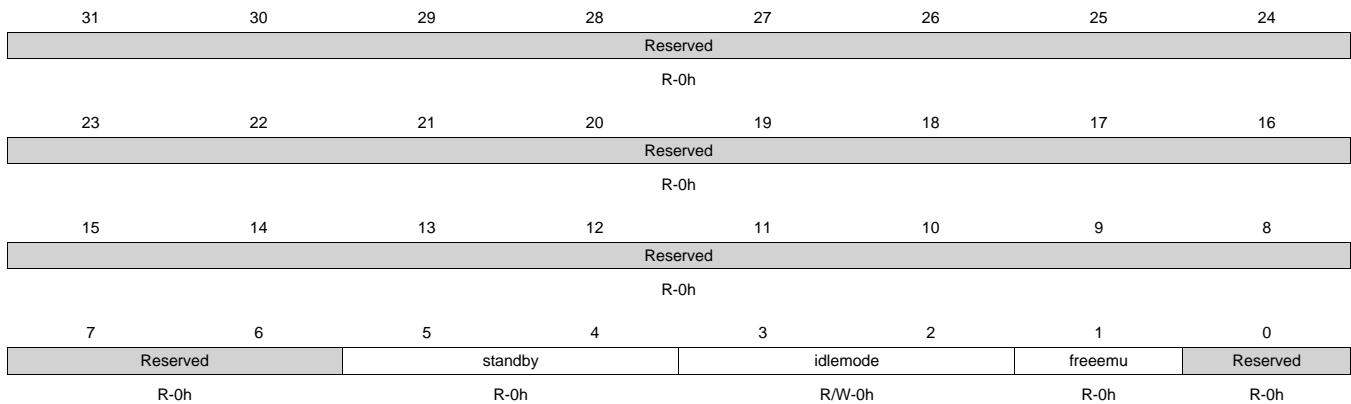
**Table 9-13. control\_hwinfo Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description         |
|------|-----------|------|-------|---------------------|
| 31-0 | ip_hwinfo | R    | 0h    | IP Module dependent |

### 9.3.4 control\_sysconfig Register (offset = 10h) [reset = 0h]

control\_sysconfig is shown in [Figure 9-7](#) and described in [Table 9-14](#).

**Figure 9-7. control\_sysconfig Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

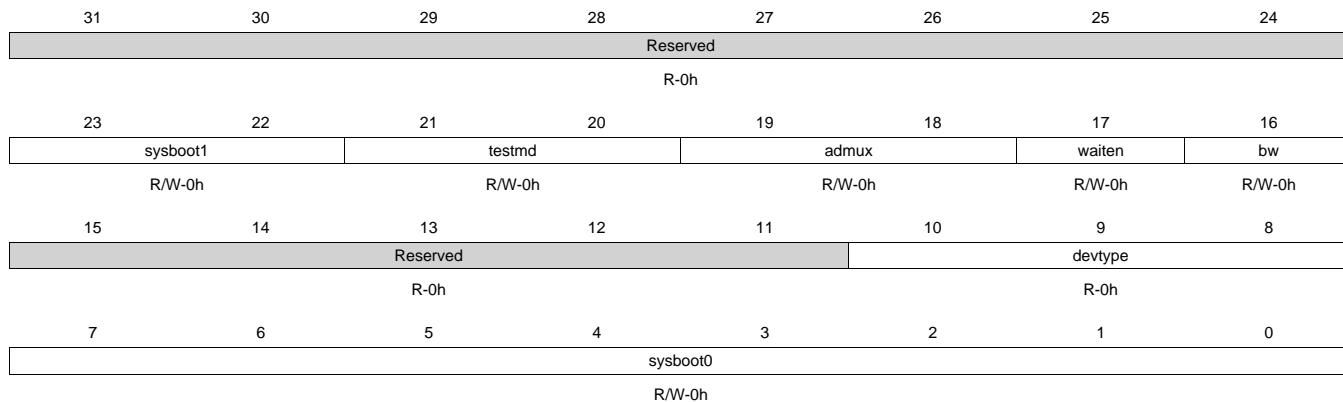
**Table 9-14. control\_sysconfig Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-6 | Reserved | R    | 0h    |  |
| 5-4  | standby  | R    | 0h    | Configure local initiator state management<br>00: Force Standby<br>01: No Standby Mode<br>10: Smart Standby<br>11: Smart Standby wakeup capable<br>Reserved in Control Module since it has no local initiator. |
| 3-2  | idlemode | R/W  | 0h    | Configure local target state management<br>00: Force Idle<br>01: No Idle<br>10: Smart Idle<br>11: Smart Idle wakeup capable  |
| 1    | freeemu  | R    | 0h    | Sensitivity to Emulation suspend input.<br>0: Module is sensitive to EMU suspend<br>1: Module not sensitive to EMU suspend   |
| 0    | Reserved | R    | 0h    |  |

### 9.3.5 control\_status Register (offset = 40h) [reset = 0h]

control\_status is shown in [Figure 9-8](#) and described in [Table 9-15](#).

**Figure 9-8. control\_status Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-15. control\_status Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | Reserved | R    | 0h    |  |
| 23-22 | sysboot1 | R/W  | 0h    | Used to select crystal clock frequency.<br>See SYSBOOT Configuration Pins.<br>Reset value is from SYSBOOT[15:14].  |
| 21-20 | testmd   | R/W  | 0h    | Set to 00b.<br>See SYSBOOT Configuration Pins for more information.<br>Reset value is from SYSBOOT[13:12].   |
| 19-18 | admux    | R/W  | 0h    | GPMC CS0 Default Address Muxing<br>00: No Addr/Data Muxing<br>01: Addr/Addr/Data Muxing<br>10: Addr/Data Muxing<br>11: Reserved<br>Reset value is from SYSBOOT[11:10]. |
| 17    | waiten   | R/W  | 0h    | GPMC CS0 Default Wait Enable<br>0: Ignore WAIT input<br>1: Use WAIT input<br>See SYSBOOT Configuration Pins for more information. Reset value is from SYSBOOT[9].      |
| 16    | bw       | R/W  | 0h    | GPMC CS0 Default Bus Width<br>0: 8-bit data bus<br>1: 16-bit data bus<br>See SYSBOOT Configuration Pins for more information. Reset value is from SYSBOOT[8].          |
| 15-11 | Reserved | R    | 0h    |  |
| 10-8  | devtype  | R    | 0h    | 000: Reserved<br>001: Reserved<br>010: Reserved<br>011: General Purpose (GP) Device<br>111: Reserved   |
| 7-0   | sysboot0 | R/W  | 0h    | Selected boot mode.<br>See SYSBOOT Configuration Pins for more information.<br>Reset value is from SYSBOOT[7:0].   |

### 9.3.6 control\_emif\_sdram\_config Register (offset = 110h) [reset = 0h]

The CONTROL\_EMIF\_SDRAM\_CONFIG register exports SDRAM configuration information to the EMIF after resuming from low power scenarios.

control\_emif\_sdram\_config is shown in [Figure 9-9](#) and described in [Table 9-16](#).

**Figure 9-9. control\_emif\_sdram\_config Register**

| 31          | 30         | 29    | 28        | 27          | 26       | 25       | 24 |
|-------------|------------|-------|-----------|-------------|----------|----------|----|
|             | SDRAM_TYPE |       | IBANK_POS |             | DDR_TERM |          |    |
| R/W-0h      | R/W-0h     |       |           | R/W-0h      | R/W-0h   |          |    |
| 23          | 22         | 21    | 20        | 19          | 18       | 17       | 16 |
| Reserved    | DYN_ODT    |       | Reserved  | SDRAM_DRIVE |          | CWL      |    |
| R-0h        | R/W-0h     |       |           | R-0h        | R/W-0h   |          |    |
| 15          | 14         | 13    | 12        | 11          | 10       | 9        | 8  |
| NARROW_MODE |            |       | CL        |             | ROWSIZE  |          |    |
| R/W-0h      | R/W-0h     |       |           | R/W-0h      | R/W-0h   |          |    |
| 7           | 6          | 5     | 4         | 3           | 2        | 1        | 0  |
| ROWSIZE     |            | IBANK |           | EBANK       |          | PAGESIZE |    |
| R/W-0h      | R/W-0h     |       |           | R/W-0h      | R/W-0h   |          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-16. control\_emif\_sdram\_config Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-29 | SDRAM_TYPE | R/W  | 0h    | SDRAM Type selection<br>000 – Reserved<br>001 – LPDDR1<br>010 – DDR2<br>011 – DDR3<br>100 – Reserved<br>101 – Reserved<br>110 – Reserved<br>111 – Reserved   |
| 28-27 | IBANK_POS  | R/W  | 0h    | Internal bank position.<br>00 - All Bank Address bits assigned from OCP address above column address bits.<br>01 – Bank Address bits [1:0] assigned from OCP address above column address bits and bit [2] from OCP address bits above row address bits.<br>10 – Bank Address bit [0] assigned from OCP address above column address bits and bit [2:1] from OCP address bits above row address bits.<br>11 – All Bank Address bits assigned from OCP address bits above row address bits. |
| 26-24 | DDR_TERM   | R/W  | 0h    | DDR2 and DDR3 termination resistor value. Set to 0 to disable termination.<br>For DDR2, set to 1 for 75 ohm, set to 2 for 150 ohm, and set to 3 for 50 ohm.<br>For DDR3, set to 1 for RZQ/4, set to 2 for RZQ/2, set to 3 for RZQ/6, set to 4 for RZQ/12, and set to 5 for RZQ/8.<br>All other values are reserved.  |
| 23    | Reserved   | R    | 0h    | Reserved. Read returns 0.  |
| 22-21 | DYN_ODT    | R/W  | 0h    | DDR3 Dynamic ODT.<br>Set to 0 to turn off dynamic ODT.<br>Set to 1 for RZQ/4 and set to 2 for RZQ/2.<br>All other values are reserved.   |

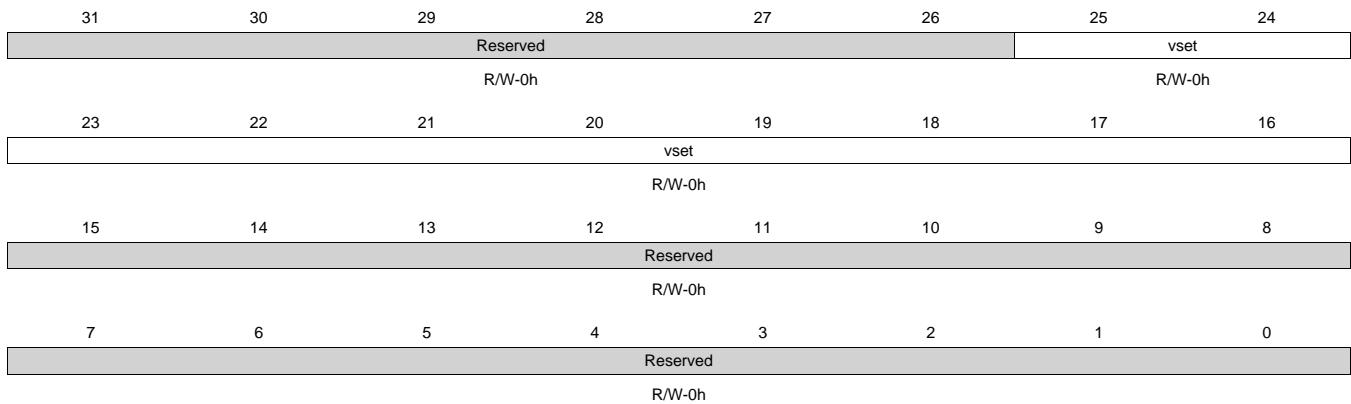
**Table 9-16. control\_emif\_sdram\_config Register Field Descriptions (continued)**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 20    | Reserved    | R    | 0h    | Reserved. Read returns 0.   |
| 19-18 | SDRAM_DRIVE | R/W  | 0h    | SDRAM drive strength.<br>For DDR2, set to 0 for normal, and set to 1 for weak drive strength.<br>For DDR3, set to 0 for RZQ/6 and set to 1 for RZQ/7.<br>All other values are reserved.   |
| 17-16 | CWL         | R/W  | 0h    | DDR3 CAS Write latency. Value of 0, 1, 2, and 3 (CAS write latency of 5, 6, 7, and 8) are supported. Use the lowest value supported for best performance. All other values are reserved.  |
| 15-14 | NARROW_MODE | R/W  | 0h    | SDRAM data bus width. set to 1 for 32-bit, and set to 2 for 16-bit. All other values are reserved.  |
| 13-10 | CL          | R/W  | 0h    | CAS Latency. The value of this field defines the CAS latency to be used when accessing connected SDRAM devices. Value of 2, 3, 4, and 5 (CAS latency of 2, 3, 4, and 5) are supported for DDR2. Value of 2, 4, 6, 8, 10, 12, and 14 (CAS latency of 5, 6, 7, 8, 9, 10, and 11) are supported for DDR3. All other values are reserved.   |
| 9-7   | ROWSIZE     | R/W  | 0h    | Row Size. Defines the number of row address bits of connected SDRAM devices. Set to 0 for 9 row bits, set to 1 for 10 row bits, set to 2 for 11 row bits, set to 3 for 12 row bits, set to 4 for 13 row bits, set to 5 for 14 row bits, set to 6 for 15 row bits, and set to 7 for 16 row bits. This field is only used when ibank_pos field in SDRAM Config register is set to 1, 2, or 3. |
| 6-4   | IBANK       | R/W  | 0h    | Internal Bank setup. Defines number of banks inside connected SDRAM devices. Set to 0 for 1 bank, set to 1 for 2 banks, set to 2 for 4 banks, and set to 3 for 8 banks. All other values are reserved.  |
| 3     | EBANK       | R/W  | 0h    | External chip select setup. Defines whether SDRAM accesses will use 1 or 2 chip select lines. Set to 0 to use pad_cs_o_n[0] only. Set to 1 to use pad_cs_o_n[1:0].  |
| 2-0   | PAGESIZE    | R/W  | 0h    | Page Size. Defines the internal page size of connected SDRAM devices. Set to 0 for 256-word page (8 column bits), set to 1 for 512-word page (9 column bits), set to 2 for 1024-word page (10 column bits), and set to 3 for 2048-word page (11 column bits). All other values are reserved.  |

### 9.3.7 core\_sldo\_ctrl Register (offset = 428h) [reset = 0h]

core\_sldo\_ctrl is shown in [Figure 9-10](#) and described in [Table 9-17](#).

**Figure 9-10. core\_sldo\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

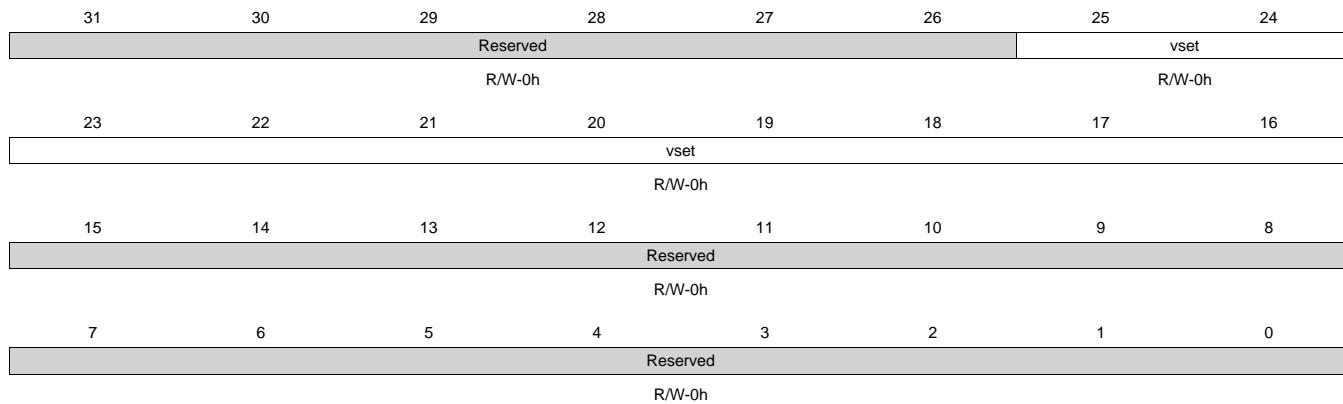
**Table 9-17. core\_sldo\_ctrl Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description |
|-------|----------|------|-------|-------------|
| 31-26 | Reserved | R/W  | 0h    |             |
| 25-16 | vset     | R/W  | 0h    | Trims VDDAR |
| 15-0  | Reserved | R/W  | 0h    |             |

### **9.3.8 mpu\_sldo\_ctrl Register (offset = 42Ch) [reset = 0h]**

mpu\_sldo\_ctrl is shown in [Figure 9-11](#) and described in [Table 9-18](#).

**Figure 9-11. mpu\_sldo\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

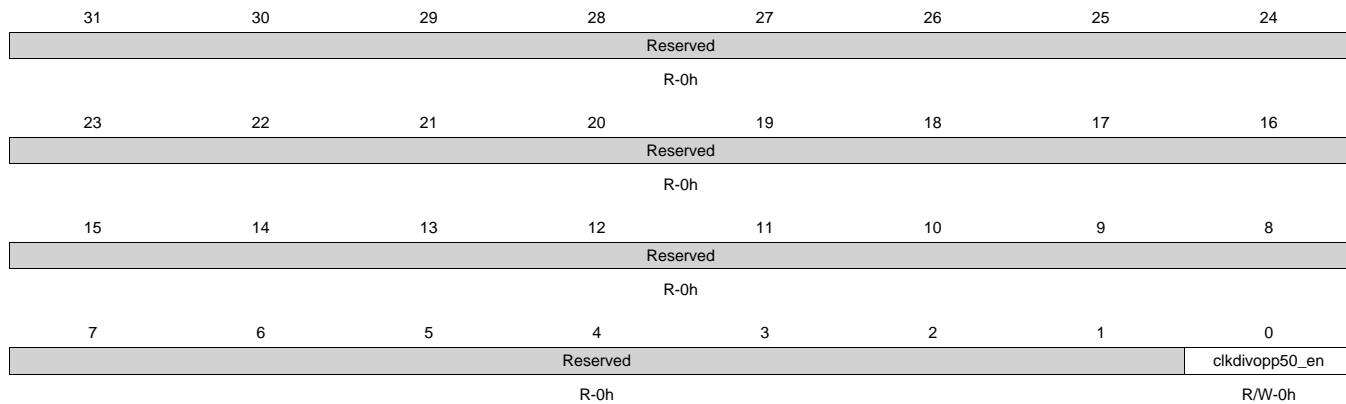
**Table 9-18. mpu\_sldo\_ctrl Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description |
|-------|----------|------|-------|-------------|
| 31-26 | Reserved | R/W  | 0h    |             |
| 25-16 | vset     | R/W  | 0h    | Trims VDDAR |
| 15-0  | Reserved | R/W  | 0h    |             |

### 9.3.9 clk32kdivratio\_ctrl Register (offset = 444h) [reset = 0h]

clk32kdivratio\_ctrl is shown in [Figure 9-12](#) and described in [Table 9-19](#).

**Figure 9-12. clk32kdivratio\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

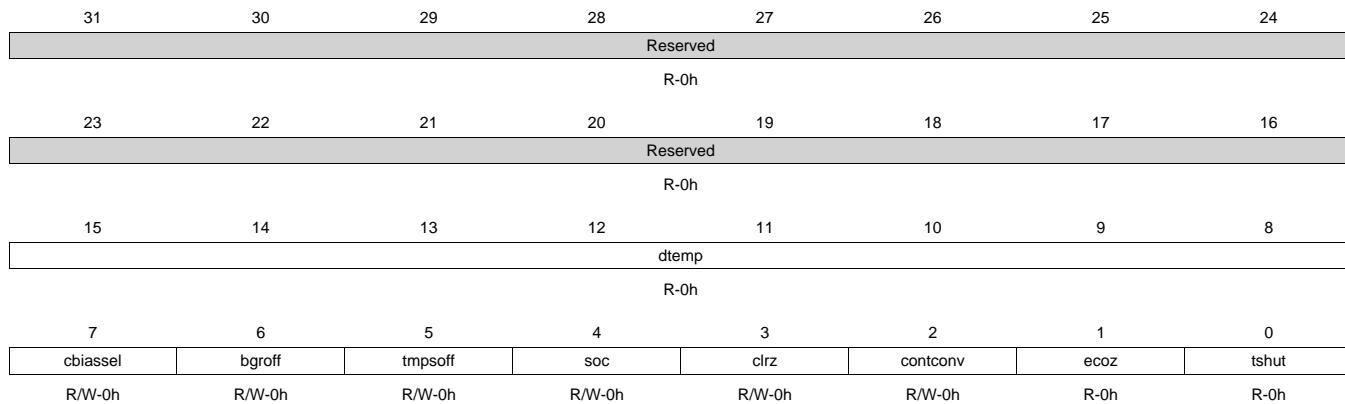
**Table 9-19. clk32kdivratio\_ctrl Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-1 | Reserved       | R    | 0h    |   |
| 0    | clkdivopp50_en | R/W  | 0h    | 0 : OPP100 operation, use ratio for 24MHz to 32KHz division<br>1 : OPP50 operation, use ratio for 12MHz to 32KHz division |

### 9.3.10 bandgap\_ctrl Register (offset = 448h) [reset = 0h]

bandgap\_ctrl is shown in [Figure 9-13](#) and described in [Table 9-20](#).

**Figure 9-13. bandgap\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-20. bandgap\_ctrl Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15-8  | dtemp    | R    | 0h    | Temperature data from ADC.<br>To be used when end of conversion (EOCZ) is 0.         |
| 7     | cbiasel  | R/W  | 0h    | 0: Select bandgap voltage as reference<br>1: Select resistor divider as reference    |
| 6     | bgroff   | R/W  | 0h    | 0: Normal operation<br>1: Bandgap is OFF (OFF Mode)                                  |
| 5     | tmpsoff  | R/W  | 0h    | 0: Normal operation<br>1: Temperature sensor is off and thermal shutdown in OFF mode |
| 4     | soc      | R/W  | 0h    | ADC start of conversion.<br>Transition to high starts a new ADC conversion cycle.    |
| 3     | clrz     | R/W  | 0h    | 0: Resets the digital outputs  |
| 2     | contconv | R/W  | 0h    | 0: ADC single conversion mode<br>1: ADC continuous conversion mode                   |
| 1     | ecoz     | R    | 0h    | ADC end of conversion<br>0: End of conversion<br>1: Conversion in progress           |
| 0     | tshut    | R    | 0h    | 0: Normal operation<br>1: Thermal shutdown event (greater than 147C)                 |

### 9.3.11 bandgap\_trim Register (offset = 44Ch) [reset = 0h]

bandgap\_trim is shown in [Figure 9-14](#) and described in [Table 9-21](#).

**Figure 9-14. bandgap\_trim Register**

|           |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|
| 31        | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| dtrbgapc  |    |    |    |    |    |    |    |
| R/W-0h    |    |    |    |    |    |    |    |
| 23        | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| dtrgapv   |    |    |    |    |    |    |    |
| R/W-0h    |    |    |    |    |    |    |    |
| 15        | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| dtrtemp   |    |    |    |    |    |    |    |
| R/W-0h    |    |    |    |    |    |    |    |
| 7         | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| dtrtempsc |    |    |    |    |    |    |    |
| R/W-0h    |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

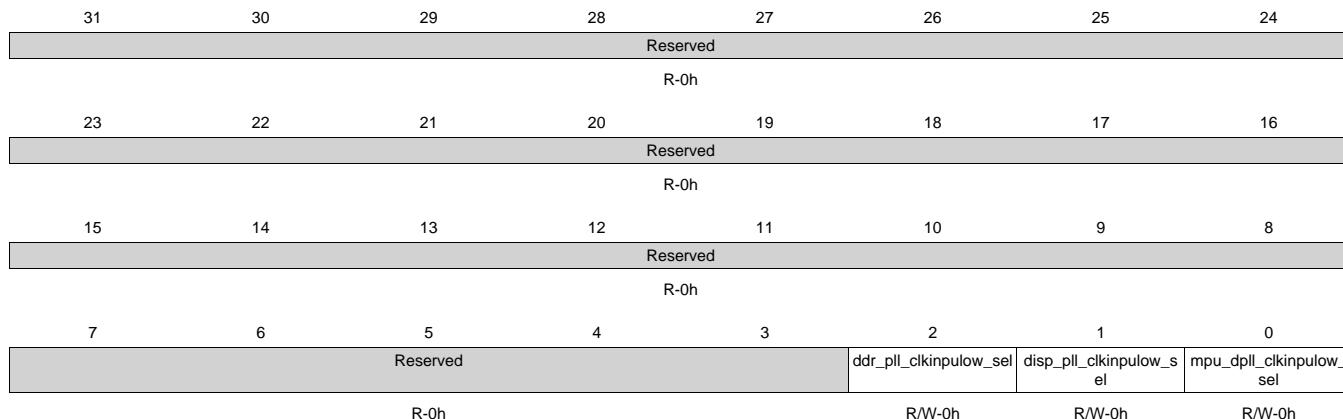
**Table 9-21. bandgap\_trim Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description                        |
|-------|-----------|------|-------|------------------------------------|
| 31-24 | dtrbgapc  | R/W  | 0h    | trim the output voltage of bandgap |
| 23-16 | dtrgapv   | R/W  | 0h    | trim the output voltage of bandgap |
| 15-8  | dtrtemp   | R/W  | 0h    | trim the temperature sensor        |
| 7-0   | dtrtempsc | R/W  | 0h    | trim the temperature sensor        |

### **9.3.12 pll\_clkinpulow\_ctrl Register (offset = 458h) [reset = 0h]**

pll\_clkinpulow\_ctrl is shown in [Figure 9-15](#) and described in [Table 9-22](#).

**Figure 9-15. pll\_clkinpulow\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

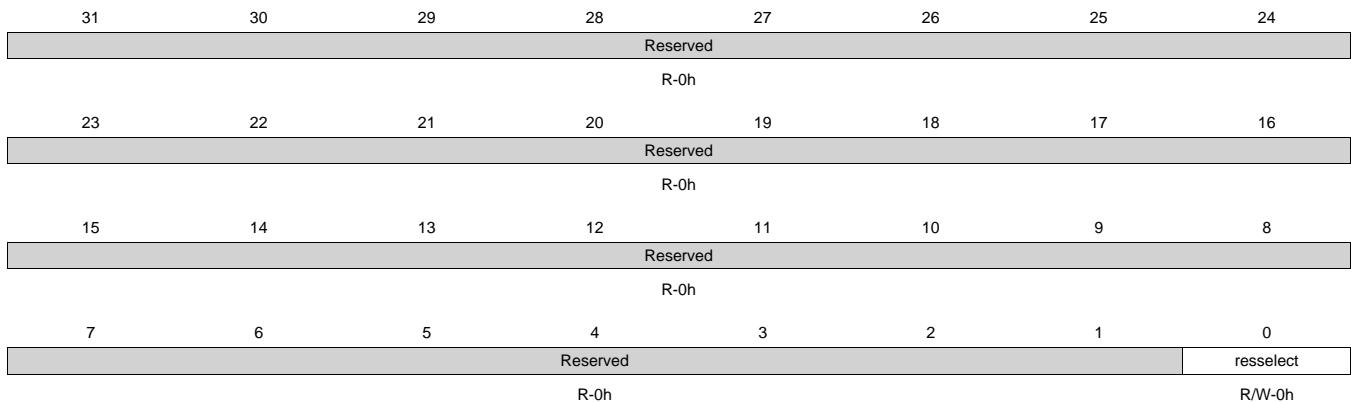
**Table 9-22. pll\_clkinpulow\_ctrl Register Field Descriptions**

| Bit  | Field                   | Type | Reset | Description   |
|------|-------------------------|------|-------|---|
| 31-3 | Reserved                | R    | 0h    |   |
| 2    | ddr_pll_clkinpulow_sel  | R/W  | 0h    | 0 : Select CORE_CLKOUT_M6 clock as CLKINPULOW<br>1 : Select PER_CLKOUT_M2 clock as CLKINPULOW |
| 1    | disp_pll_clkinpulow_sel | R/W  | 0h    | 0 : Select CORE_CLKOUT_M6 clock as CLKINPULOW<br>1 : Select PER_CLKOUT_M2 clock as CLKINPULOW |
| 0    | mpu_dpll_clkinpulow_sel | R/W  | 0h    | 0 : Select CORE_CLKOUT_M6 clock as CLKINPULOW<br>1 : Select PER_CLKOUT_M2 clock as CLKINPULOW |

### 9.3.13 mosc\_ctrl Register (offset = 468h) [reset = 0h]

mosc\_ctrl is shown in [Figure 9-16](#) and described in [Table 9-23](#).

**Figure 9-16. mosc\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

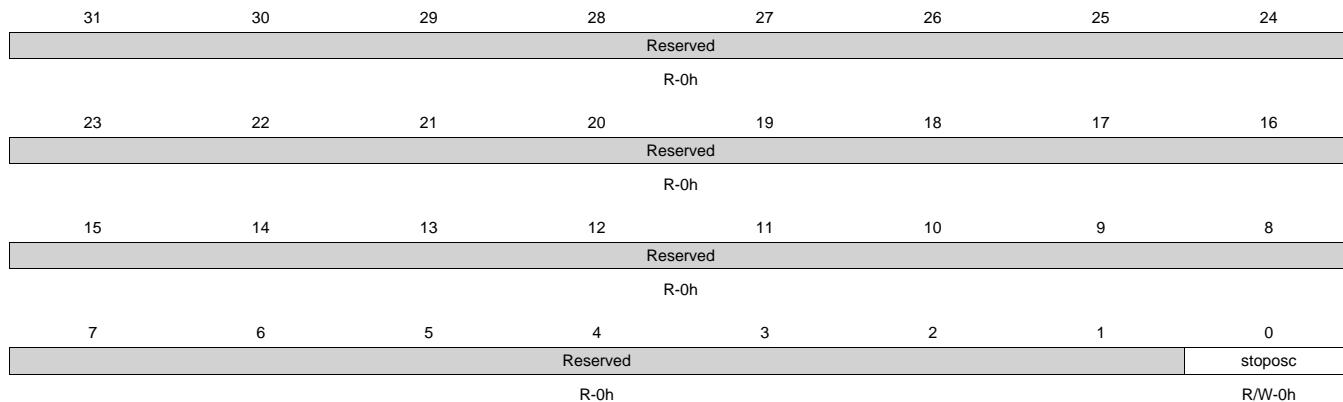
**Table 9-23. mosc\_ctrl Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-1 | Reserved  | R    | 0h    |  |
| 0    | resselect | R/W  | 0h    | For oscillation with a crystal while RESSELECT is 1, an external resistor must be connected between padxi and padxo to provide bias.<br>0: 1-MHz ohm resistor is connected between padxi and padxo for oscillator bias<br>1: Internal resistor is disconnected |

### **9.3.14 rcosc\_ctrl Register (offset = 46Ch) [reset = 0h]**

rcosc\_ctrl is shown in [Figure 9-17](#) and described in [Table 9-24](#).

**Figure 9-17. rcosc\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

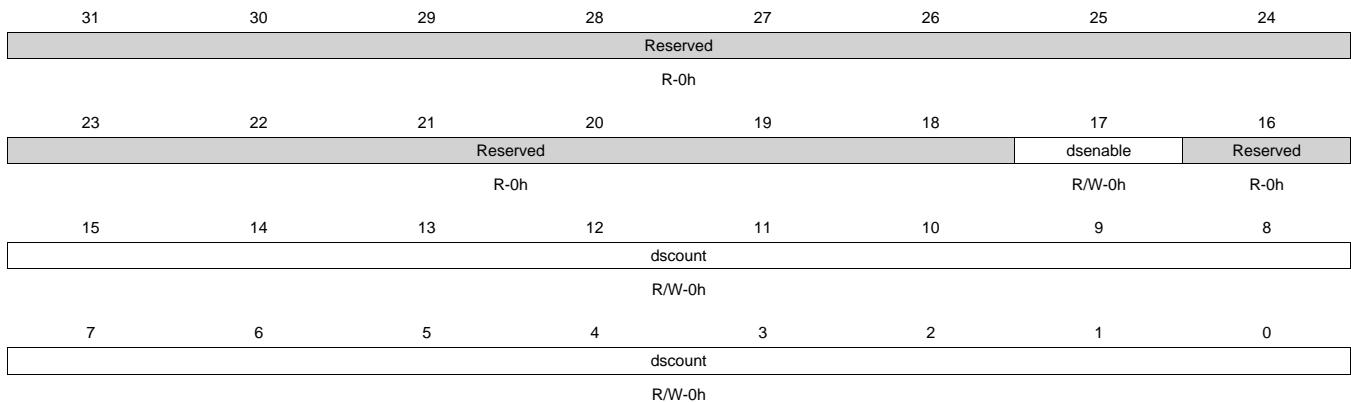
**Table 9-24. rcosc\_ctrl Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-1 | Reserved | R    | 0h    |   |
| 0    | stoposc  | R/W  | 0h    | 0: Oscillator is active<br>1: Oscillator is disabled (power down). Oscillator core output is kept high. |

### 9.3.15 deepsleep\_ctrl Register (offset = 470h) [reset = 0h]

deepsleep\_ctrl is shown in Figure 9-18 and described in Table 9-25.

**Figure 9-18. deepsleep\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-25. deepsleep\_ctrl Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-18 | Reserved | R    | 0h    |   |
| 17    | dsenable | R/W  | 0h    | Deep sleep enable<br>0: Normal operation<br>1: Master oscillator output is gated                |
| 16    | Reserved | R    | 0h    |   |
| 15-0  | dscount  | R/W  | 0h    | Programmable count of how many CLK_M_OSC clocks needs to be seen before exiting deep sleep mode |

### 9.3.16 dev\_feature Register (offset = 604h) [reset = 0h]

dev\_feature is shown in [Figure 9-19](#) and described in [Table 9-26](#).

**Figure 9-19. dev\_feature Register**

| 31       | 30 | 29       | 28 | 27       | 26             | 25             | 24 |
|----------|----|----------|----|----------|----------------|----------------|----|
| Reserved |    | sgx      |    | Reserved |                |                |    |
| R-0h     |    | R-0h     |    | R-0h     |                |                |    |
| 23       | 22 | 21       | 20 | 19       | 18             | 17             | 16 |
|          |    | Reserved |    |          | pru_icssfea[1] | pru_icssfea[0] |    |
|          |    | R-0h     |    | R-0h     | R-0h           | R-0h           |    |
| 15       | 14 | 13       | 12 | 11       | 10             | 9              | 8  |
|          |    | Reserved |    |          | Reserved       | Reserved       |    |
|          |    | R-0h     |    | R-0h     | R-0h           | R-0h           |    |
| 7        | 6  | 5        | 4  | 3        | 2              | 1              | 0  |
| dcan     |    | Reserved |    |          | cpsw           | pru_icss       |    |
| R-0h     |    | R-0h     |    | R-0h     | R-0h           | R-0h           |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-26. dev\_feature Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description  |
|-------|----------------|------|-------|--|
| 31-30 | Reserved       | R    | 0h    |  |
| 29    | sgx            | R    | 0h    | 0: 3D graphics module (SGX) is disabled<br>1: SGX enabled<br>Reset value is device-dependent.  |
| 28-24 | Reserved       | R    | 0h    |  |
| 23-18 | Reserved       | R    | 0h    |  |
| 17    | pru_icssfea[1] | R    | 0h    | 0: TX_AUTO_SEQUENCE disabled<br>1: TX_AUTO_SEQUENCE enabled<br>Reset value is device-dependent.  |
| 16    | pru_icssfea[0] | R    | 0h    | 0: EtherCAT functionality disabled, ODD_NIBBLE disabled<br>1: EtherCAT functionality enabled, ODD_NIBBLE enabled<br>Reset value is device-dependent. |
| 15-10 | Reserved       | R    | 0h    |  |
| 9     | Reserved       | R    | 0h    |  |
| 8     | Reserved       | R    | 0h    |  |
| 7     | dcan           | R    | 0h    | 0: DCAN0, DCAN1 IPs are disabled<br>1: DCAN0, DCAN1 IPs are enabled<br>Reset value is device-dependent.  |
| 6-2   | Reserved       | R    | 0h    |  |
| 1     | cpsw           | R    | 0h    | 0: CP Switch IP (Ethernet) is disabled<br>1: CP Switch IP (Ethernet) is enabled<br>Reset value is device-dependent.                                  |
| 0     | pru_icss       | R    | 0h    | 0: PRU-ICSS is disabled<br>1: PRU-ICSS is enabled<br>Reset value is device-dependent.  |

### 9.3.17 init\_priority\_0 Register (offset = 608h) [reset = 0h]

init\_priority\_0 is shown in [Figure 9-20](#) and described in [Table 9-27](#).

**Figure 9-20. init\_priority\_0 Register**

|          |          |    |          |        |          |    |    |
|----------|----------|----|----------|--------|----------|----|----|
| 31       | 30       | 29 | 28       | 27     | 26       | 25 | 24 |
| Reserved |          |    |          | tcwr2  | tcrd2    |    |    |
| R-0h     |          |    |          | R/W-0h | R/W-0h   |    |    |
| 23       | 22       | 21 | 20       | 19     | 18       | 17 | 16 |
| tcwr1    | tcrd1    |    | tcwr0    | tcrd0  |          |    |    |
| R/W-0h   | R/W-0h   |    | R/W-0h   | R/W-0h |          |    |    |
| 15       | 14       | 13 | 12       | 11     | 10       | 9  | 8  |
| p1500    | Reserved |    |          |        | R-0h     |    |    |
| R/W-0h   | R/W-0h   |    |          |        | R-0h     |    |    |
| 7        | 6        | 5  | 4        | 3      | 2        | 1  | 0  |
| mmu      | pru_icss |    | Reserved |        | host_arm |    |    |
| R/W-0h   | R/W-0h   |    | R/W-0h   | R/W-0h |          |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-27. init\_priority\_0 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                          |
|-------|----------|------|-------|--------------------------------------|
| 31-28 | Reserved | R    | 0h    |                                      |
| 27-26 | tcwr2    | R/W  | 0h    | TPTC 2 Write Port initiator priority |
| 25-24 | tcrd2    | R/W  | 0h    | TPTC 2 Read Port initiator priority  |
| 23-22 | tcwr1    | R/W  | 0h    | TPTC 1 Write Port initiator priority |
| 21-20 | tcrd1    | R/W  | 0h    | TPTC 1 Read Port initiator priority  |
| 19-18 | tcwr0    | R/W  | 0h    | TPTC 0 Write Port initiator priority |
| 17-16 | tcrd0    | R/W  | 0h    | TPTC 0 Read Port initiator priority  |
| 15-14 | p1500    | R/W  | 0h    | P1500 Port Initiator priority        |
| 13-8  | Reserved | R    | 0h    |                                      |
| 7-6   | mmu      | R/W  | 0h    | System MMU initiator priority        |
| 5-4   | pru_icss | R/W  | 0h    | PRU-ICSS initiator priority          |
| 3-2   | Reserved | R    | 0h    |                                      |
| 1-0   | host_arm | R/W  | 0h    | Host Cortex A8 initiator priority    |

### 9.3.18 init\_priority\_1 Register (offset = 60Ch) [reset = 0h]

init\_priority\_1 is shown in [Figure 9-21](#) and described in [Table 9-28](#).

**Figure 9-21. init\_priority\_1 Register**

|          |    |         |    |          |    |          |    |
|----------|----|---------|----|----------|----|----------|----|
| 31       | 30 | 29      | 28 | 27       | 26 | 25       | 24 |
| Reserved |    |         |    |          |    | debug    |    |
| R-0h     |    |         |    |          |    | R/W-0h   |    |
| 23       | 22 | 21      | 20 | 19       | 18 | 17       | 16 |
| lcd      |    | sgx     |    | Reserved |    | Reserved |    |
| R/W-0h   |    | R/W-0h  |    | R-0h     |    | R-0h     |    |
| 15       | 14 | 13      | 12 | 11       | 10 | 9        | 8  |
| Reserved |    |         |    |          |    |          |    |
| R-0h     |    |         |    |          |    |          |    |
| 7        | 6  | 5       | 4  | 3        | 2  | 1        | 0  |
| usb_qmgr |    | usb_dma |    | Reserved |    | cpsw     |    |
| R/W-0h   |    | R/W-0h  |    | R-0h     |    | R/W-0h   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

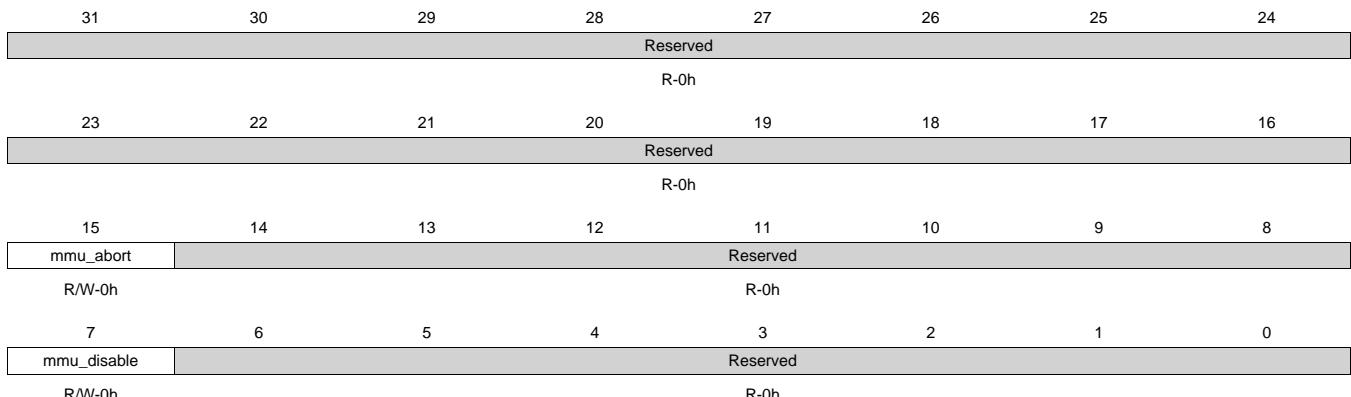
**Table 9-28. init\_priority\_1 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                          |
|-------|----------|------|-------|--------------------------------------|
| 31-26 | Reserved | R    | 0h    |                                      |
| 25-24 | debug    | R/W  | 0h    | Debug Subsystem initiator priority   |
| 23-22 | lcd      | R/W  | 0h    | LCD initiator priority               |
| 21-20 | sgx      | R/W  | 0h    | SGX initiator priority               |
| 19-18 | Reserved | R    | 0h    |                                      |
| 17-16 | Reserved | R    | 0h    |                                      |
| 15-8  | Reserved | R    | 0h    |                                      |
| 7-6   | usb_qmgr | R/W  | 0h    | USB Queue Manager initiator priority |
| 5-4   | usb_dma  | R/W  | 0h    | USB DMA port initiator priority      |
| 3-2   | Reserved | R    | 0h    |                                      |
| 1-0   | cpsw     | R/W  | 0h    | CPSW initiator priority              |

### 9.3.19 mmu\_cfg Register (offset = 610h) [reset = 0h]

mmu\_cfg is shown in [Figure 9-22](#) and described in [Table 9-29](#).

**Figure 9-22. mmu\_cfg Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

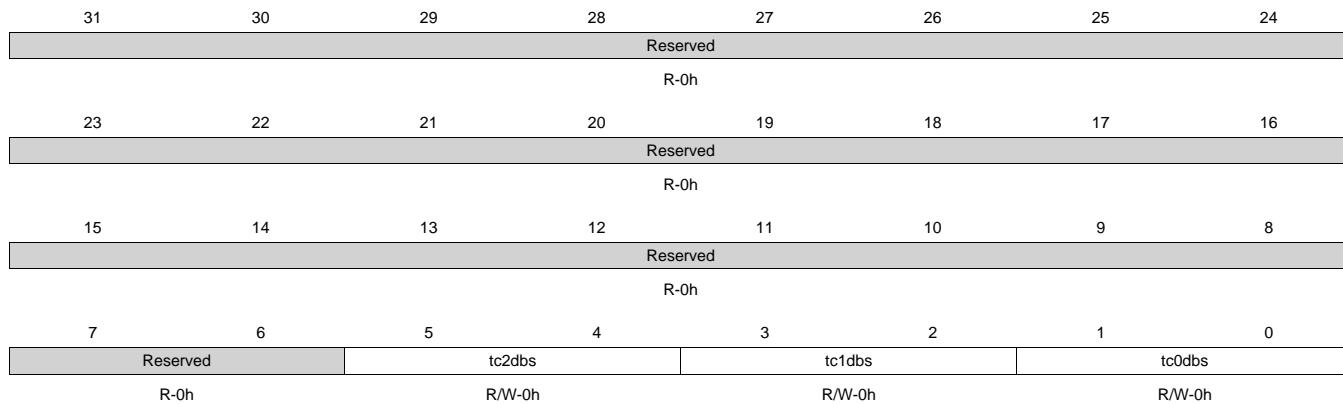
**Table 9-29. mmu\_cfg Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-16 | Reserved    | R    | 0h    |  |
| 15    | mmu_abort   | R/W  | 0h    | MMU abort operation This bit causes the MMU to abort the current operation in case of lockup.  |
| 14-8  | Reserved    | R    | 0h    |  |
| 7     | mmu_disable | R/W  | 0h    | MMU Disable Setting this bit disables MMU table lookup and causes accesses to use the non-translated address.<br>This bit defaults to enabled but an identical bit within an MMU configuration register defaults to disabled and must be set after the page tables are programmed for MMU operation. |
| 6-0   | Reserved    | R    | 0h    |  |

### 9.3.20 tptc\_cfg Register (offset = 614h) [reset = 0h]

tptc\_cfg is shown in [Figure 9-23](#) and described in [Table 9-30](#).

**Figure 9-23. tptc\_cfg Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-30. tptc\_cfg Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-6 | Reserved | R    | 0h    |   |
| 5-4  | tc2dbs   | R/W  | 0h    | TPTC2 Default Burst Size<br>00: 16 byte<br>01: 32 byte<br>10: 64 byte<br>11: 128 byte |
| 3-2  | tc1dbs   | R/W  | 0h    | TPTC1 Default Burst Size<br>00: 16 byte<br>01: 32 byte<br>10: 64 byte<br>11: 128 byte |
| 1-0  | tc0dbs   | R/W  | 0h    | TPTC0 Default Burst Size<br>00: 16 byte<br>01: 32 byte<br>10: 64 byte<br>11: 128 byte |

### 9.3.21 usb\_ctrl0 Register (offset = 620h) [reset = 0h]

usb\_ctrl0 is shown in [Figure 9-24](#) and described in [Table 9-31](#).

**Figure 9-24. usb\_ctrl0 Register**

| 31               | 30             | 29           | 28          | 27           | 26          | 25         | 24       |
|------------------|----------------|--------------|-------------|--------------|-------------|------------|----------|
| Reserved         |                |              |             |              |             |            |          |
| R/W-3Ch          |                |              |             |              |             |            |          |
| 23               | 22             | 21           | 20          | 19           | 18          | 17         | 16       |
| datapolarity_inv | Reserved       | Reserved     | otgessenden | otgvdet_en   | dmgpio_pd   | dppgpio_pd | Reserved |
| R/W-0h           | R-0h           | R-0h         | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |
| 15               | 14             | 13           | 12          | 11           | 10          | 9          | 8        |
| Reserved         | gpio_sig_cross | gpio_sig_inv | gpiomode    | Reserved     | cdet_extctl | dppullup   | dmpullup |
| R/W-0h           | R/W-0h         | R/W-0h       | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |
| 7                | 6              | 5            | 4           | 3            | 2           | 1          | 0        |
| chgvsrc_en       | chgisink_en    | sinkondp     | srcondm     | chgdet_rstrt | chgdet_dis  | otg_pwrnd  | cm_pwrnd |
| R/W-0h           | R/W-0h         | R/W-0h       | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-31. usb\_ctrl0 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | Reserved         | R/W  | 3Ch   | Reserved. Any writes to this register must keep these bits set to 0x3C.   |
| 23    | datapolarity_inv | R/W  | 0h    | Data Polarity Invert:<br>0: DP/DM (normal polarity matching port definition)<br>1: DM/DP (inverted polarity of port definition) |
| 22    | Reserved         | R    | 0h    |   |
| 21    | Reserved         | R    | 0h    |   |
| 20    | otgessenden      | R/W  | 0h    | Session End Detect Enable<br>0: Disable Session End Comparator<br>1: Turns on Session End Comparator                            |
| 19    | otgvdet_en       | R/W  | 0h    | VBUS Detect Enable<br>0: Disable VBUS Detect Enable<br>1: Turns on all comparators except Session End comparator                |
| 18    | dmgpio_pd        | R/W  | 0h    | Pulldown on DM in GPIO Mode<br>0: Enables pulldown<br>1: Disables pulldown  |
| 17    | dppgpio_pd       | R/W  | 0h    | Pulldown on DP in GPIO Mode<br>0: Enables pulldown<br>1: Disables pulldown  |
| 16    | Reserved         | R/W  | 0h    |   |
| 15    | Reserved         | R/W  | 0h    |   |
| 14    | gpio_sig_cross   | R/W  | 0h    | UART TX -> DM<br>UART RX -> DP  |
| 13    | gpio_sig_inv     | R/W  | 0h    | UART TX -> Invert -> DP<br>UART RX -> Invert -> DM  |
| 12    | gpiomode         | R/W  | 0h    | GPIO Mode<br>0: USB Mode<br>1: GPIO Mode (UART Mode)  |
| 11    | Reserved         | R/W  | 0h    |   |
| 10    | cdet_extctl      | R/W  | 0h    | Bypass the charger detection state machine<br>0: Charger detection on<br>1: Charger detection is bypassed                       |
| 9     | dppullup         | R/W  | 0h    | Pullup on DP line<br>0: No effect<br>1: Enable pullup on DP line  |

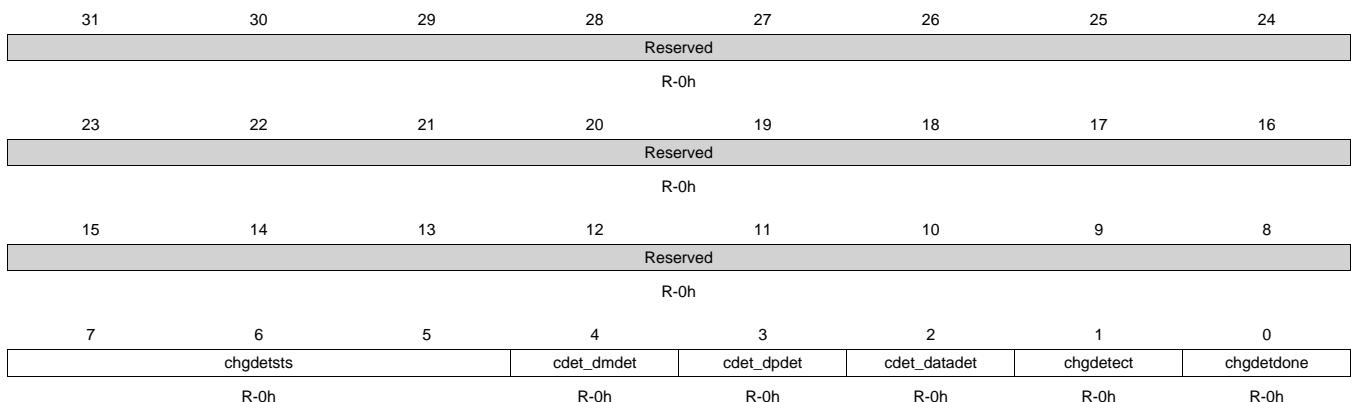
**Table 9-31. usb\_ctrl0 Register Field Descriptions (continued)**

| Bit | Field        | Type | Reset | Description  |
|-----|--------------|------|-------|--|
| 8   | dmpullup     | R/W  | 0h    | Pullup on DM line<br>0: No effect<br>1: Enable pullup on DM line           |
| 7   | chgvsr_en    | R/W  | 0h    | Enable VSRC on DP line (Host Charger case)                                 |
| 6   | chgisink_en  | R/W  | 0h    | Enable ISINK on DM line (Host Charger case)                                |
| 5   | sinkondp     | R/W  | 0h    | Sink on DP<br>0: Sink on DM<br>1: Sink on DP                               |
| 4   | srcondm      | R/W  | 0h    | Source on DM<br>0: Source on DP<br>1: Source on DM                         |
| 3   | chgdet_rstrt | R/W  | 0h    | Restart Charger Detect   |
| 2   | chgdet_dis   | R/W  | 0h    | Charger Detect Disable<br>0: Enable<br>1: Disable                          |
| 1   | otg_pwrndn   | R/W  | 0h    | Power down the USB OTG PHY<br>0: PHY in normal mode<br>1: PHY Powered down |
| 0   | cm_pwrndn    | R/W  | 0h    | Power down the USB CM PHY<br>0: PHY in normal mode<br>1: PHY Powered down  |

### 9.3.22 usb\_sts0 Register (offset = 624h) [reset = 0h]

usb\_sts0 is shown in [Figure 9-25](#) and described in [Table 9-32](#).

**Figure 9-25. usb\_sts0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-32. usb\_sts0 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-8 | Reserved     | R    | 0h    |  |
| 7-5  | chgdets      | R    | 0h    | Charge Detection Status<br>000: Wait State (When a D+WPU and D-15K are connected, it enters into this state and will remain in this state unless it enters into other state)<br>001: No Contact<br>010: PS/2<br>011: Unknown error<br>100: Dedicated charger(valid if CE is HIGH)<br>101: HOST charger (valid if CE is HIGH)<br>110: PC<br>111: Interrupt (if any of the pullup is enabled, charger detect routine gets interrupted and will restart from the beginning if the same is disabled) |
| 4    | cdet_dmdet   | R    | 0h    | DM Comparator Output   |
| 3    | cdet_dpdet   | R    | 0h    | DP Comparator Output   |
| 2    | cdet_datadet | R    | 0h    | Charger Comparator Output  |
| 1    | chgdetect    | R    | 0h    | Charger Detection Status<br>0: Charger was no detected<br>1: Charger was detected  |
| 0    | chgdetdone   | R    | 0h    | Charger Detection Protocol Done  |

### 9.3.23 usb\_ctrl1 Register (offset = 628h) [reset = 0h]

usb\_ctrl1 is shown in [Figure 9-26](#) and described in [Table 9-33](#).

**Figure 9-26. usb\_ctrl1 Register**

| 31               | 30             | 29           | 28          | 27           | 26          | 25         | 24       |
|------------------|----------------|--------------|-------------|--------------|-------------|------------|----------|
| Reserved         |                |              |             |              |             |            |          |
| R/W-3Ch          |                |              |             |              |             |            |          |
| 23               | 22             | 21           | 20          | 19           | 18          | 17         | 16       |
| datapolarity_inv | Reserved       | Reserved     | otgessenden | otgvdet_en   | dmgpio_pd   | dppgpio_pd | Reserved |
| R/W-0h           | R-0h           | R-0h         | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |
| 15               | 14             | 13           | 12          | 11           | 10          | 9          | 8        |
| Reserved         | gpio_sig_cross | gpio_sig_inv | gpiomode    | Reserved     | cdet_extctl | dppullup   | dmpullup |
| R/W-0h           | R/W-0h         | R/W-0h       | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |
| 7                | 6              | 5            | 4           | 3            | 2           | 1          | 0        |
| chgvsr_en        | chgisink_en    | sinkondp     | srcondm     | chgdet_rstrt | chgdet_dis  | otg_pwrnd  | cm_pwrnd |
| R/W-0h           | R/W-0h         | R/W-0h       | R/W-0h      | R/W-0h       | R/W-0h      | R/W-0h     | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-33. usb\_ctrl1 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | Reserved         | R/W  | 3Ch   | Reserved. Any writes to this register must keep these bits set to 0x3C.   |
| 23    | datapolarity_inv | R/W  | 0h    | Data Polarity Invert:<br>0: DP/DM (normal polarity matching port definition)<br>1: DM/DP (inverted polarity of port definition) |
| 22    | Reserved         | R    | 0h    |   |
| 21    | Reserved         | R    | 0h    |   |
| 20    | otgessenden      | R/W  | 0h    | Session End Detect Enable<br>0: Disable Session End Comparator<br>1: Turns on Session End Comparator                            |
| 19    | otgvdet_en       | R/W  | 0h    | VBUS Detect Enable<br>0: Disable VBUS Detect Enable<br>1: Turns on all comparators except Session End comparator                |
| 18    | dmgpio_pd        | R/W  | 0h    | Pulldown on DM in GPIO Mode<br>0: Enables pulldown<br>1: Disables pulldown  |
| 17    | dppgpio_pd       | R/W  | 0h    | Pulldown on DP in GPIO Mode<br>0: Enables pulldown<br>1: Disables pulldown  |
| 16    | Reserved         | R/W  | 0h    |   |
| 15    | Reserved         | R/W  | 0h    |   |
| 14    | gpio_sig_cross   | R/W  | 0h    | UART TX -> DM<br>UART RX -> DP  |
| 13    | gpio_sig_inv     | R/W  | 0h    | UART TX -> INV -> DP<br>UART RX -> INV -> DM  |
| 12    | gpiomode         | R/W  | 0h    | GPIO Mode<br>0: USB Mode<br>1: GPIO Mode (UART)   |
| 11    | Reserved         | R/W  | 0h    |   |
| 10    | cdet_extctl      | R/W  | 0h    | Bypass the charger detection state machine<br>0: Charger detection on<br>1: Charger detection is bypassed                       |
| 9     | dppullup         | R/W  | 0h    | Pullup on DP line<br>0: No effect<br>1: Enable pullup on DP line  |

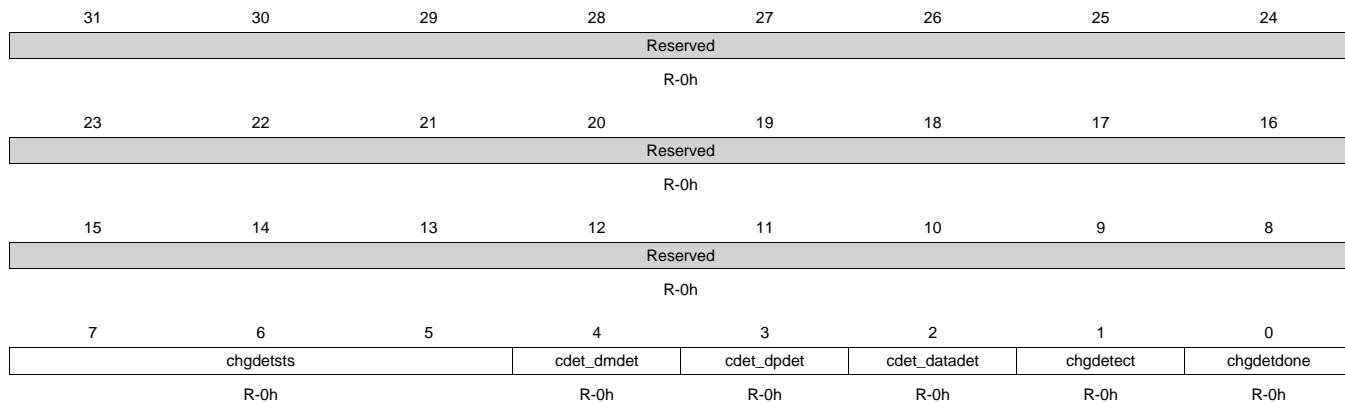
**Table 9-33. usb\_ctrl1 Register Field Descriptions (continued)**

| Bit | Field        | Type | Reset | Description  |
|-----|--------------|------|-------|--|
| 8   | dmpullup     | R/W  | 0h    | Pullup on DM line<br>0: No effect<br>1: Enable pullup on DM line           |
| 7   | chgvsr_en    | R/W  | 0h    | Enable VSRC on DP line (Host Charger case)                                 |
| 6   | chgisink_en  | R/W  | 0h    | Enable ISINK on DM line (Host Charger case)                                |
| 5   | sinkondp     | R/W  | 0h    | Sink on DP<br>0: Sink on DM<br>1: Sink on DP                               |
| 4   | srcondm      | R/W  | 0h    | Source on DM<br>0: Source on DP<br>1: Source on DM                         |
| 3   | chgdet_rstrt | R/W  | 0h    | Restart Charger Detect   |
| 2   | chgdet_dis   | R/W  | 0h    | Charger Detect Disable<br>0: Enable<br>1: Disable                          |
| 1   | otg_pwrdown  | R/W  | 0h    | Power down the USB OTG PHY<br>0: PHY in normal mode<br>1: PHY Powered down |
| 0   | cm_pwrdown   | R/W  | 0h    | Power down the USB CM PHY<br>1: PHY Powered down<br>0: PHY in normal mode  |

### **9.3.24 usb\_sts1 Register (offset = 62Ch) [reset = 0h]**

usb\_sts1 is shown in [Figure 9-27](#) and described in [Table 9-34](#).

**Figure 9-27. usb\_sts1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

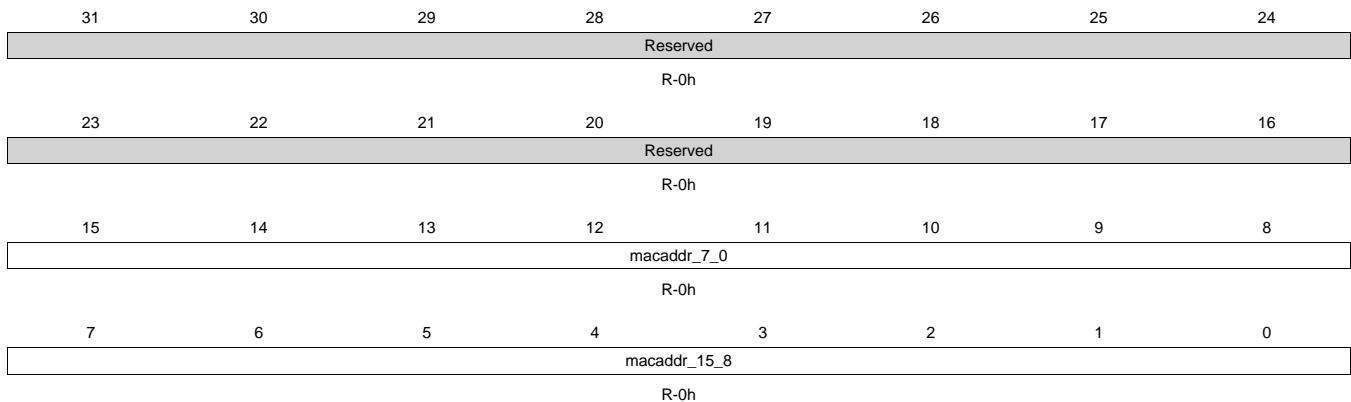
**Table 9-34. usb\_sts1 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-8 | Reserved     | R    | 0h    |  |
| 7-5  | chgdets      | R    | 0h    | Charge Detection Status<br>000: Wait State (When a D+WPU and D-15K are connected, it enters into this state and will remain in this state unless it enters into other state)<br>001: No Contact<br>010: PS/2<br>011: Unknown error<br>100: Dedicated charger(valid if CE is HIGH)<br>101: HOST charger (valid if CE is HIGH)<br>110: PC<br>111: Interrupt (if any of the pullup is enabled, charger detect routine gets interrupted and will restart from the beginning if the same is disabled) |
| 4    | cdet_dmdet   | R    | 0h    | DM Comparator Output   |
| 3    | cdet_dpdet   | R    | 0h    | DP Comparator Output   |
| 2    | cdet_datadet | R    | 0h    | Charger Comparator Output  |
| 1    | chgdetect    | R    | 0h    | Charger Detection Status<br>0: Charger was no detected<br>1: Charger was detected  |
| 0    | chgdetdone   | R    | 0h    | Charger Detection Protocol Done  |

### 9.3.25 mac\_id0\_lo Register (offset = 630h) [reset = 0h]

mac\_id0\_lo is shown in [Figure 9-28](#) and described in [Table 9-35](#).

**Figure 9-28. mac\_id0\_lo Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

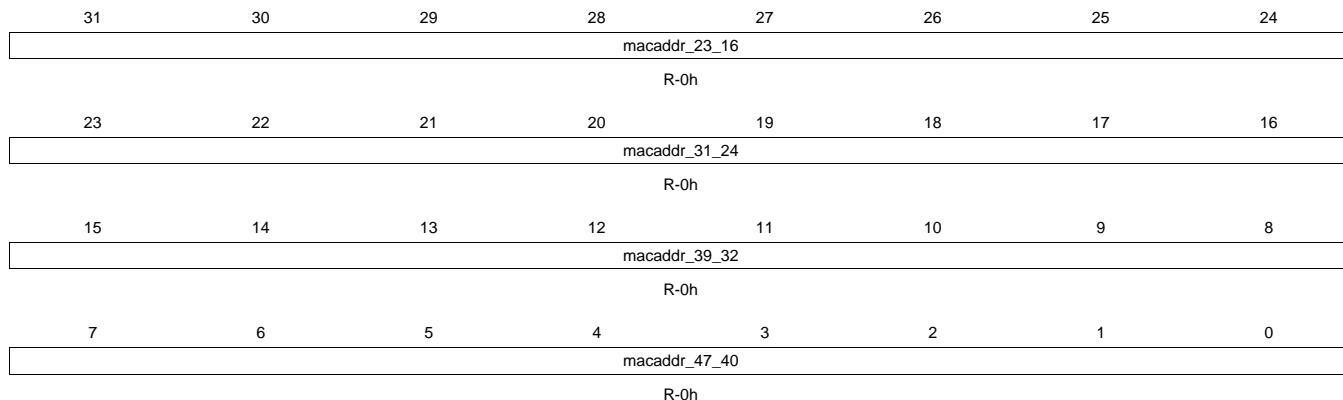
**Table 9-35. mac\_id0\_lo Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31-16 | Reserved     | R    | 0h    |   |
| 15-8  | macaddr_7_0  | R    | 0h    | MAC0 Address - Byte 0<br>Reset value is device-dependent. |
| 7-0   | macaddr_15_8 | R    | 0h    | MAC0 Address - Byte 1<br>Reset value is device-dependent. |

### 9.3.26 mac\_id0\_hi Register (offset = 634h) [reset = 0h]

mac\_id0\_hi is shown in [Figure 9-29](#) and described in [Table 9-36](#).

**Figure 9-29. mac\_id0\_hi Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

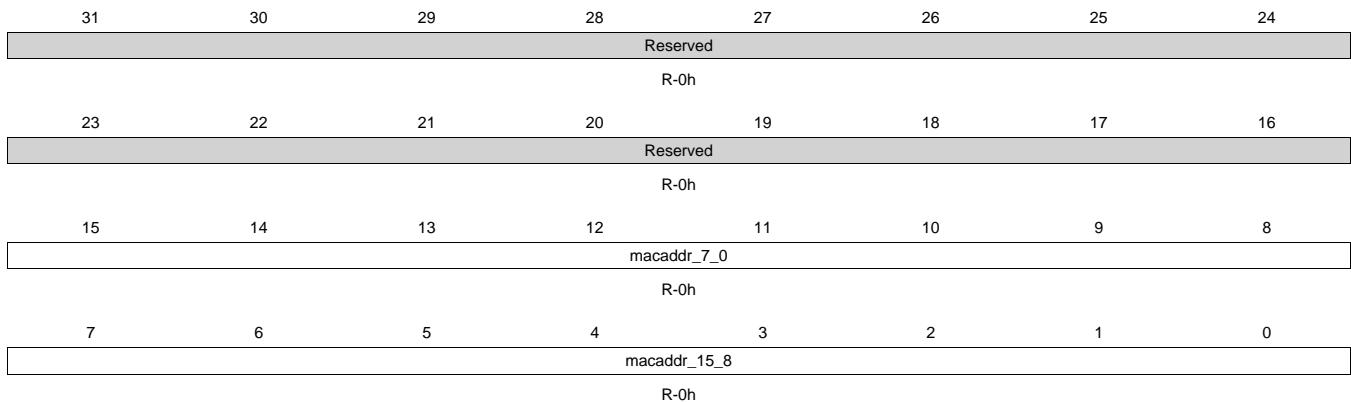
**Table 9-36. mac\_id0\_hi Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-24 | macaddr_23_16 | R    | 0h    | MAC0 Address - Byte 2<br>Reset value is device-dependent. |
| 23-16 | macaddr_31_24 | R    | 0h    | MAC0 Address - Byte 3<br>Reset value is device-dependent. |
| 15-8  | macaddr_39_32 | R    | 0h    | MAC0 Address - Byte 4<br>Reset value is device-dependent. |
| 7-0   | macaddr_47_40 | R    | 0h    | MAC0 Address - Byte 5<br>Reset value is device-dependent. |

### 9.3.27 mac\_id1\_lo Register (offset = 638h) [reset = 0h]

mac\_id1\_lo is shown in [Figure 9-30](#) and described in [Table 9-37](#).

**Figure 9-30. mac\_id1\_lo Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

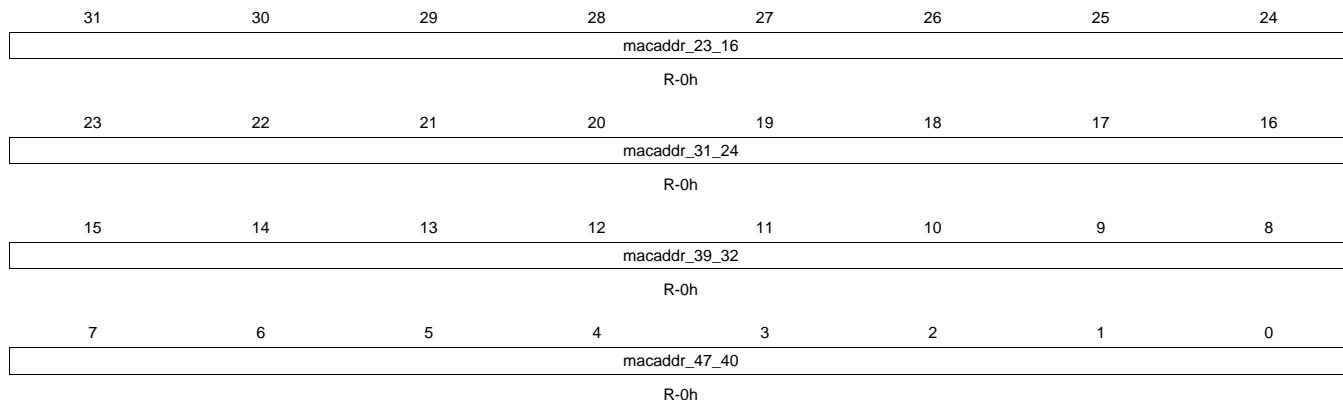
**Table 9-37. mac\_id1\_lo Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31-16 | Reserved     | R    | 0h    |   |
| 15-8  | macaddr_7_0  | R    | 0h    | MAC0 Address - Byte 0<br>Reset value is device-dependent. |
| 7-0   | macaddr_15_8 | R    | 0h    | MAC0 Address - Byte 1<br>Reset value is device-dependent. |

### **9.3.28 mac\_id1\_hi Register (offset = 63Ch) [reset = 0h]**

mac\_id1\_hi is shown in [Figure 9-31](#) and described in [Table 9-38](#).

**Figure 9-31. mac\_id1\_hi Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

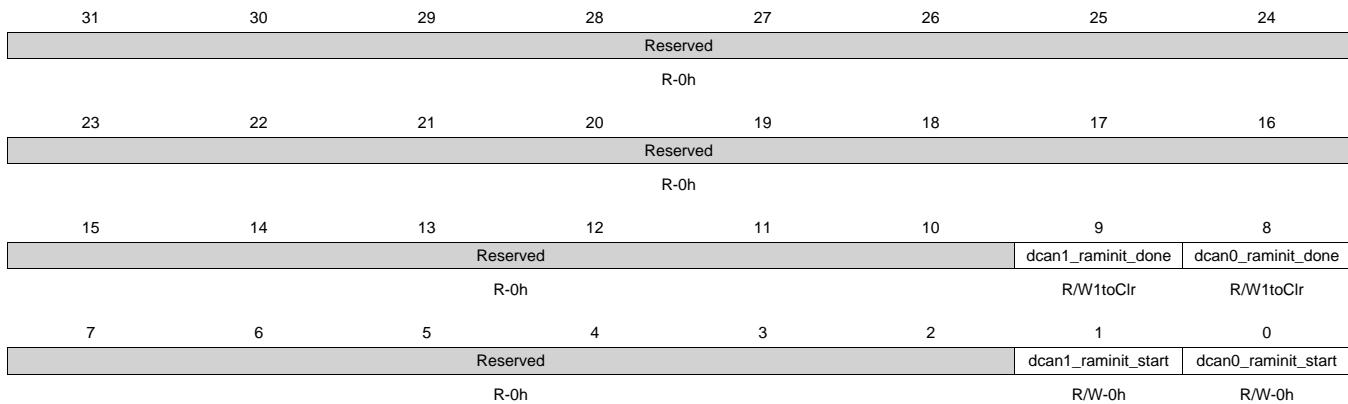
**Table 9-38. mac\_id1\_hi Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-24 | macaddr_23_16 | R    | 0h    | MAC0 Address - Byte 2<br>Reset value is device-dependent. |
| 23-16 | macaddr_31_24 | R    | 0h    | MAC0 Address - Byte 3<br>Reset value is device-dependent. |
| 15-8  | macaddr_39_32 | R    | 0h    | MAC0 Address - Byte 4<br>Reset value is device-dependent. |
| 7-0   | macaddr_47_40 | R    | 0h    | MAC0 Address - Byte 5<br>Reset value is device-dependent. |

### 9.3.29 dcan\_raminit Register (offset = 644h) [reset = 0h]

dcan\_raminit is shown in [Figure 9-32](#) and described in [Table 9-39](#).

**Figure 9-32. dcan\_raminit Register**



LEGEND: R/W = Read/Write; R = Read only; W1toClr = Write 1 to clear bit; -n = value after reset

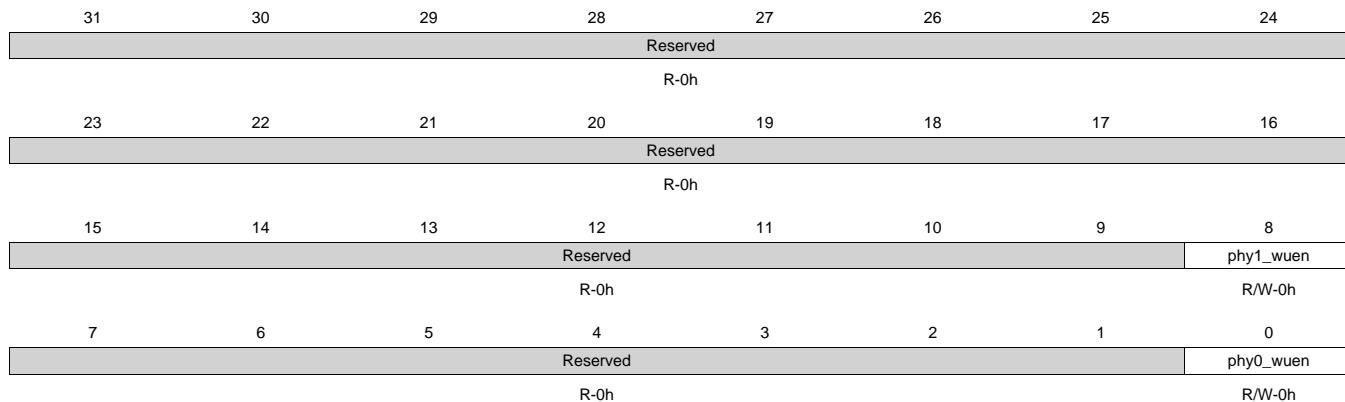
**Table 9-39. dcan\_raminit Register Field Descriptions**

| Bit   | Field               | Type      | Reset | Description  |
|-------|---------------------|-----------|-------|--|
| 31-10 | Reserved            | R         | 0h    |  |
| 9     | dcan1_raminit_done  | R/W1toClr | 0h    | 0: DCAN1 RAM Initialization NOT complete<br>1: DCAN1 RAM Initialization complete |
| 8     | dcan0_raminit_done  | R/W1toClr | 0h    | 0: DCAN0 RAM Initialization NOT complete<br>1: DCAN0 RAM Initialization complete |
| 7-2   | Reserved            | R         | 0h    |  |
| 1     | dcan1_raminit_start | R/W       | 0h    | A transition from 0 to 1 will start DCAN1 RAM initialization sequence.           |
| 0     | dcan0_raminit_start | R/W       | 0h    | A transition from 0 to 1 will start DCAN0 RAM initialization sequence.           |

### **9.3.30 usb\_wkup\_ctrl Register (offset = 648h) [reset = 0h]**

usb\_wkup\_ctrl is shown in [Figure 9-33](#) and described in [Table 9-40](#).

**Figure 9-33. usb\_wkup\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

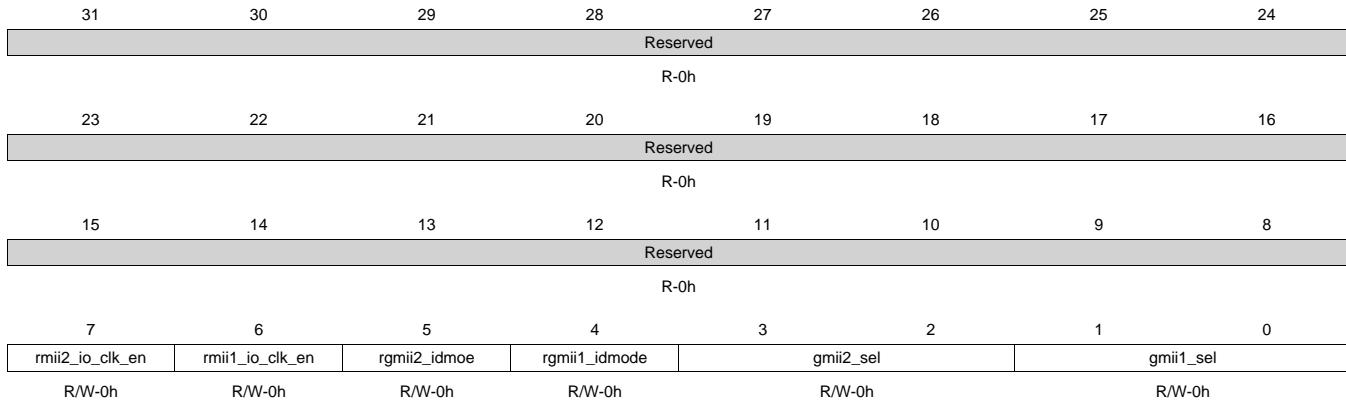
**Table 9-40. usb\_wkup\_ctrl Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-9 | Reserved  | R    | 0h    |  |
| 8    | phy1_wuen | R/W  | 0h    | PHY1 Wakeup Enable.<br>Write to 1 enables WKUP from USB PHY1 |
| 7-1  | Reserved  | R    | 0h    |  |
| 0    | phy0_wuen | R/W  | 0h    | PHY0 Wakeup Enable.<br>Write to 1 enables WKUP from USB PHY0 |

### 9.3.31 gmii\_sel Register (offset = 650h) [reset = 0h]

gmii\_sel is shown in [Figure 9-34](#) and described in [Table 9-41](#).

**Figure 9-34. gmii\_sel Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

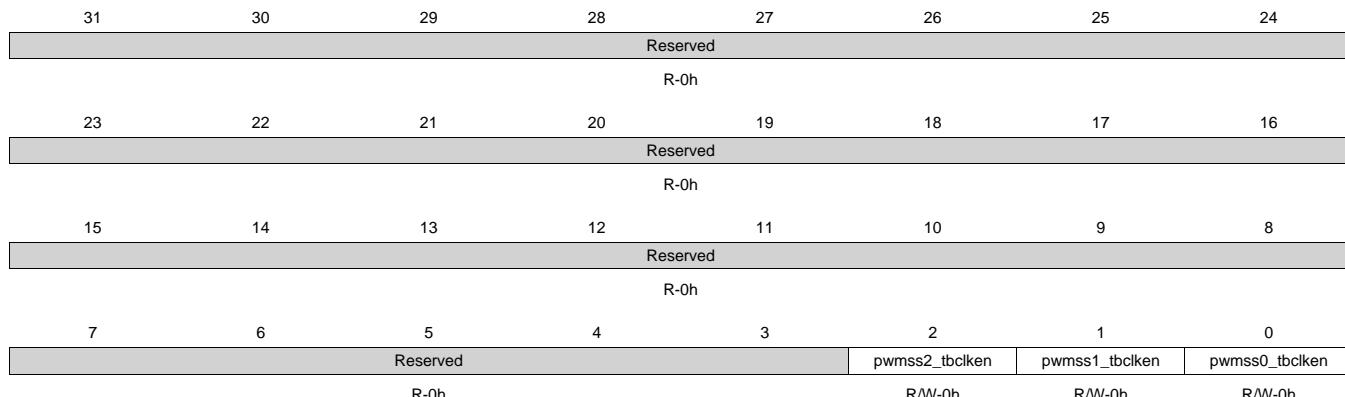
**Table 9-41. gmii\_sel Register Field Descriptions**

| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-8 | Reserved        | R    | 0h    |   |
| 7    | rmii2_io_clk_en | R/W  | 0h    | 1: Enable RMII clock to be sourced from chip pin<br>0: Enable RMII clock to be sourced from PLL |
| 6    | rmii1_io_clk_en | R/W  | 0h    | 1: Enable RMII clock to be sourced from chip pin<br>0: Enable RMII clock to be sourced from PLL |
| 5    | rgmii2_idmode   | R/W  | 0h    | RGMII2 Internal Delay Mode<br>0: Reserved<br>1: No Internal Delay                               |
| 4    | rgmii1_idmode   | R/W  | 0h    | RGMII1 Internal Delay Mode<br>0: Reserved<br>1: No Internal Delay                               |
| 3-2  | gmii2_sel       | R/W  | 0h    | 00: Port2 GMII/MII Mode<br>01: Port2 RMII Mode<br>10: Port2 RGMII Mode<br>11: Not Used          |
| 1-0  | gmii1_sel       | R/W  | 0h    | 00: Port1 GMII/MII Mode<br>01: Port1 RMII Mode<br>10: Port1 RGMII Mode<br>11: Not Used          |

### 9.3.32 pwmss\_ctrl Register (offset = 664h) [reset = 0h]

pwmss\_ctrl is shown in [Figure 9-35](#) and described in [Table 9-42](#).

**Figure 9-35. pwmss\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-42. pwmss\_ctrl Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description                      |
|------|----------------|------|-------|----------------------------------|
| 31-3 | Reserved       | R    | 0h    |                                  |
| 2    | pwmss2_tbclken | R/W  | 0h    | Timebase clock enable for PWMSS2 |
| 1    | pwmss1_tbclken | R/W  | 0h    | Timebase clock enable for PWMSS1 |
| 0    | pwmss0_tbclken | R/W  | 0h    | Timebase clock enable for PWMSS0 |

### 9.3.33 mreqprio\_0 Register (offset = 670h) [reset = 0h]

mreqprio\_0 is shown in [Figure 9-36](#) and described in [Table 9-43](#).

**Figure 9-36. mreqprio\_0 Register**

| 31       | 30 | 29        | 28 | 27       | 26 | 25            | 24 |
|----------|----|-----------|----|----------|----|---------------|----|
| Reserved |    | sgx       |    | Reserved |    | usb1          |    |
| R-0h     |    | R/W-0h    |    | R-0h     |    | R/W-0h        |    |
| 23       | 22 | 21        | 20 | 19       | 18 | 17            | 16 |
| Reserved |    | usb0      |    | Reserved |    | cpsw          |    |
| R-0h     |    | R/W-0h    |    | R-0h     |    | R/W-0h        |    |
| 15       | 14 | 13        | 12 | 11       | 10 | 9             | 8  |
| Reserved |    | Reserved  |    | Reserved |    | pru_icss_pru0 |    |
| R-0h     |    | R-0h      |    | R-0h     |    | R/W-0h        |    |
| 7        | 6  | 5         | 4  | 3        | 2  | 1             | 0  |
| Reserved |    | sab_init1 |    | Reserved |    | sab_init0     |    |
| R-0h     |    | R/W-0h    |    | R-0h     |    | R/W-0h        |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

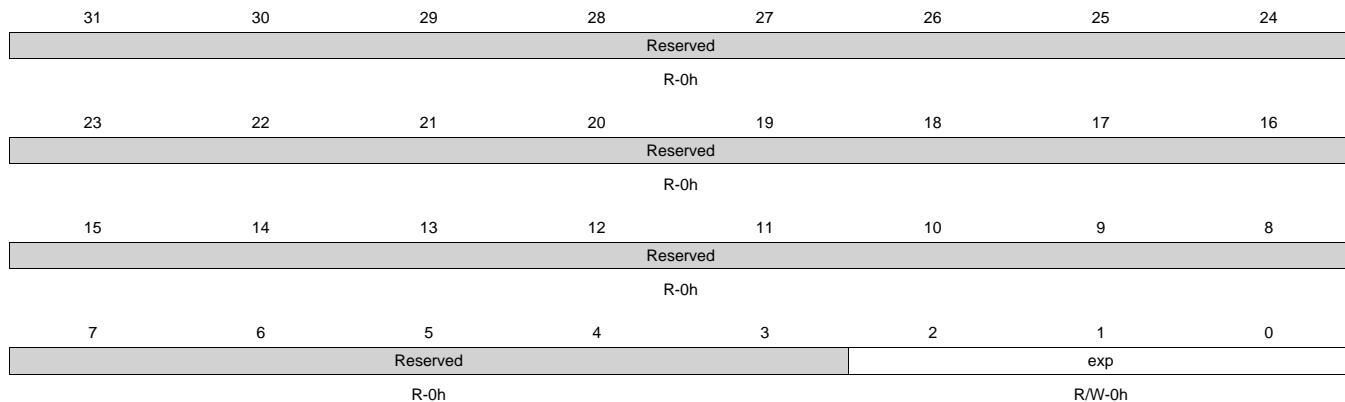
**Table 9-43. mreqprio\_0 Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31    | Reserved      | R    | 0h    |  |
| 30-28 | sgx           | R/W  | 0h    | MReqPriority for SGX Initiator OCP Interface           |
| 27    | Reserved      | R    | 0h    |  |
| 26-24 | usb1          | R/W  | 0h    | MReqPriority for USB1 Initiator OCP Interface          |
| 23    | Reserved      | R    | 0h    |  |
| 22-20 | usb0          | R/W  | 0h    | MReqPriority for USB0 Initiator OCP Interface          |
| 19    | Reserved      | R    | 0h    |  |
| 18-16 | cpsw          | R/W  | 0h    | MReqPriority for CPSW Initiator OCP Interface          |
| 15    | Reserved      | R    | 0h    |  |
| 14-12 | Reserved      | R    | 0h    |  |
| 11    | Reserved      | R    | 0h    |  |
| 10-8  | pru_icss_pru0 | R/W  | 0h    | MReqPriority for PRU-ICSS PRU0 Initiator OCP Interface |
| 7     | Reserved      | R    | 0h    |  |
| 6-4   | sab_init1     | R/W  | 0h    | MReqPriority for MPU Initiator 1 OCP Interface         |
| 3     | Reserved      | R    | 0h    |  |
| 2-0   | sab_init0     | R/W  | 0h    | MReqPriority for MPU Initiator 0 OCP Interface         |

### **9.3.34 mreqprio\_1 Register (offset = 674h) [reset = 0h]**

mreqprio\_1 is shown in [Figure 9-37](#) and described in [Table 9-44](#).

**Figure 9-37. mreqprio\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

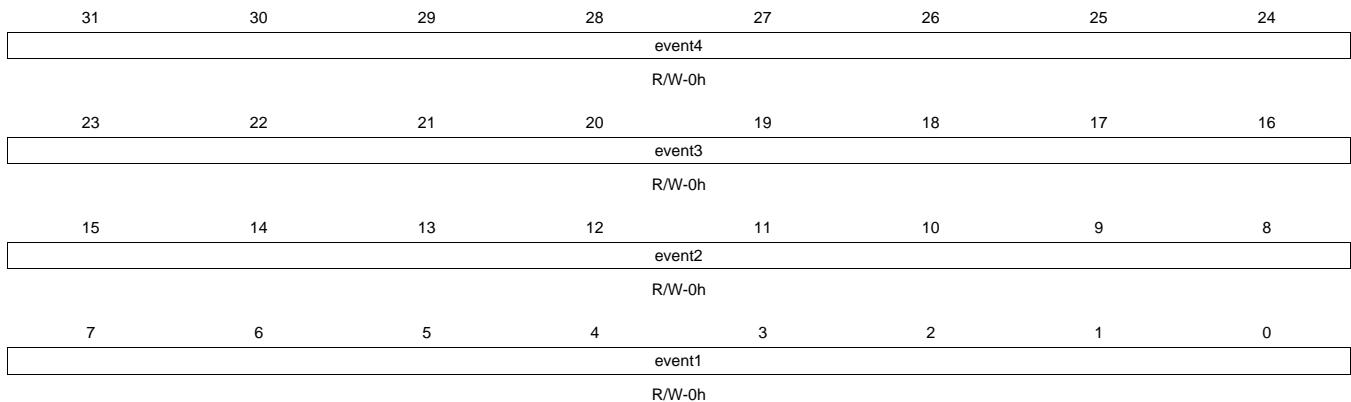
**Table 9-44. mreqprio\_1 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-3 | Reserved | R    | 0h    |  |
| 2-0  | exp      | R/W  | 0h    | MReqPriority for Expansion Initiator OCP Interface |

### 9.3.35 hw\_event\_sel\_grp1 Register (offset = 690h) [reset = 0h]

hw\_event\_sel\_grp1 is shown in [Figure 9-38](#) and described in [Table 9-45](#).

**Figure 9-38. hw\_event\_sel\_grp1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

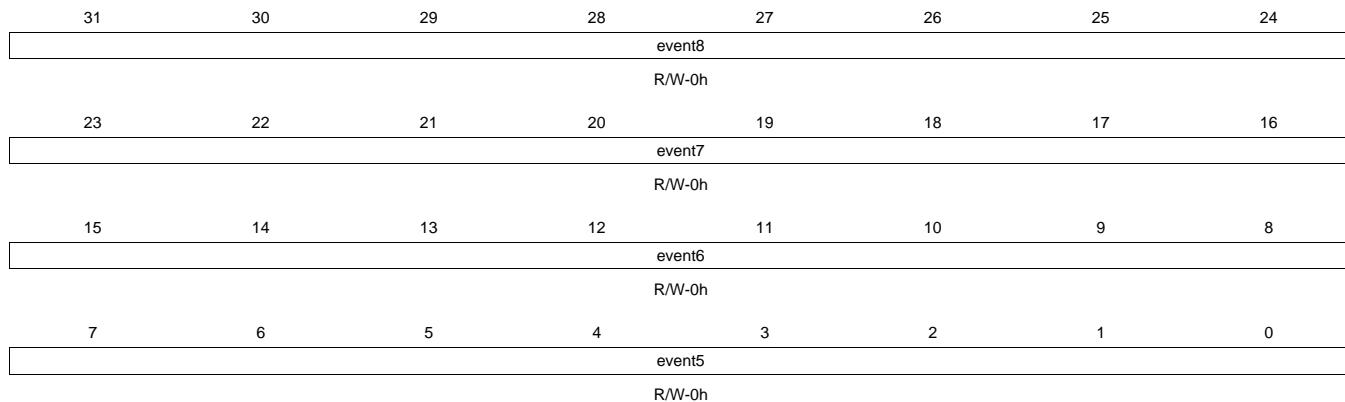
**Table 9-45. hw\_event\_sel\_grp1 Register Field Descriptions**

| Bit   | Field  | Type | Reset | Description                         |
|-------|--------|------|-------|-------------------------------------|
| 31-24 | event4 | R/W  | 0h    | Select 4th trace event from group 1 |
| 23-16 | event3 | R/W  | 0h    | Select 3rd trace event from group 1 |
| 15-8  | event2 | R/W  | 0h    | Select 2nd trace event from group 1 |
| 7-0   | event1 | R/W  | 0h    | Select 1st trace event from group 1 |

### **9.3.36 hw\_event\_sel\_grp2 Register (offset = 694h) [reset = 0h]**

hw\_event\_sel\_grp2 is shown in [Figure 9-39](#) and described in [Table 9-46](#).

**Figure 9-39. hw\_event\_sel\_grp2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

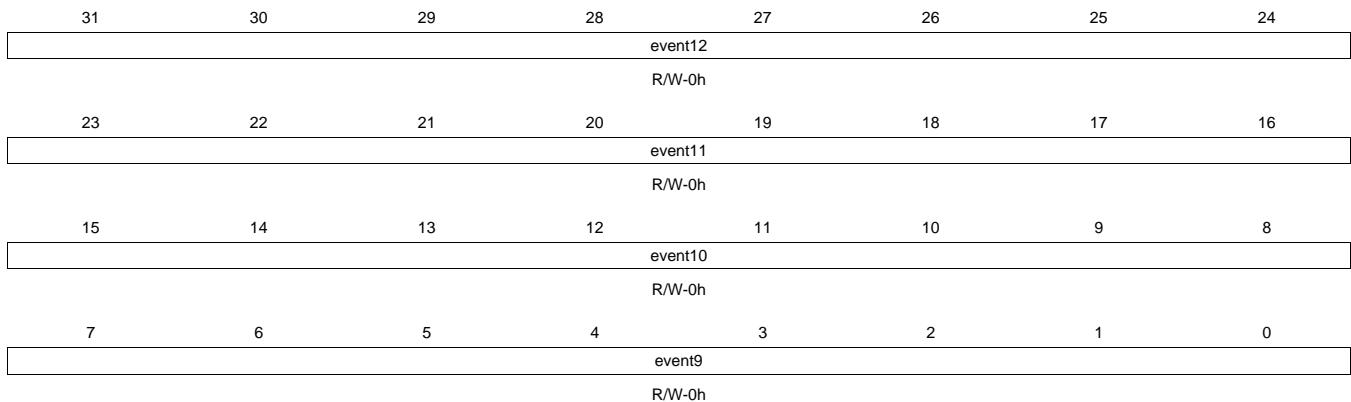
**Table 9-46. hw\_event\_sel\_grp2 Register Field Descriptions**

| Bit   | Field  | Type | Reset | Description                         |
|-------|--------|------|-------|-------------------------------------|
| 31-24 | event8 | R/W  | 0h    | Select 8th trace event from group 2 |
| 23-16 | event7 | R/W  | 0h    | Select 7th trace event from group 2 |
| 15-8  | event6 | R/W  | 0h    | Select 6th trace event from group 2 |
| 7-0   | event5 | R/W  | 0h    | Select 5th trace event from group 2 |

### 9.3.37 hw\_event\_sel\_grp3 Register (offset = 698h) [reset = 0h]

hw\_event\_sel\_grp3 is shown in [Figure 9-40](#) and described in [Table 9-47](#).

**Figure 9-40. hw\_event\_sel\_grp3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

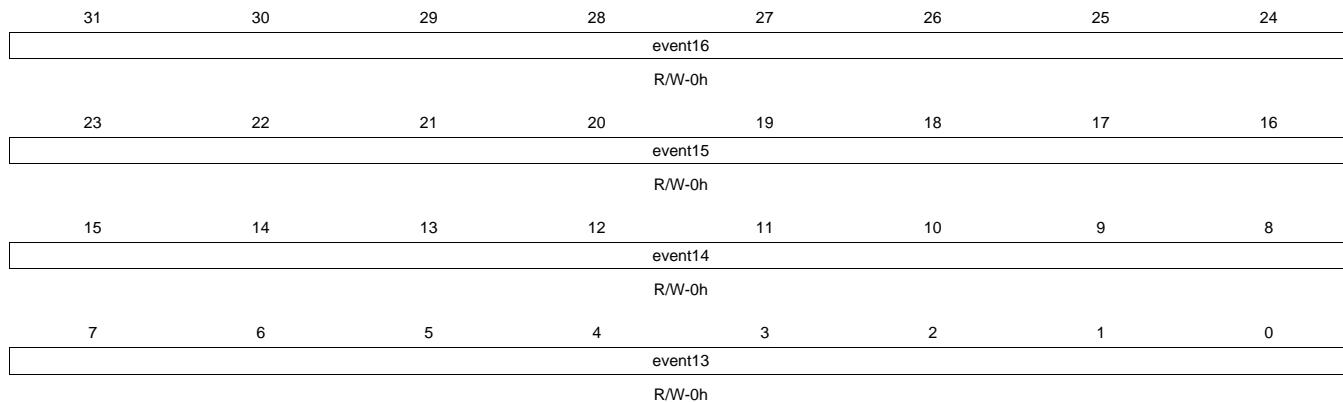
**Table 9-47. hw\_event\_sel\_grp3 Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description                          |
|-------|---------|------|-------|--------------------------------------|
| 31-24 | event12 | R/W  | 0h    | Select 12th trace event from group 3 |
| 23-16 | event11 | R/W  | 0h    | Select 11th trace event from group 3 |
| 15-8  | event10 | R/W  | 0h    | Select 10th trace event from group 3 |
| 7-0   | event9  | R/W  | 0h    | Select 9th trace event from group 3  |

### **9.3.38 hw\_event\_sel\_grp4 Register (offset = 69Ch) [reset = 0h]**

hw\_event\_sel\_grp4 is shown in [Figure 9-41](#) and described in [Table 9-48](#).

**Figure 9-41. hw\_event\_sel\_grp4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

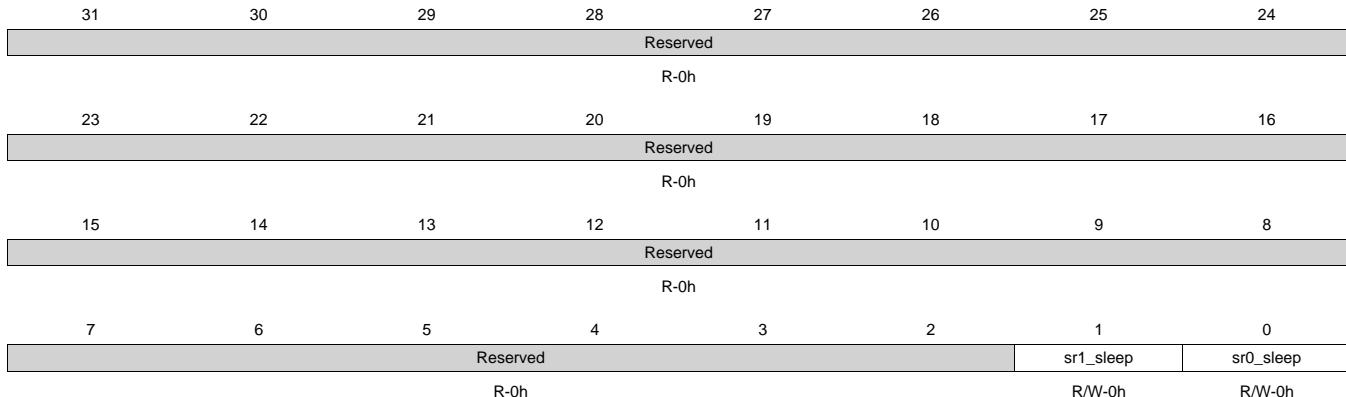
**Table 9-48. hw\_event\_sel\_grp4 Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description                          |
|-------|---------|------|-------|--------------------------------------|
| 31-24 | event16 | R/W  | 0h    | Select 16th trace event from group 4 |
| 23-16 | event15 | R/W  | 0h    | Select 15th trace event from group 4 |
| 15-8  | event14 | R/W  | 0h    | Select 14th trace event from group 4 |
| 7-0   | event13 | R/W  | 0h    | Select 13th trace event from group 4 |

### 9.3.39 smrt\_ctrl Register (offset = 6A0h) [reset = 0h]

smrt\_ctrl is shown in [Figure 9-42](#) and described in [Table 9-49](#).

**Figure 9-42. smrt\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

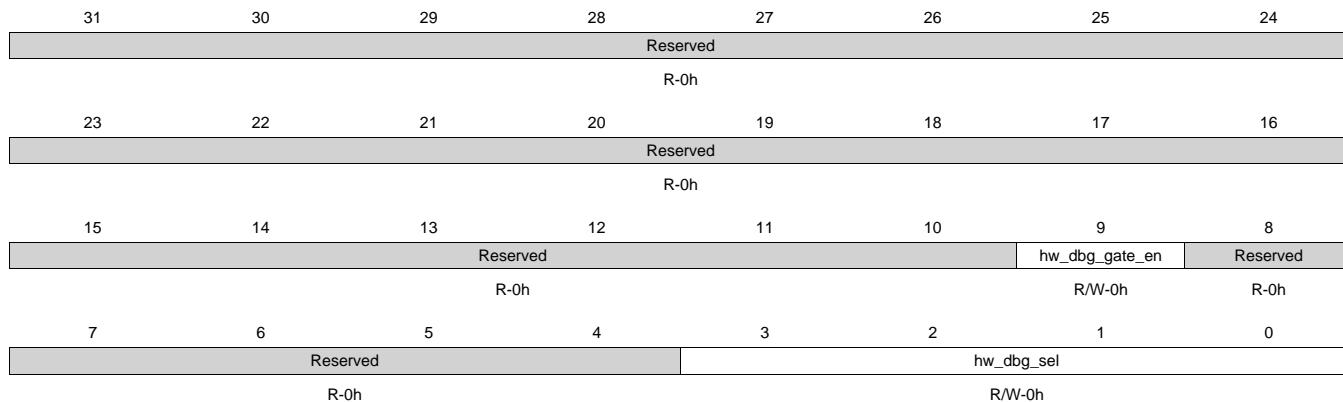
**Table 9-49. smrt\_ctrl Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-2 | Reserved  | R    | 0h    |  |
| 1    | sr1_sleep | R/W  | 0h    | 0: Disable sensor (SRSLEEP on sensor driven to 1)<br>1: Enable sensor (SRSLEEP on sensor driven to 0). |
| 0    | sr0_sleep | R/W  | 0h    | 0: Disable sensor (SRSLEEP on sensor driven to 1)<br>1: Enable sensor (SRSLEEP on sensor driven to 0). |

### 9.3.40 mpuss\_hw\_debug\_sel Register (offset = 6A4h) [reset = 0h]

mpuss\_hw\_debug\_sel is shown in [Figure 9-43](#) and described in [Table 9-50](#).

**Figure 9-43. mpuss\_hw\_debug\_sel Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

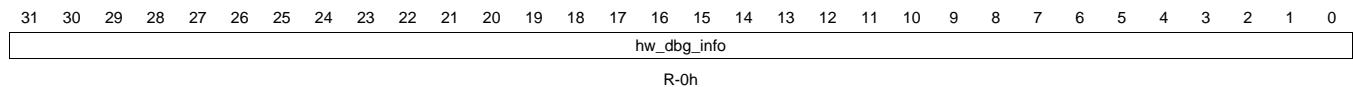
**Table 9-50. mpuss\_hw\_debug\_sel Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description   |
|-------|----------------|------|-------|---|
| 31-10 | Reserved       | R    | 0h    |   |
| 9     | hw_dbg_gate_en | R/W  | 0h    | To save power input to MPUSS_HW_DBG_INFO is gated off to all zeros when HW_DBG_GATE_EN bit is low.<br>0: Debug info gated off<br>1: Debug info not gated off  |
| 8     | Reserved       | R    | 0h    |   |
| 7-4   | Reserved       | R    | 0h    |   |
| 3-0   | hw_dbg_sel     | R/W  | 0h    | Selects which Group of signals are sent out to the MODENA_HW_DBG_INFO register. Please see MPU functional spec for more details<br>0000: Group 0<br>0001: Group 1<br>0010: Group 2<br>0011: Group 3<br>0100: Group 4<br>0101: Group 5<br>0110: Group 6<br>0111: Group 7<br>1xxx: Reserved |

### 9.3.41 mpuss\_hw\_dbg\_info Register (offset = 6A8h) [reset = 0h]

mpuss\_hw\_dbg\_info is shown in [Figure 9-44](#) and described in [Table 9-51](#).

**Figure 9-44. mpuss\_hw\_dbg\_info Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

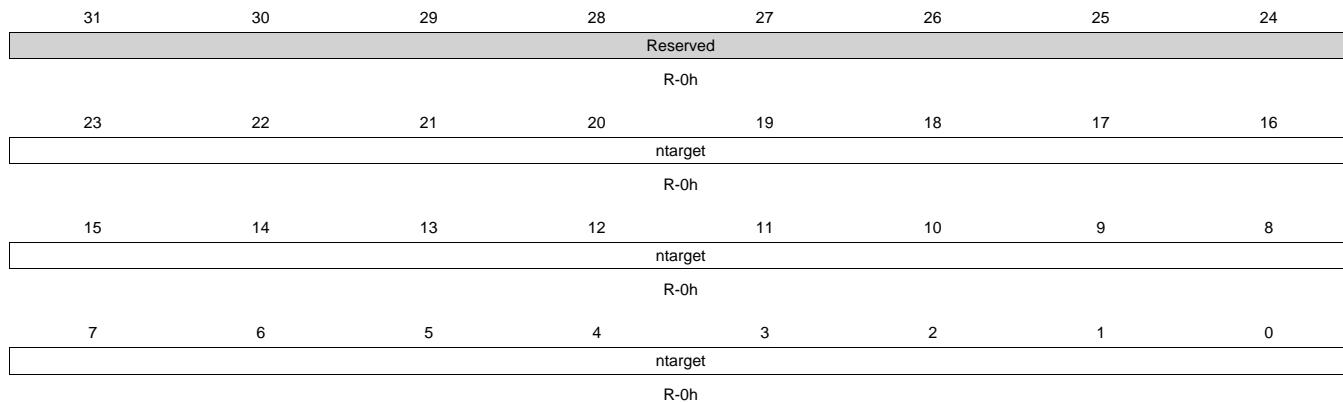
**Table 9-51. mpuss\_hw\_dbg\_info Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description                   |
|------|-------------|------|-------|-------------------------------|
| 31-0 | hw_dbg_info | R    | 0h    | Hardware Debug Info from MPU. |

### **9.3.42 vdd\_mpu\_opp\_050 Register (offset = 770h) [reset = 0h]**

vdd\_mpu\_opp\_050 is shown in [Figure 9-45](#) and described in [Table 9-52](#).

**Figure 9-45. vdd\_mpu\_opp\_050 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

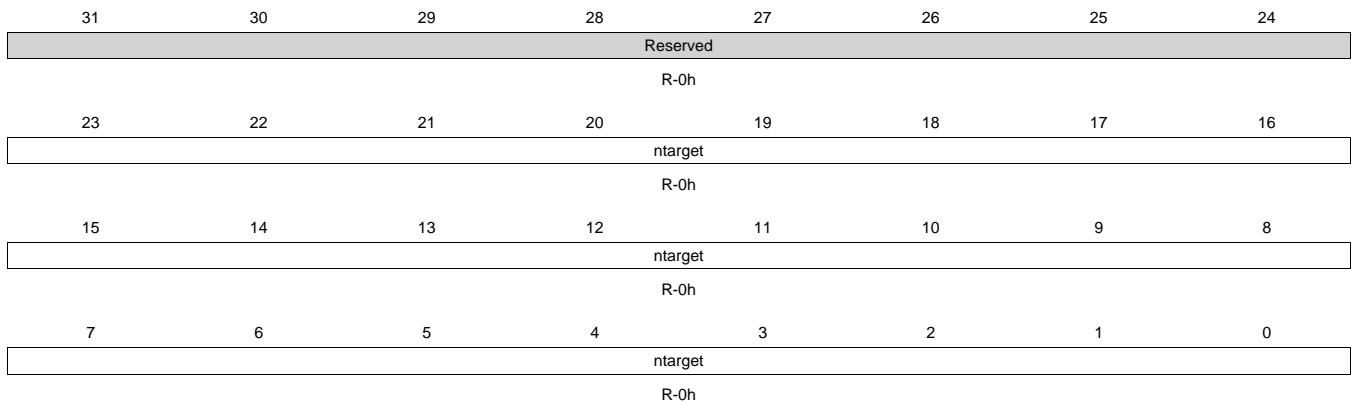
**Table 9-52. vdd\_mpu\_opp\_050 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | Reserved | R    | 0h    |  |
| 23-0  | ntarget  | R    | 0h    | Ntarget value for MPU Voltage domain OPP50<br>Reset value is device-dependent. |

### 9.3.43 vdd\_mpu\_opp\_100 Register (offset = 774h) [reset = 0h]

vdd\_mpu\_opp\_100 is shown in [Figure 9-46](#) and described in [Table 9-53](#).

**Figure 9-46. vdd\_mpu\_opp\_100 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

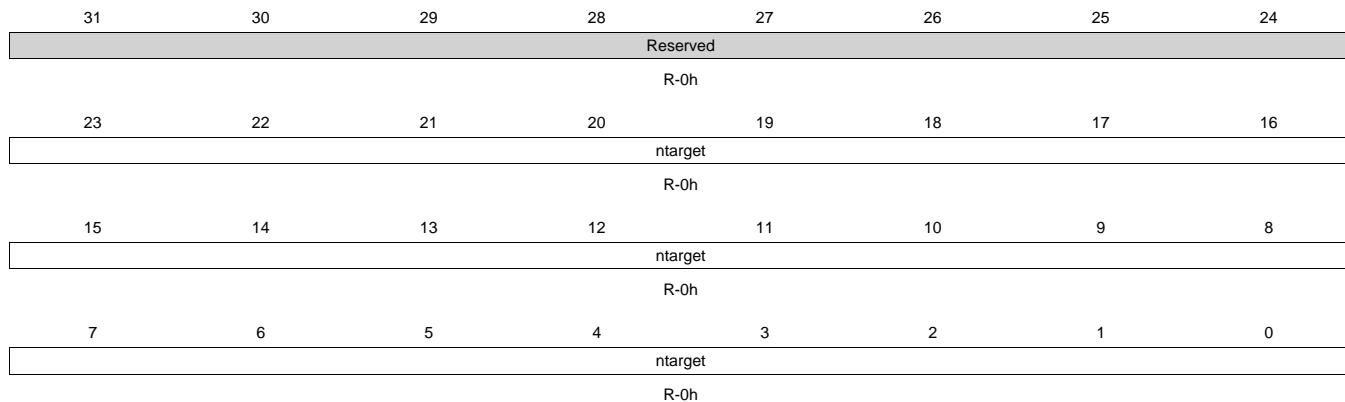
**Table 9-53. vdd\_mpu\_opp\_100 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-24 | Reserved | R    | 0h    |   |
| 23-0  | ntarget  | R    | 0h    | Ntarget value for MPU Voltage domain OPP100<br>Reset value is device-dependent. |

### **9.3.44 vdd\_mpu\_opp\_120 Register (offset = 778h) [reset = 0h]**

vdd\_mpu\_opp\_120 is shown in [Figure 9-47](#) and described in [Table 9-54](#).

**Figure 9-47. vdd\_mpu\_opp\_120 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

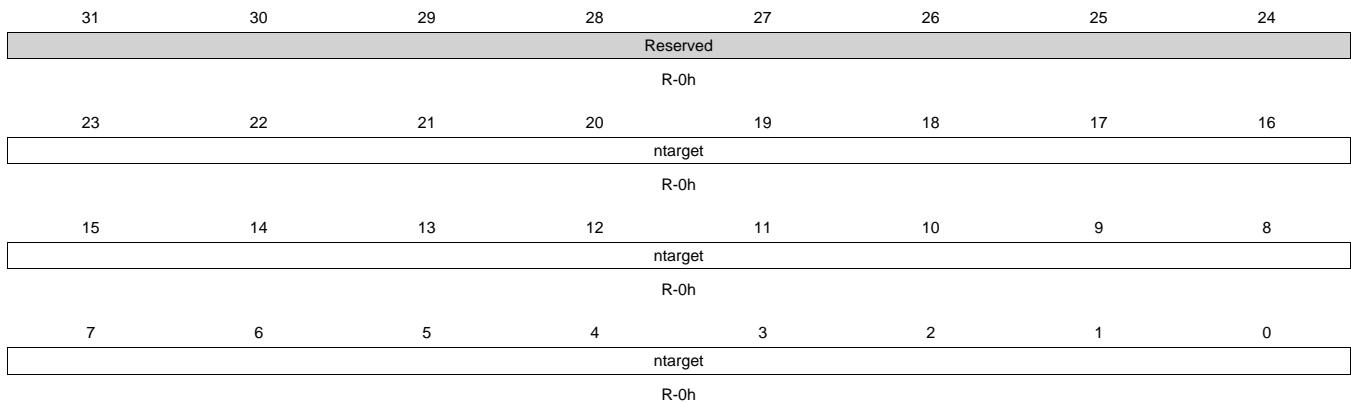
**Table 9-54. vdd\_mpu\_opp\_120 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-24 | Reserved | R    | 0h    |   |
| 23-0  | nttarget | R    | 0h    | Ntarget value for MPU Voltage domain OPP120<br>Reset value is device-dependent. |

### 9.3.45 vdd\_mpu\_opp\_turbo Register (offset = 77Ch) [reset = 0h]

vdd\_mpu\_opp\_turbo is shown in [Figure 9-48](#) and described in [Table 9-55](#).

**Figure 9-48. vdd\_mpu\_opp\_turbo Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

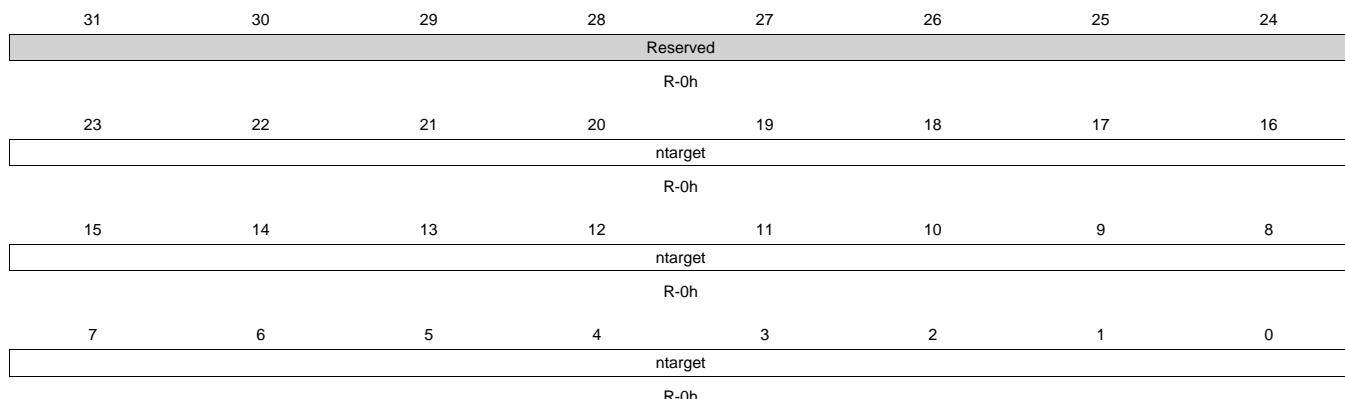
**Table 9-55. vdd\_mpu\_opp\_turbo Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-24 | Reserved | R    | 0h    |   |
| 23-0  | ntarget  | R    | 0h    | Ntarget value for MPU Voltage domain OPPTURBO<br>Reset value is device-dependent. |

### **9.3.46 vdd\_core\_opp\_050 Register (offset = 7B8h) [reset = 0h]**

vdd\_core\_opp\_050 is shown in [Figure 9-49](#) and described in [Table 9-56](#).

**Figure 9-49. vdd\_core\_opp\_050 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

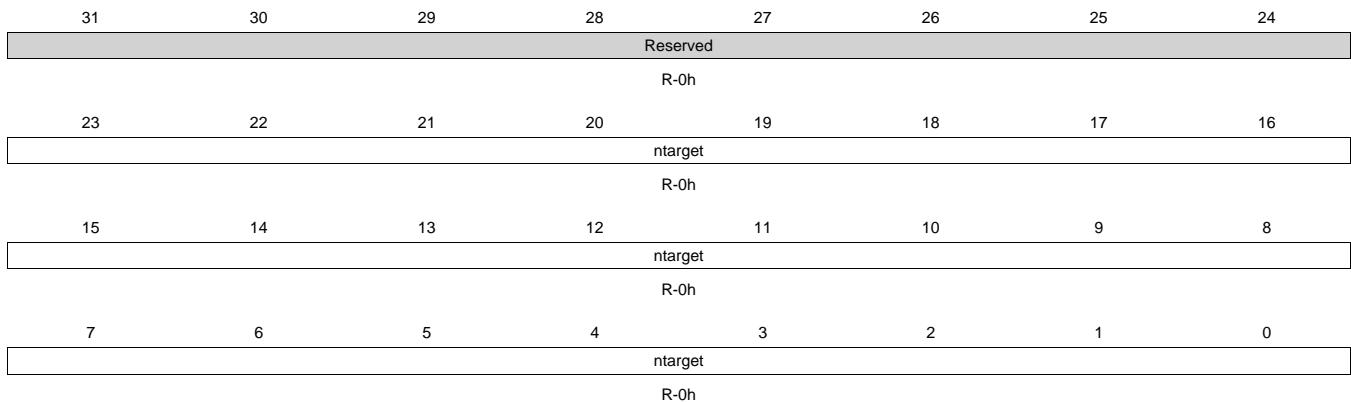
**Table 9-56. vdd\_core\_opp\_050 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-24 | Reserved | R    | 0h    |   |
| 23-0  | nttarget | R    | 0h    | Ntarget value for CORE Voltage domain OPP50<br>Reset value is device-dependent. |

### 9.3.47 vdd\_core\_opp\_100 Register (offset = 7BCh) [reset = 0h]

vdd\_core\_opp\_100 is shown in [Figure 9-50](#) and described in [Table 9-57](#).

**Figure 9-50. vdd\_core\_opp\_100 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

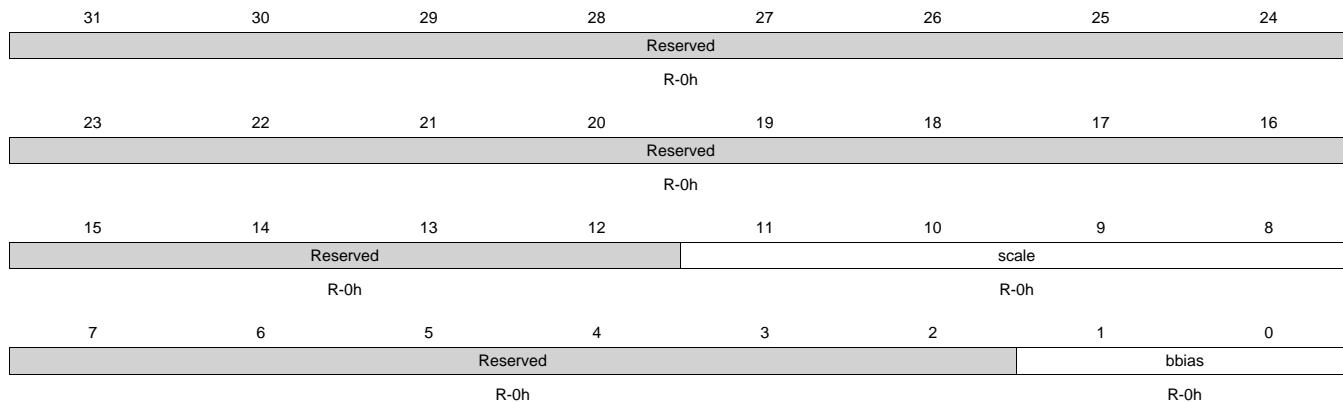
**Table 9-57. vdd\_core\_opp\_100 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | Reserved | R    | 0h    |  |
| 23-0  | ntarget  | R    | 0h    | Ntarget value for CORE Voltage domain OPP100<br>Reset value is device-dependent. |

### **9.3.48 bb\_scale Register (offset = 7D0h) [reset = 0h]**

bb\_scale is shown in [Figure 9-51](#) and described in [Table 9-58](#).

**Figure 9-51. bb\_scale Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

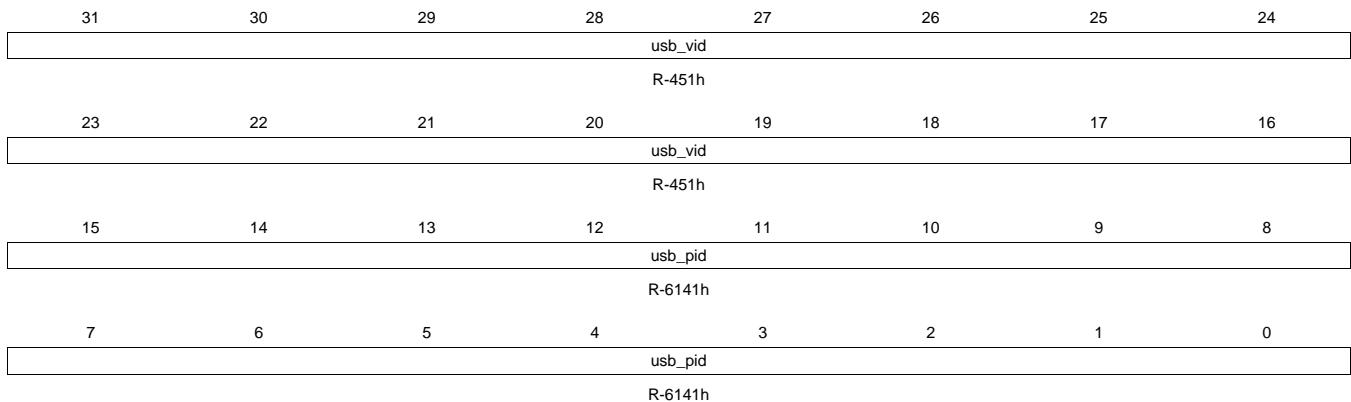
**Table 9-58. bb\_scale Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                              |
|-------|----------|------|-------|--|
| 31-12 | Reserved | R    | 0h    |  |
| 11-8  | scale    | R    | 0h    | Dynamic core voltage scaling for class 0 |
| 7-2   | Reserved | R    | 0h    |  |
| 1-0   | bbias    | R    | 0h    | BBIAS value from Efuse                   |

### 9.3.49 *usb\_vid\_pid* Register (offset = 7F4h) [reset = 4516141h]

*usb\_vid\_pid* is shown in [Figure 9-52](#) and described in [Table 9-59](#).

**Figure 9-52. *usb\_vid\_pid* Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-59. *usb\_vid\_pid* Register Field Descriptions**

| Bit   | Field   | Type | Reset  | Description    |
|-------|---------|------|--------|----------------|
| 31-16 | usb_vid | R    | 0x451  | USB Vendor ID  |
| 15-0  | usb_pid | R    | 0x6141 | USB Product ID |

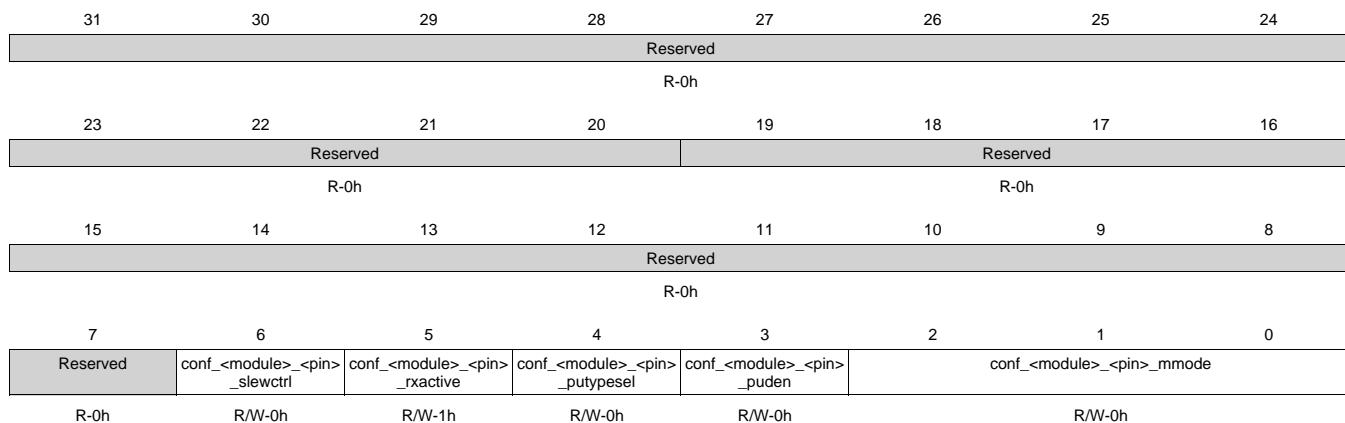
### 9.3.50 conf\_<module>\_<pin> Register (offset = 800h–A34h)

See the device datasheet for information on default pin mux configurations. Note that the device ROM may change the default pin mux for certain pins based on the SYSBOOT mode settings.

See [Table 9-10, Control Module Registers Table](#), for the full list of offsets for each module/pin configuration.

conf\_<module>\_<pin> is shown in [Figure 9-53](#) and described in [Table 9-60](#).

**Figure 9-53. conf\_<module>\_<pin> Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-60. conf\_<module>\_<pin> Register Field Descriptions**

| Bit   | Field                         | Type | Reset | Description  |
|-------|-------------------------------|------|-------|--|
| 31-20 | Reserved                      | R    | 0h    |  |
| 19-7  | Reserved                      | R    | 0h    |  |
| 6     | conf_<module>_<pin>_slewctrl  | R/W  | X     | Select between faster or slower slew rate<br>0: Fast<br>1: Slow<br>Reset value is pad-dependent.                         |
| 5     | conf_<module>_<pin>_rxactive  | R/W  | 1h    | Input enable value for the PAD<br>0: Receiver disabled<br>1: Receiver enabled  |
| 4     | conf_<module>_<pin>_putypesel | R/W  | X     | Pad pullup/pulldown type selection<br>0: Pulldown selected<br>1: Pullup selected<br>Reset value is pad-dependent.        |
| 3     | conf_<module>_<pin>_puoden    | R/W  | X     | Pad pullup/pulldown enable<br>0: Pullup/pulldown enabled<br>1: Pullup/pulldown disabled<br>Reset value is pad-dependent. |
| 2-0   | conf_<module>_<pin>_mmode     | R/W  | X     | Pad functional signal mux select.<br>Reset value is pad-dependent.   |

### 9.3.51 cqdetect\_status Register (offset = E00h) [reset = 0h]

cqdetect\_status is shown in Figure 9-54 and described in Table 9-61.

**Figure 9-54. cqdetect\_status Register**

| 31       | 30             | 29            | 28            | 27              | 26              | 25          | 24       |
|----------|----------------|---------------|---------------|-----------------|-----------------|-------------|----------|
| Reserved |                |               |               |                 |                 |             |          |
| R-0h     |                |               |               |                 |                 |             |          |
| 23       | 22             | 21            | 20            | 19              | 18              | 17          | 16       |
| Reserved | Reserved       | Reserved      | Reserved      | Reserved        | Reserved        | Reserved    | Reserved |
| R-0h     | R-0h           | R-0h          | R-0h          | R-0h            | R-0h            | R-0h        | R-0h     |
| 15       | 14             | 13            | 12            | 11              | 10              | 9           | 8        |
| Reserved | cqerr_general  | cqerr_gmac_b  | cqerr_gmac_a  | cqerr_mmcbsd_b  | cqerr_mmcbsd_a  | cqerr_gpmc  |          |
| R-0h     | R-0h           | R-0h          | R-0h          | R-0h            | R-0h            | R-0h        | R-0h     |
| 7        | 6              | 5             | 4             | 3               | 2               | 1           | 0        |
| Reserved | cqstat_general | cqstat_gmac_b | cqstat_gmac_a | cqstat_mmcbsd_b | cqstat_mmcbsd_a | cqstat_gpmc |          |
| R-0h     | R-0h           | R-0h          | R-0h          | R-0h            | R-0h            | R-0h        | R-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-61. cqdetect\_status Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description                                  |
|-------|-----------------|------|-------|--|
| 31-22 | Reserved        | R    | 0h    |  |
| 21    | Reserved        | R    | 0h    |  |
| 20    | Reserved        | R    | 0h    |  |
| 19    | Reserved        | R    | 0h    |  |
| 18    | Reserved        | R    | 0h    |  |
| 17    | Reserved        | R    | 0h    |  |
| 16    | Reserved        | R    | 0h    |  |
| 15-14 | Reserved        | R    | 0h    |  |
| 13    | cqerr_general   | R    | 0h    | CQDetect Mode Error Status                   |
| 12    | cqerr_gmac_b    | R    | 0h    | CQDetect Mode Error Status                   |
| 11    | cqerr_gmac_a    | R    | 0h    | CQDetect Mode Error Status                   |
| 10    | cqerr_mmcbsd_b  | R    | 0h    | CQDetect Mode Error Status                   |
| 9     | cqerr_mmcbsd_a  | R    | 0h    | CQDetect Mode Error Status                   |
| 8     | cqerr_gpmc      | R    | 0h    | CQDetect Mode Error Status                   |
| 7-6   | Reserved        | R    | 0h    |  |
| 5     | cqstat_general  | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |
| 4     | cqstat_gmac_b   | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |
| 3     | cqstat_gmac_a   | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |
| 2     | cqstat_mmcbsd_b | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |
| 1     | cqstat_mmcbsd_a | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |
| 0     | cqstat_gpmc     | R    | 0h    | 1: IOs are 3.3V mode<br>0: IOs are 1.8V mode |

### 9.3.52 ddr\_io\_ctrl Register (offset = E04h) [reset = 0h]

ddr\_io\_ctrl is shown in [Figure 9-55](#) and described in [Table 9-62](#).

**Figure 9-55. ddr\_io\_ctrl Register**

| 31               | 30                | 29       | 28       | 27       | 26       | 25 | 24 |
|------------------|-------------------|----------|----------|----------|----------|----|----|
| ddr3_rst_def_val | ddr_wuclk_disable | Reserved | mddr_sel |          | Reserved |    |    |
| R/W-0h           | R/W-0h            | R-0h     | R/W-0h   |          | R/W-0h   |    |    |
| 23               | 22                | 21       | 20       | 19       | 18       | 17 | 16 |
|                  |                   |          |          | Reserved |          |    |    |
|                  |                   |          |          | R/W-0h   |          |    |    |
| 15               | 14                | 13       | 12       | 11       | 10       | 9  | 8  |
|                  |                   |          |          | Reserved |          |    |    |
|                  |                   |          |          | R/W-0h   |          |    |    |
| 7                | 6                 | 5        | 4        | 3        | 2        | 1  | 0  |
|                  |                   |          |          | Reserved |          |    |    |
|                  |                   |          |          | R/W-0h   |          |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

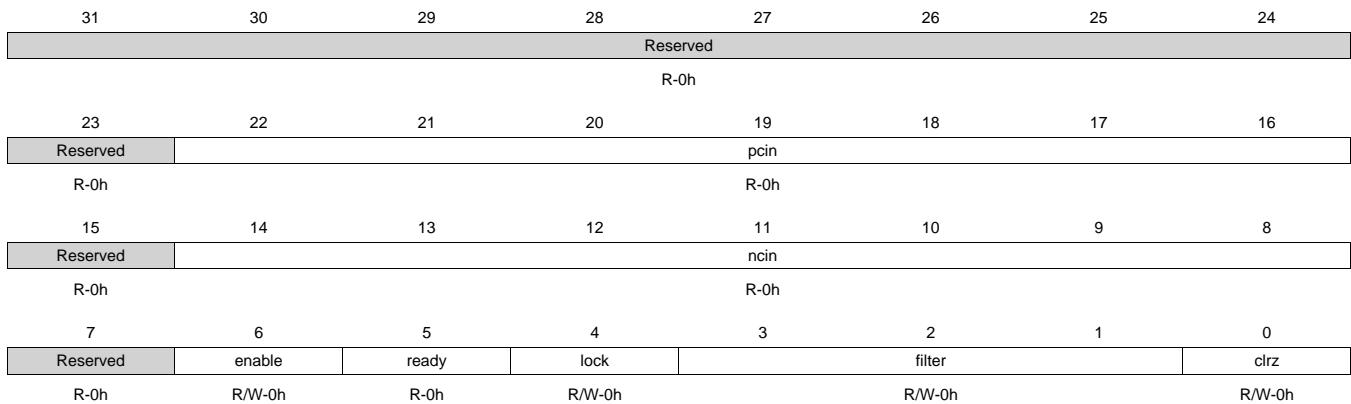
**Table 9-62. ddr\_io\_ctrl Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 31   | ddr3_rst_def_val  | R/W  | 0h    | DDR3 reset default value   |
| 30   | ddr_wuclk_disable | R/W  | 0h    | Disables the slow clock to WUCLKIN and ISOCLKIN of DDR emif SS and IOs (required for proper initialization, after which clock could be shut off).<br>0 = free running SLOW (32k) clock<br>1 = clock is synchronously gated |
| 29   | Reserved          | R    | 0h    |  |
| 28   | mddr_sel          | R/W  | 0h    | 0: IOs set for DDR2/DDR3 (STL mode)<br>1: IOs set for mDDR (CMOS mode)   |
| 27-0 | Reserved          | R/W  | 0h    |  |

### 9.3.53 vtp\_ctrl Register (offset = E0Ch) [reset = 0h]

vtp\_ctrl is shown in Figure 9-56 and described in Table 9-63.

**Figure 9-56. vtp\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

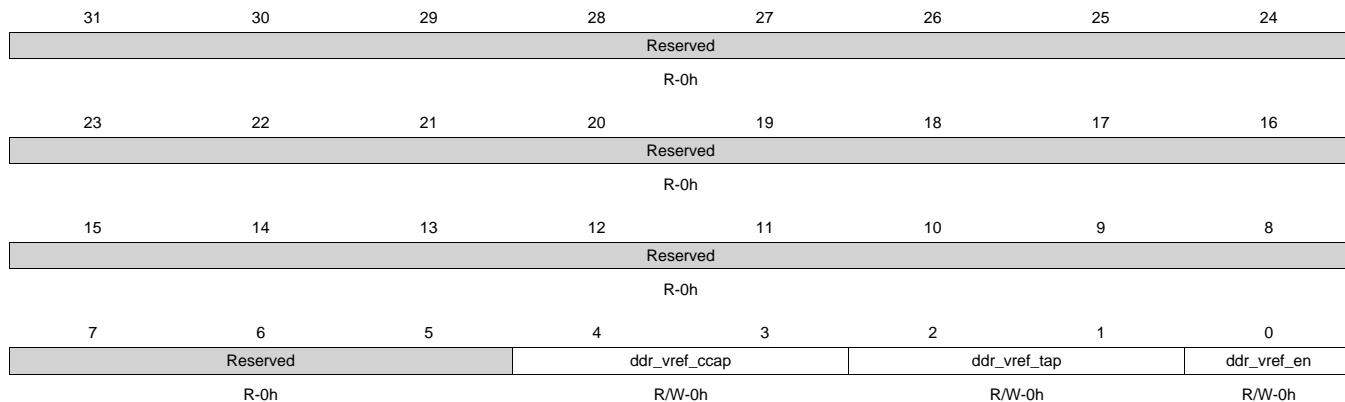
**Table 9-63. vtp\_ctrl Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-23 | Reserved | R    | 0h    |   |
| 22-16 | pcin     | R    | 0h    | Default/reset values of 'P' for the VTP controller  |
| 15    | Reserved | R    | 0h    |   |
| 14-8  | ncin     | R    | 0h    | Default/reset values of 'N' for the VTP controller  |
| 7     | Reserved | R    | 0h    |   |
| 6     | enable   | R/W  | 0h    | 0: VTP macro in bypass mode. P and N are driven from PCIN and NCIN.<br>1: Dynamic VTP compensation mode   |
| 5     | ready    | R    | 0h    | 0: Training sequence is not complete<br>1: Training sequence is complete  |
| 4     | lock     | R/W  | 0h    | 0: Normal operation dynamic update<br>1: freeze dynamic update, pwrndn controller   |
| 3-1   | filter   | R/W  | 0h    | Digital filter bits to prevent the controller from making excessive number of changes.<br>000: Filter off<br>001: Update on two consecutive update requests<br>010: Update on three consecutive update requests<br>011: Update on four consecutive update requests<br>100: Update on five consecutive update requests<br>101: Update on six consecutive update requests<br>110: Update on seven consecutive update requests<br>111: Update on eight consecutive update requests |
| 0     | clrz     | R/W  | 0h    | clears flops, start count again, after low going pulse  |

### 9.3.54 vref\_ctrl Register (offset = E14h) [reset = 0h]

vref\_ctrl is shown in [Figure 9-57](#) and described in [Table 9-64](#).

**Figure 9-57. vref\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-64. vref\_ctrl Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-5 | Reserved      | R    | 0h    |  |
| 4-3  | ddr_vref_ccap | R/W  | 0h    | select for coupling cap for DDR<br>00 : No capacitor connected<br>01 : Capacitor between BIAS2 and VSS<br>10 : Capacitor between BIAS2 and VDDS<br>11: Capacitor between BIAS2 and VSS andamp<br>Capacitor between BIAS2 and VDDS  |
| 2-1  | ddr_vref_tap  | R/W  | 0h    | select for int ref for DDR<br>00 : Pad/Bias2 connected to internal reference VDDS/2 for 2uA current load<br>01 : Pad/Bias2 connected to internal reference VDDS/2 for 4uA current load<br>10 : Pad/Bias2 connected to internal reference VDDS/2 for 6uA current load<br>11 : Pad/Bias2 connected to internal reference VDDS/2 for 8uA current load |
| 0    | ddr_vref_en   | R/W  | 0h    | active high internal reference enable for DDR  |

### 9.3.55 tpcc\_evt\_mux\_0\_3 Register (offset = F90h) [reset = 0h]

tpcc\_evt\_mux\_0\_3 is shown in [Figure 9-58](#) and described in [Table 9-65](#).

**Figure 9-58. tpcc\_evt\_mux\_0\_3 Register**

|          |    |    |    |           |    |    |    |
|----------|----|----|----|-----------|----|----|----|
| 31       | 30 | 29 | 28 | 27        | 26 | 25 | 24 |
| Reserved |    |    |    | evt_mux_3 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 23       | 22 | 21 | 20 | 19        | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_2 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 15       | 14 | 13 | 12 | 11        | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_1 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 7        | 6  | 5  | 4  | 3         | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_0 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-65. tpcc\_evt\_mux\_0\_3 Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description                            |
|-------|-----------|------|-------|--|
| 31-30 | Reserved  | R    | 0h    |  |
| 29-24 | evt_mux_3 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 3 |
| 23-22 | Reserved  | R    | 0h    |  |
| 21-16 | evt_mux_2 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 2 |
| 15-14 | Reserved  | R    | 0h    |  |
| 13-8  | evt_mux_1 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 1 |
| 7-6   | Reserved  | R    | 0h    |  |
| 5-0   | evt_mux_0 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 0 |

### **9.3.56 tpcc\_evt\_mux\_4\_7 Register (offset = F94h) [reset = 0h]**

tpcc\_evt\_mux\_4\_7 is shown in [Figure 9-59](#) and described in [Table 9-66](#).

**Figure 9-59. tpcc\_evt\_mux\_4\_7 Register**

|          |    |    |    |           |    |    |    |
|----------|----|----|----|-----------|----|----|----|
| 31       | 30 | 29 | 28 | 27        | 26 | 25 | 24 |
| Reserved |    |    |    | evt_mux_7 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 23       | 22 | 21 | 20 | 19        | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_6 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 15       | 14 | 13 | 12 | 11        | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_5 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |
| 7        | 6  | 5  | 4  | 3         | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_4 |    |    |    |
| R-0h     |    |    |    | R/W-0h    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-66. tpcc\_evt\_mux\_4\_7 Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description                            |
|-------|-----------|------|-------|--|
| 31-30 | Reserved  | R    | 0h    |  |
| 29-24 | evt_mux_7 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 7 |
| 23-22 | Reserved  | R    | 0h    |  |
| 21-16 | evt_mux_6 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 6 |
| 15-14 | Reserved  | R    | 0h    |  |
| 13-8  | evt_mux_5 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 5 |
| 7-6   | Reserved  | R    | 0h    |  |
| 5-0   | evt_mux_4 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 4 |

### 9.3.57 tpcc\_evt\_mux\_8\_11 Register (offset = F98h) [reset = 0h]

tpcc\_evt\_mux\_8\_11 is shown in [Figure 9-60](#) and described in [Table 9-67](#).

**Figure 9-60. tpcc\_evt\_mux\_8\_11 Register**

|          |    |    |    |            |    |    |    |
|----------|----|----|----|------------|----|----|----|
| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
| Reserved |    |    |    | evt_mux_11 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_10 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_9  |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_8  |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-67. tpcc\_evt\_mux\_8\_11 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_11 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 11 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_10 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 10 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_9  | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 9  |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_8  | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 8  |

### **9.3.58 tpcc\_evt\_mux\_12\_15 Register (offset = F9Ch) [reset = 0h]**

tpcc\_evt\_mux\_12\_15 is shown in [Figure 9-61](#) and described in [Table 9-68](#).

**Figure 9-61. tpcc\_evt\_mux\_12\_15 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_15 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_14 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_13 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_12 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-68. tpcc\_evt\_mux\_12\_15 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_15 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 15 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_14 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 14 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_13 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 13 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_12 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 12 |

### 9.3.59 tpcc\_evt\_mux\_16\_19 Register (offset = FA0h) [reset = 0h]

tpcc\_evt\_mux\_16\_19 is shown in [Figure 9-62](#) and described in [Table 9-69](#).

**Figure 9-62. tpcc\_evt\_mux\_16\_19 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_19 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_18 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_17 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_16 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-69. tpcc\_evt\_mux\_16\_19 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_19 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 19 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_18 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 18 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_17 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 17 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_16 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 16 |

### **9.3.60 tpcc\_evt\_mux\_20\_23 Register (offset = FA4h) [reset = 0h]**

tpcc\_evt\_mux\_20\_23 is shown in [Figure 9-63](#) and described in [Table 9-70](#).

**Figure 9-63. tpcc\_evt\_mux\_20\_23 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_23 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_22 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_21 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_20 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-70. tpcc\_evt\_mux\_20\_23 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_23 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 23 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_22 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 22 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_21 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 21 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_20 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 20 |

### 9.3.61 tpcc\_evt\_mux\_24\_27 Register (offset = FA8h) [reset = 0h]

tpcc\_evt\_mux\_24\_27 is shown in [Figure 9-64](#) and described in [Table 9-71](#).

**Figure 9-64. tpcc\_evt\_mux\_24\_27 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_27 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_26 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_25 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_24 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-71. tpcc\_evt\_mux\_24\_27 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_27 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 27 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_26 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 26 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_25 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 25 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_24 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 24 |

### **9.3.62 tpcc\_evt\_mux\_28\_31 Register (offset = FACH) [reset = 0h]**

tpcc\_evt\_mux\_28\_31 is shown in [Figure 9-65](#) and described in [Table 9-72](#).

**Figure 9-65. tpcc\_evt\_mux\_28\_31 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_31 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_30 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_29 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_28 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-72. tpcc\_evt\_mux\_28\_31 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_31 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 31 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_30 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 30 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_29 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 29 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_28 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 28 |

### 9.3.63 tpcc\_evt\_mux\_32\_35 Register (offset = FB0h) [reset = 0h]

tpcc\_evt\_mux\_32\_35 is shown in [Figure 9-66](#) and described in [Table 9-73](#).

**Figure 9-66. tpcc\_evt\_mux\_32\_35 Register**

|          |    |    |    |            |    |    |    |
|----------|----|----|----|------------|----|----|----|
| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
| Reserved |    |    |    | evt_mux_35 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_34 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_33 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_32 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-73. tpcc\_evt\_mux\_32\_35 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_35 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 35 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_34 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 34 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_33 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 33 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_32 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 32 |

### **9.3.64 tpcc\_evt\_mux\_36\_39 Register (offset = FB4h) [reset = 0h]**

tpcc\_evt\_mux\_36\_39 is shown in [Figure 9-67](#) and described in [Table 9-74](#).

**Figure 9-67. tpcc\_evt\_mux\_36\_39 Register**

|          |    |    |    |            |    |    |    |
|----------|----|----|----|------------|----|----|----|
| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
| Reserved |    |    |    | evt_mux_39 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_38 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_37 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_36 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-74. tpcc\_evt\_mux\_36\_39 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_39 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 39 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_38 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 38 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_37 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 37 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_36 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 36 |

### 9.3.65 tpcc\_evt\_mux\_40\_43 Register (offset = FB8h) [reset = 0h]

tpcc\_evt\_mux\_40\_43 is shown in [Figure 9-68](#) and described in [Table 9-75](#).

**Figure 9-68. tpcc\_evt\_mux\_40\_43 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_43 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_42 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_41 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_40 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-75. tpcc\_evt\_mux\_40\_43 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_43 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 43 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_42 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 42 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_41 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 41 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_40 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 40 |

### **9.3.66 tpcc\_evt\_mux\_44\_47 Register (offset = FBCh) [reset = 0h]**

tpcc\_evt\_mux\_44\_47 is shown in [Figure 9-69](#) and described in [Table 9-76](#).

**Figure 9-69. tpcc\_evt\_mux\_44\_47 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_47 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_46 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_45 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_44 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-76. tpcc\_evt\_mux\_44\_47 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_47 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 47 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_46 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 46 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_45 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 45 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_44 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 44 |

### 9.3.67 tpcc\_evt\_mux\_48\_51 Register (offset = FC0h) [reset = 0h]

tpcc\_evt\_mux\_48\_51 is shown in [Figure 9-70](#) and described in [Table 9-77](#).

**Figure 9-70. tpcc\_evt\_mux\_48\_51 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_51 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_50 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_49 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_48 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-77. tpcc\_evt\_mux\_48\_51 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_51 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 51 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_50 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 50 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_49 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 49 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_48 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 48 |

### **9.3.68 tpcc\_evt\_mux\_52\_55 Register (offset = FC4h) [reset = 0h]**

tpcc\_evt\_mux\_52\_55 is shown in [Figure 9-71](#) and described in [Table 9-78](#).

**Figure 9-71. tpcc\_evt\_mux\_52\_55 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_55 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_54 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_53 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_52 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-78. tpcc\_evt\_mux\_52\_55 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_55 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 55 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_54 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 54 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_53 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 53 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_52 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 52 |

### 9.3.69 tpcc\_evt\_mux\_56\_59 Register (offset = FC8h) [reset = 0h]

tpcc\_evt\_mux\_56\_59 is shown in [Figure 9-72](#) and described in [Table 9-79](#).

**Figure 9-72. tpcc\_evt\_mux\_56\_59 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_59 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_58 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_57 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_56 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-79. tpcc\_evt\_mux\_56\_59 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_59 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 59 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_58 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 58 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_57 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 57 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_56 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 56 |

### **9.3.70 tpcc\_evt\_mux\_60\_63 Register (offset = FCCh) [reset = 0h]**

tpcc\_evt\_mux\_60\_63 is shown in [Figure 9-73](#) and described in [Table 9-80](#).

**Figure 9-73. tpcc\_evt\_mux\_60\_63 Register**

| 31       | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|----------|----|----|----|------------|----|----|----|
| Reserved |    |    |    | evt_mux_63 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 23       | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
| Reserved |    |    |    | evt_mux_62 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 15       | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
| Reserved |    |    |    | evt_mux_61 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |
| 7        | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
| Reserved |    |    |    | evt_mux_60 |    |    |    |
| R-0h     |    |    |    | R/W-0h     |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

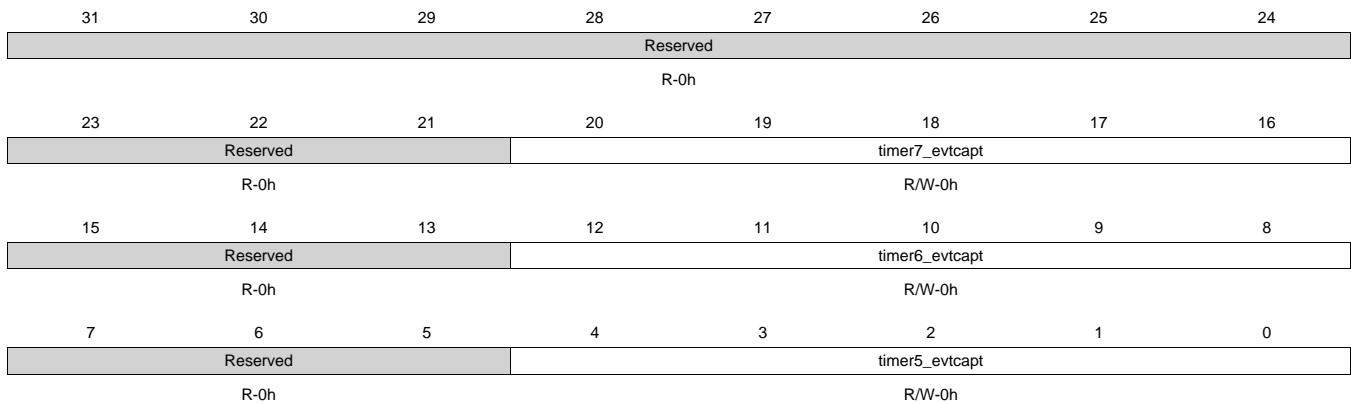
**Table 9-80. tpcc\_evt\_mux\_60\_63 Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description                             |
|-------|------------|------|-------|---|
| 31-30 | Reserved   | R    | 0h    |   |
| 29-24 | evt_mux_63 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 63 |
| 23-22 | Reserved   | R    | 0h    |   |
| 21-16 | evt_mux_62 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 62 |
| 15-14 | Reserved   | R    | 0h    |   |
| 13-8  | evt_mux_61 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 61 |
| 7-6   | Reserved   | R    | 0h    |   |
| 5-0   | evt_mux_60 | R/W  | 0h    | Selects 1 of 64 inputs for DMA event 60 |

### 9.3.71 timer\_evt\_capt Register (offset = FD0h) [reset = 0h]

timer\_evt\_capt is shown in Figure 9-74 and described in Table 9-81.

**Figure 9-74. timer\_evt\_capt Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

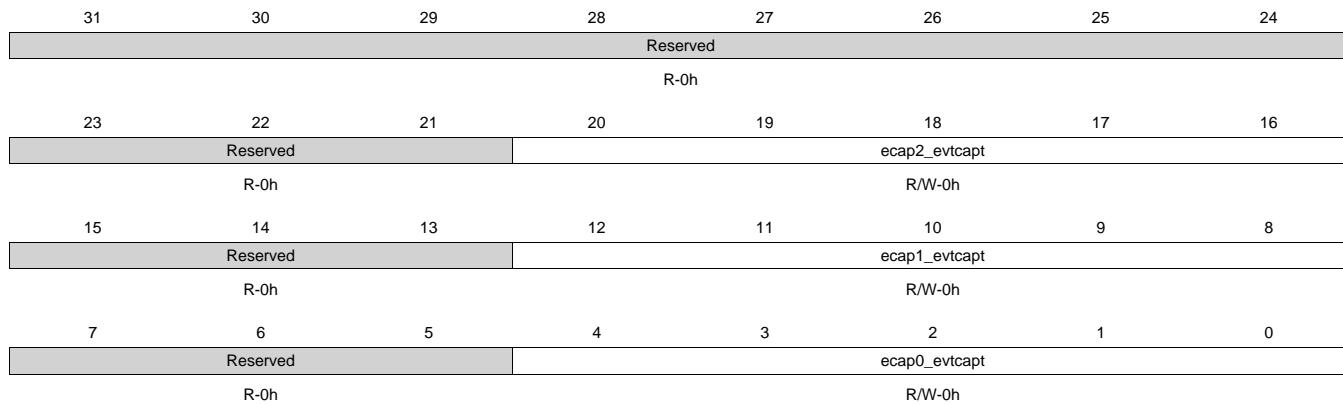
**Table 9-81. timer\_evt\_capt Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description               |
|-------|----------------|------|-------|---------------------------|
| 31-21 | Reserved       | R    | 0h    |                           |
| 20-16 | timer7_evtcapt | R/W  | 0h    | Timer 7 event capture mux |
| 15-13 | Reserved       | R    | 0h    |                           |
| 12-8  | timer6_evtcapt | R/W  | 0h    | Timer 6 event capture mux |
| 7-5   | Reserved       | R    | 0h    |                           |
| 4-0   | timer5_evtcapt | R/W  | 0h    | Timer 5 event capture mux |

### 9.3.72 ecap\_evt\_capt Register (offset = FD4h) [reset = 0h]

ecap\_evt\_capt is shown in Figure 9-75 and described in Table 9-82.

**Figure 9-75. ecap\_evt\_capt Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

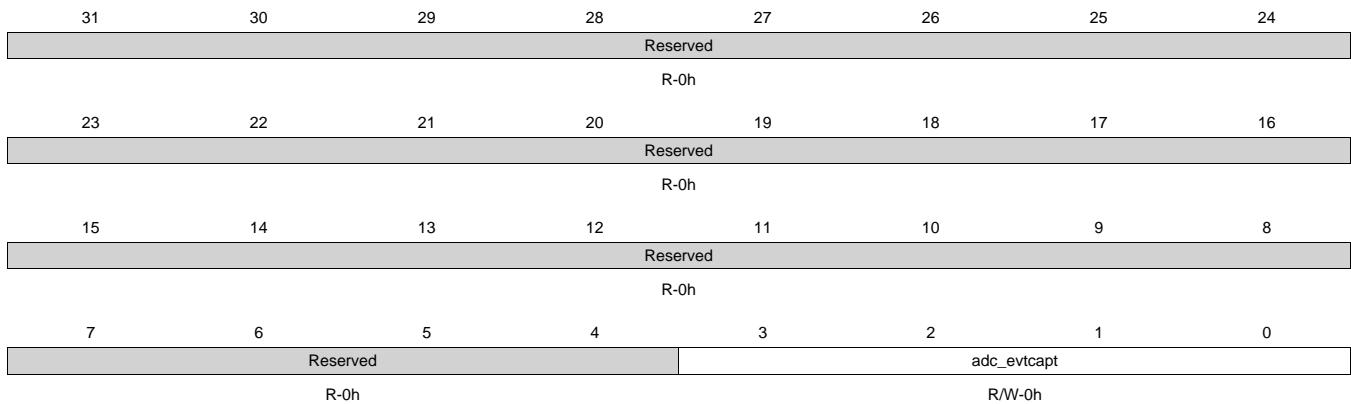
**Table 9-82. ecap\_evt\_capt Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description             |
|-------|---------------|------|-------|-------------------------|
| 31-21 | Reserved      | R    | 0h    |                         |
| 20-16 | ecap2_evtcapt | R/W  | 0h    | ECAP2 event capture mux |
| 15-13 | Reserved      | R    | 0h    |                         |
| 12-8  | ecap1_evtcapt | R/W  | 0h    | ECAP1 event capture mux |
| 7-5   | Reserved      | R    | 0h    |                         |
| 4-0   | ecap0_evtcapt | R/W  | 0h    | ECAP0 event capture mux |

### 9.3.73 adc\_evt\_capt Register (offset = FD8h) [reset = 0h]

adc\_evt\_capt is shown in Figure 9-76 and described in Table 9-83.

**Figure 9-76. adc\_evt\_capt Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

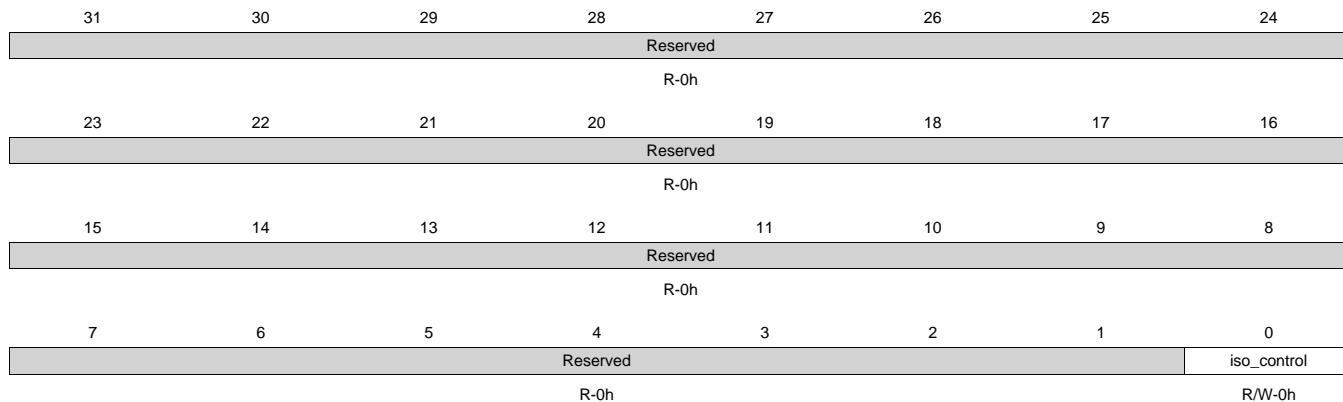
**Table 9-83. adc\_evt\_capt Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description             |
|------|-------------|------|-------|-------------------------|
| 31-4 | Reserved    | R    | 0h    |                         |
| 3-0  | adc_evtcapt | R/W  | 0h    | ECAP0 event capture mux |

### **9.3.74 reset\_iso Register (offset = 1000h) [reset = 0h]**

reset\_iso is shown in [Figure 9-77](#) and described in [Table 9-84](#).

**Figure 9-77. reset\_iso Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

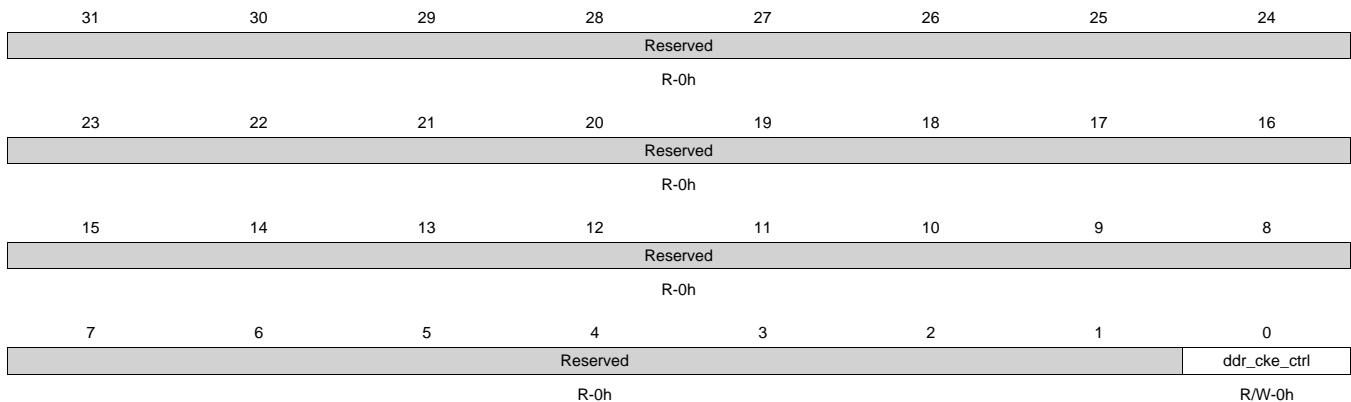
**Table 9-84. reset\_iso Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-1 | Reserved    | R    | 0h    |  |
| 0    | iso_control | R/W  | 0h    | 0 : Ethernet Switch is not isolated<br>1 : Ethernet Switch is isolated |

### 9.3.75 ddr\_cke\_ctrl Register (offset = 131Ch) [reset = 0h]

ddr\_cke\_ctrl is shown in [Figure 9-78](#) and described in [Table 9-85](#).

**Figure 9-78. ddr\_cke\_ctrl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

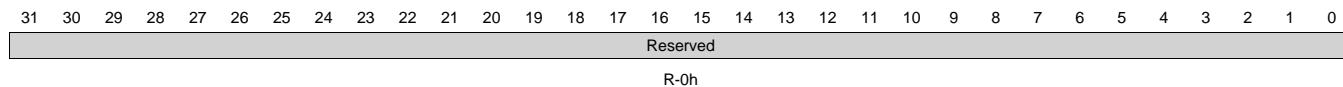
**Table 9-85. ddr\_cke\_ctrl Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-1 | Reserved     | R    | 0h    |   |
| 0    | ddr_cke_ctrl | R/W  | 0h    | CKE from EMIF/DDDRPHY is ANDed with this bit.<br>0: CKE to memories gated off to zero. External DRAM memories will not able to register DDR commands from device<br>1: Normal operation. CKE is now controlled by EMIF/DDR PHY. |

### **9.3.76 sma2 Register (offset = 1320h) [reset = 0h]**

sma2 is shown in [Figure 9-79](#) and described in [Table 9-86](#).

**Figure 9-79. sma2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

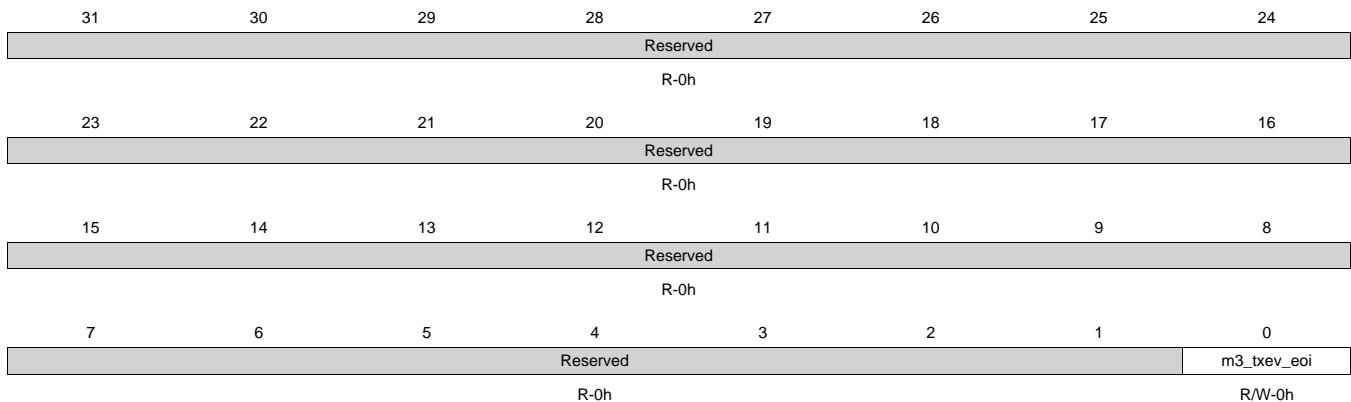
**Table 9-86. sma2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-0 | Reserved | R    | 0h    |             |

### 9.3.77 m3\_txev\_eoi Register (offset = 1324h) [reset = 0h]

m3\_txev\_eoi is shown in [Figure 9-80](#) and described in [Table 9-87](#).

**Figure 9-80. m3\_txev\_eoi Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-87. m3\_txev\_eoi Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-1 | Reserved    | R    | 0h    |  |
| 0    | m3_txev_eoi | R/W  | 0h    | <p>TXEV (Event) from M3 processor is a pulse signal connected as interrupt to MPU IRQ[78]. Since MPU expects level signals. The TXEV pulse from M3 is converted to a level in glue logic. The logic works as follows:</p> <ul style="list-style-type: none"> <li>-On a 0-1 transition on TXEV, the IRQ[78] is set.</li> <li>-For clearing the interrupt, S/W must do the following:<br/>S/W must clear the IRQ[78] by writing a 1 to M3_TXEV_EOI bit in this register.<br/>This bit is sticky and for re-arming the IRQ[78], S/W must write a 0 to this field in the ISR.</li> </ul> |

### **9.3.78 ipc\_msg\_reg0 Register (offset = 1328h) [reset = 0h]**

ipc\_msg\_reg0 is shown in [Figure 9-81](#) and described in [Table 9-88](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-81. ipc\_msg\_reg0 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-88. ipc\_msg\_reg0 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg0 | R/W  | 0h    | Inter Processor Messaging Register |

### 9.3.79 ipc\_msg\_reg1 Register (offset = 132Ch) [reset = 0h]

ipc\_msg\_reg1 is shown in [Figure 9-82](#) and described in [Table 9-89](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-82. ipc\_msg\_reg1 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-89. ipc\_msg\_reg1 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg1 | R/W  | 0h    | Inter Processor Messaging Register |

### **9.3.80 ipc\_msg\_reg2 Register (offset = 1330h) [reset = 0h]**

ipc\_msg\_reg2 is shown in [Figure 9-83](#) and described in [Table 9-90](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-83. ipc\_msg\_reg2 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-90. ipc\_msg\_reg2 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg2 | R/W  | 0h    | Inter Processor Messaging Register |

### 9.3.81 ipc\_msg\_reg3 Register (offset = 1334h) [reset = 0h]

ipc\_msg\_reg3 is shown in [Figure 9-84](#) and described in [Table 9-91](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-84. ipc\_msg\_reg3 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg3 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-91. ipc\_msg\_reg3 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg3 | R/W  | 0h    | Inter Processor Messaging Register |

### **9.3.82 ipc\_msg\_reg4 Register (offset = 1338h) [reset = 0h]**

ipc\_msg\_reg4 is shown in [Figure 9-85](#) and described in [Table 9-92](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-85. ipc\_msg\_reg4 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg4 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-92. ipc\_msg\_reg4 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg4 | R/W  | 0h    | Inter Processor Messaging Register |

### 9.3.83 ipc\_msg\_reg5 Register (offset = 133Ch) [reset = 0h]

ipc\_msg\_reg5 is shown in [Figure 9-86](#) and described in [Table 9-93](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-86. ipc\_msg\_reg5 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg5 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-93. ipc\_msg\_reg5 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg5 | R/W  | 0h    | Inter Processor Messaging Register |

### **9.3.84 ipc\_msg\_reg6 Register (offset = 1340h) [reset = 0h]**

ipc\_msg\_reg6 is shown in [Figure 9-87](#) and described in [Table 9-94](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-87. ipc\_msg\_reg6 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg6 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-94. ipc\_msg\_reg6 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg6 | R/W  | 0h    | Inter Processor Messaging Register |

### 9.3.85 ipc\_msg\_reg7 Register (offset = 1344h) [reset = 0h]

ipc\_msg\_reg7 is shown in [Figure 9-88](#) and described in [Table 9-95](#). This register is typically used for messaging between Cortex A8 and CortexM3 (WKUP). The contents of this register is user defined.

**Figure 9-88. ipc\_msg\_reg7 Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ipc_msg_reg7 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-95. ipc\_msg\_reg7 Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                        |
|------|--------------|------|-------|------------------------------------|
| 31-0 | ipc_msg_reg7 | R/W  | 0h    | Inter Processor Messaging Register |

### 9.3.86 ddr\_cmd0\_iocrtl Register (offset = 1404h) [reset = 0h]

ddr\_cmd0\_iocrtl is shown in Figure 9-89 and described in Table 9-96.

**Figure 9-89. ddr\_cmd0\_iocrtl Register**

|                  |    |    |              |                  |    |                  |    |
|------------------|----|----|--------------|------------------|----|------------------|----|
| 31               | 30 | 29 | 28           | 27               | 26 | 25               | 24 |
| io_config_gp_wd1 |    |    |              |                  |    |                  |    |
| R/W-0h           |    |    |              |                  |    |                  |    |
| 23               | 22 | 21 | 20           | 19               | 18 | 17               | 16 |
| io_config_gp_wd1 |    |    |              | io_config_gp_wd0 |    |                  |    |
| R/W-0h           |    |    |              | R/W-0h           |    |                  |    |
| 15               | 14 | 13 | 12           | 11               | 10 | 9                | 8  |
| io_config_gp_wd0 |    |    |              |                  |    | io_config_sr_clk |    |
| R/W-0h           |    |    |              |                  |    | R-0h             |    |
| 7                | 6  | 5  | 4            | 3                | 2  | 1                | 0  |
| io_config_i_clk  |    |    | io_config_sr |                  |    | io_config_i      |    |
| R/W-0h           |    |    | R/W-0h       |                  |    | R/W-0h           |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-96. ddr\_cmd0\_iocrtl Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-21 | io_config_gp_wd1 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0.<br/> For example:<br/> macro pin 0: WD1 is bit 21, WD0 is bit 10<br/> macro pin 1: WD1 is bit 22, WD0 is bit 11<br/> ...<br/> macro pin 10: WD1 is bit 31, WD0 is bit 20<br/> See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/> WD1:WD0<br/> 00: Pullup/Pulldown disabled<br/> 01: Weak pullup enabled<br/> 10: Weak pulldown enabled<br/> 11: Weak keeper enabled</p> |
| 20-10 | io_config_gp_wd0 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0.<br/> For example:<br/> macro pin 0: WD1 is bit 21, WD0 is bit 10<br/> macro pin 1: WD1 is bit 22, WD0 is bit 11<br/> ...<br/> macro pin 10: WD1 is bit 31, WD0 is bit 20<br/> See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/> WD1:WD0<br/> 00: Pullup/Pulldown disabled<br/> 01: Weak pullup enabled<br/> 10: Weak pulldown enabled<br/> 11: Weak keeper enabled</p> |
| 9-8   | io_config_sr_clk | R    | 0h    | <p>2 bit to program clock IO Pads (DDR_CK/DDR_CKN) output slew rate.<br/> These connect as SR1, SR0 to the corresponding DDR IO buffer.<br/> See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 7-5   | io_config_i_clk  | R/W  | 0h    | <p>3-bit configuration input to program clock IO pads (DDR_CK/DDR_CKN) output impedance.<br/> These connect as I2, I1, I0 to the corresponding DDR IO buffer.<br/> See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 4-3   | io_config_sr     | R/W  | 0h    | <p>2 bit to program addr/cmd IO Pads output slew rate.<br/> These connect as SR1, SR0 to the corresponding DDR IO buffer.<br/> See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |

**Table 9-96. ddr\_cmd0\_ioctl Register Field Descriptions (continued)**

| Bit | Field       | Type | Reset | Description  |
|-----|-------------|------|-------|--|
| 2-0 | io_config_i | R/W  | 0h    | 3-bit configuration input to program addr/cmd IO output impedance. These connect as I2, I1, I0 to the corresponding DDR IO buffer. See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits. |

### 9.3.87 ddr\_cmd1\_iocrtl Register (offset = 1408h) [reset = 0h]

ddr\_cmd1\_iocrtl is shown in Figure 9-90 and described in Table 9-97.

**Figure 9-90. ddr\_cmd1\_iocrtl Register**

|                  |    |    |              |                  |    |                  |    |
|------------------|----|----|--------------|------------------|----|------------------|----|
| 31               | 30 | 29 | 28           | 27               | 26 | 25               | 24 |
| io_config_gp_wd1 |    |    |              |                  |    |                  |    |
| R/W-0h           |    |    |              |                  |    |                  |    |
| 23               | 22 | 21 | 20           | 19               | 18 | 17               | 16 |
| io_config_gp_wd1 |    |    |              | io_config_gp_wd0 |    |                  |    |
| R/W-0h           |    |    |              | R/W-0h           |    |                  |    |
| 15               | 14 | 13 | 12           | 11               | 10 | 9                | 8  |
| io_config_gp_wd0 |    |    |              |                  |    | io_config_sr_clk |    |
| R/W-0h           |    |    |              |                  |    | R/W-0h           |    |
| 7                | 6  | 5  | 4            | 3                | 2  | 1                | 0  |
| io_config_i_clk  |    |    | io_config_sr |                  |    | io_config_i      |    |
| R/W-0h           |    |    | R/W-0h       |                  |    | R/W-0h           |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-97. ddr\_cmd1\_iocrtl Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-21 | io_config_gp_wd1 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0. For example:</p> <p>macro pin 0: WD1 is bit 21, WD0 is bit 10</p> <p>macro pin 1: WD1 is bit 22, WD0 is bit 11</p> <p>...</p> <p>macro pin 10: WD1 is bit 31, WD0 is bit 20</p> <p>See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.</p> <p>WD1:WD0</p> <p>00: Pullup/Pulldown disabled</p> <p>01: Weak pullup enabled</p> <p>10: Weak pulldown enabled</p> <p>11: Weak keeper enabled</p> |
| 20-10 | io_config_gp_wd0 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0. For example:</p> <p>macro pin 0: WD1 is bit 21, WD0 is bit 10</p> <p>macro pin 1: WD1 is bit 22, WD0 is bit 11</p> <p>...</p> <p>macro pin 10: WD1 is bit 31, WD0 is bit 20</p> <p>See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.</p> <p>WD1:WD0</p> <p>00: Pullup/Pulldown disabled</p> <p>01: Weak pullup enabled</p> <p>10: Weak pulldown enabled</p> <p>11: Weak keeper enabled</p> |
| 9-8   | io_config_sr_clk | R/W  | 0h    | <p>2 bit to program clock IO Pads (DDR_CK/DDR_CKN) output slew rate.</p> <p>These connect as SR1, SR0 to the corresponding DDR IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>  |
| 7-5   | io_config_i_clk  | R/W  | 0h    | <p>3-bit configuration input to program clock IO pads (DDR_CK/DDR_CKN) output impedance.</p> <p>These connect as I2, I1, I0 to the corresponding DDR IO buffer.</p> <p>See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>  |
| 4-3   | io_config_sr     | R/W  | 0h    | <p>2 bit to program addr/cmd IO Pads output slew rate.</p> <p>These connect as SR1, SR0 to the corresponding DDR IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>  |

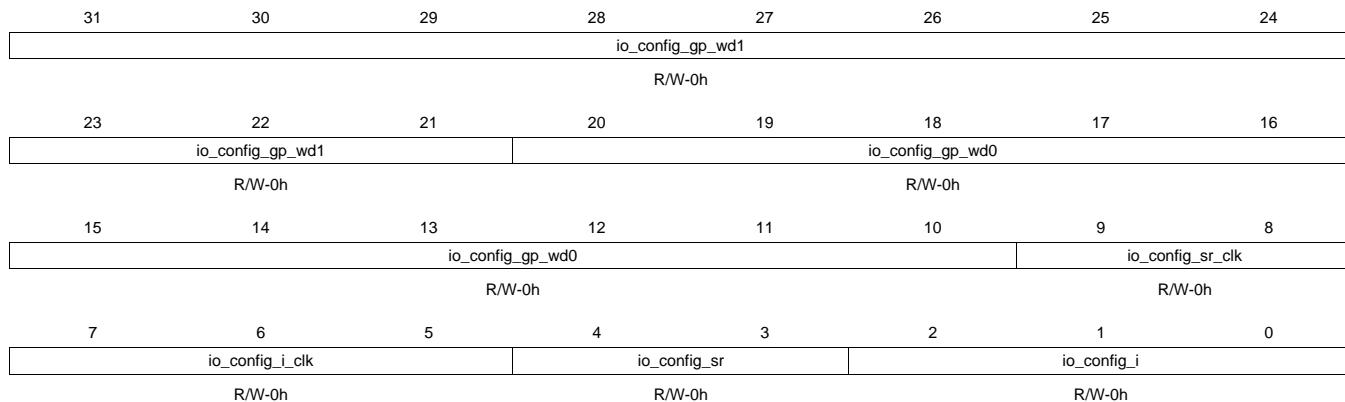
**Table 9-97. ddr\_cmd1\_ioctl Register Field Descriptions (continued)**

| Bit | Field       | Type | Reset | Description  |
|-----|-------------|------|-------|--|
| 2-0 | io_config_i | R/W  | 0h    | 3-bit configuration input to program addr/cmd IO output impedance. These connect as I2, I1, I0 to the corresponding DDR IO buffer. See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits. |

### 9.3.88 ddr\_cmd2\_iocrtl Register (offset = 140Ch) [reset = 0h]

ddr\_cmd2\_iocrtl is shown in [Figure 9-91](#) and described in [Table 9-98](#).

**Figure 9-91. ddr\_cmd2\_iocrtl Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-98. ddr\_cmd2\_iocrtl Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-21 | io_config_gp_wd1 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0.<br/> For example:<br/> macro pin 0: WD1 is bit 21, WD0 is bit 10<br/> macro pin 1: WD1 is bit 22, WD0 is bit 11<br/> ...<br/> macro pin 10: WD1 is bit 31, WD0 is bit 20<br/> See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/> WD1:WD0<br/> 00: Pullup/Pulldown disabled<br/> 01: Weak pullup enabled<br/> 10: Weak pulldown enabled<br/> 11: Weak keeper enabled</p> |
| 20-10 | io_config_gp_wd0 | R/W  | 0h    | <p>There are 2 bits per IO: io_config_gp_wd1 and io_config_gp_wd0.<br/> For example:<br/> macro pin 0: WD1 is bit 21, WD0 is bit 10<br/> macro pin 1: WD1 is bit 22, WD0 is bit 11<br/> ...<br/> macro pin 10: WD1 is bit 31, WD0 is bit 20<br/> See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/> WD1:WD0<br/> 00: Pullup/Pulldown disabled<br/> 01: Weak pullup enabled<br/> 10: Weak pulldown enabled<br/> 11: Weak keeper enabled</p> |
| 9-8   | io_config_sr_clk | R/W  | 0h    | <p>2 bit to program clock IO Pads (DDR_CK/DDR_CKN) output slew rate.<br/> These connect as SR1, SR0 to the corresponding DDR IO buffer.<br/> See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 7-5   | io_config_i_clk  | R/W  | 0h    | <p>3-bit configuration input to program clock IO pads (DDR_CK/DDR_CKN) output impedance.<br/> These connect as I2, I1, I0 to the corresponding DDR IO buffer.<br/> See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 4-3   | io_config_sr     | R/W  | 0h    | <p>2 bit to program addr/cmd IO Pads output slew rate.<br/> These connect as SR1, SR0 to the corresponding DDR IO buffer.<br/> See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |

**Table 9-98. ddr\_cmd2\_ioctl Register Field Descriptions (continued)**

| Bit | Field       | Type | Reset | Description  |
|-----|-------------|------|-------|--|
| 2-0 | io_config_i | R/W  | 0h    | 3-bit configuration input to program addr/cmd IO output impedance. These connect as I2, I1, I0 to the corresponding DDR IO buffer. See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits. |

### 9.3.89 ddr\_data0\_iocrtl Register (offset = 1440h) [reset = 0h]

ddr\_data0\_iocrtl is shown in Figure 9-92 and described in Table 9-99.

**Figure 9-92. ddr\_data0\_iocrtl Register**

|                  |    |                   |                   |                  |                  |                  |        |
|------------------|----|-------------------|-------------------|------------------|------------------|------------------|--------|
| 31               | 30 | 29                | 28                | 27               | 26               | 25               | 24     |
| Reserved         |    | io_config_wd1_dqs | io_config_wd1_dm  | io_config_wd1_dq |                  |                  |        |
| R-0h             |    |                   | R/W-0h            | R/W-0h           |                  |                  | R/W-0h |
| 23               | 22 | 21                | 20                | 19               | 18               | 17               | 16     |
| io_config_wd1_dq |    |                   | io_config_wd0_dqs | io_config_wd0_dm | io_config_wd0_dq | io_config_wd0_dq |        |
| R/W-0h           |    |                   | R/W-0h            | R/W-0h           | R/W-0h           | R/W-0h           |        |
| 15               | 14 | 13                | 12                | 11               | 10               | 9                | 8      |
| io_config_wd0_dq |    |                   |                   |                  | io_config_sr_clk |                  |        |
| R/W-0h           |    |                   |                   |                  | R/W-0h           |                  |        |
| 7                | 6  | 5                 | 4                 | 3                | 2                | 1                | 0      |
| io_config_i_clk  |    | io_config_sr      |                   |                  | io_config_i      |                  |        |
| R/W-0h           |    | R/W-0h            |                   |                  | R/W-0h           |                  |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-99. ddr\_data0\_iocrtl Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 31-30 | Reserved          | R    | 0h    |  |
| 29    | io_config_wd1_dqs | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQS.<br/>         Used with io_config_wd0_dqs to define pullup/pulldown according to the following:<br/>         WD1: WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>  |
| 28    | io_config_wd1_dm  | R/W  | 0h    | <p>Input that selects pullup or pulldown for DM.<br/>         Used with io_config_wd0_dm to define pullup/pulldown according to the following:<br/>         WD1: WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>  |
| 27-20 | io_config_wd1_dq  | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQ.<br/>         There are 2 bits per IO: io_config_wd1_dq and io_config_wd0_dq.<br/>         For example:<br/>         macro pin 0: WD1 is bit 20 WD0 is bit 10<br/>         macro pin 1: WD1 is bit 21, WD0 is bit 11<br/>         ...<br/>         macro pin 7: WD1 is bit 27, WD0 is bit 17<br/>         See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/>         WD1: WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p> |
| 19    | io_config_wd0_dqs | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQS.<br/>         Used with io_config_wd1_dqs to define pullup/pulldown according to the following:<br/>         WD1: WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>  |

**Table 9-99. ddr\_data0\_ioctrl Register Field Descriptions (continued)**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 18    | io_config_wd0_dm | R/W  | 0h    | <p>Input that selects pullup or pulldown for DM.</p> <p>Used with io_config_wd1_dm to define pullup/pulldown according to the following:</p> <p>WD1:WD0</p> <ul style="list-style-type: none"> <li>00: Pullup/Pulldown disabled</li> <li>01: Weak pullup enabled</li> <li>10: Weak pulldown enabled</li> <li>11: Weak keeper enabled</li> </ul>  |
| 17-10 | io_config_wd0_dq | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQ.</p> <p>There are 2 bits per IO: io_config_wd1_dq and io_config_wd0_dq.</p> <p>For example:</p> <ul style="list-style-type: none"> <li>macro pin 0: WD1 is bit 20, WD0 is bit 10</li> <li>macro pin 1: WD1 is bit 21, WD0 is bit 11</li> <li>...</li> <li>macro pin 7: WD1 is bit 27, WD0 is bit 17</li> </ul> <p>See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.</p> <p>WD1:WD0</p> <ul style="list-style-type: none"> <li>00: Pullup/Pulldown disabled</li> <li>01: Weak pullup enabled</li> <li>10: Weak pulldown enabled</li> <li>11: Weak keeper enabled</li> </ul> |
| 9-8   | io_config_sr_clk | R/W  | 0h    | <p>2 bit to program clock IO Pads (DDR_DQS/DDR_DQSn) output slew rate.</p> <p>These connect as SR1, SR0 of the corresponding IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 7-5   | io_config_i_clk  | R/W  | 0h    | <p>3-bit configuration input to program clock IO pads (DDR_DQS/DDR_DQSn) output impedance.</p> <p>These connect as I2, I1, I0 of the corresponding buffer.</p> <p>See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>  |
| 4-3   | io_config_sr     | R/W  | 0h    | <p>2 bit to program data IO Pads output slew rate.</p> <p>These connect as SR1, SR0 of the corresponding IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 2-0   | io_config_i      | R/W  | 0h    | <p>3-bit configuration input to program data IO output impedance.</p> <p>These connect as I2, I1, I0 of the corresponding buffer.</p> <p>See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |

### 9.3.90 ddr\_data1\_iocrtl Register (offset = 1444h) [reset = 0h]

ddr\_data1\_iocrtl is shown in Figure 9-93 and described in Table 9-100.

**Figure 9-93. ddr\_data1\_iocrtl Register**

|                  |    |                   |                   |                  |                  |                  |        |
|------------------|----|-------------------|-------------------|------------------|------------------|------------------|--------|
| 31               | 30 | 29                | 28                | 27               | 26               | 25               | 24     |
| Reserved         |    | io_config_wd1_dqs | io_config_wd1_dm  | io_config_wd1_dq |                  |                  |        |
| R-0h             |    |                   | R/W-0h            | R/W-0h           |                  |                  | R/W-0h |
| 23               | 22 | 21                | 20                | 19               | 18               | 17               | 16     |
| io_config_wd1_dq |    |                   | io_config_wd0_dqs | io_config_wd0_dm | io_config_wd0_dq | io_config_wd0_dq |        |
| R/W-0h           |    |                   |                   | R/W-0h           | R/W-0h           | R/W-0h           |        |
| 15               | 14 | 13                | 12                | 11               | 10               | 9                | 8      |
| io_config_wd0_dq |    |                   |                   |                  | io_config_sr_clk |                  |        |
| R/W-0h           |    |                   |                   |                  | R/W-0h           |                  |        |
| 7                | 6  | 5                 | 4                 | 3                | 2                | 1                | 0      |
| io_config_i_clk  |    | io_config_sr      |                   |                  | io_config_i      |                  |        |
| R/W-0h           |    | R/W-0h            |                   |                  | R/W-0h           |                  |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 9-100. ddr\_data1\_iocrtl Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 31-30 | Reserved          | R    | 0h    |  |
| 29    | io_config_wd1_dqs | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQS.<br/>         Used with io_config_wd0_dqs to define pullup/pulldown according to the following:<br/>         WD1:WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>   |
| 28    | io_config_wd1_dm  | R/W  | 0h    | <p>Input that selects pullup or pulldown for DM.<br/>         Used with io_config_wd0_dm to define pullup/pulldown according to the following:<br/>         WD1:WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>   |
| 27-20 | io_config_wd1_dq  | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQ.<br/>         There are 2 bits per IO: io_config_wd1_dq and io_config_wd0_dq.<br/>         For example:<br/>         macro pin 0: WD1 is bit 20, WD0 is bit 10<br/>         macro pin 1: WD1 is bit 21, WD0 is bit 11<br/>         ...<br/>         macro pin 7: WD1 is bit 27, WD0 is bit 17<br/>         See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.<br/>         WD1:WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p> |
| 19    | io_config_wd0_dqs | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQS.<br/>         Used with io_config_wd1_dqs to define pullup/pulldown according to the following:<br/>         WD1:WD0<br/>         00: Pullup/Pulldown disabled<br/>         01: Weak pullup enabled<br/>         10: Weak pulldown enabled<br/>         11: Weak keeper enabled</p>   |

**Table 9-100. ddr\_data1\_ioctrl Register Field Descriptions (continued)**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 18    | io_config_wd0_dm | R/W  | 0h    | <p>Input that selects pullup or pulldown for DM.</p> <p>Used with io_config_wd1_dm to define pullup/pulldown according to the following:</p> <p>WD1:WD0</p> <p>00: Pullup/Pulldown disabled</p> <p>01: Weak pullup enabled</p> <p>10: Weak pulldown enabled</p> <p>11: Weak keeper enabled</p>   |
| 17-10 | io_config_wd0_dq | R/W  | 0h    | <p>Input that selects pullup or pulldown for DQ.</p> <p>There are 2 bits per IO: io_config_wd1_dq and io_config_wd0_dq.</p> <p>For example:</p> <p>macro pin 0: WD1 is bit 20, WD0 is bit 10</p> <p>macro pin 1: WD1 is bit 21, WD0 is bit 11</p> <p>...</p> <p>macro pin 7: WD1 is bit 27, WD0 is bit 17</p> <p>See the DDR PHY to IO Pin Mapping table in the Control Module Functional Description section for a mapping of macro bits to I/Os.</p> <p>WD1:WD0</p> <p>00: Pullup/Pulldown disabled</p> <p>01: Weak pullup enabled</p> <p>10: Weak pulldown enabled</p> <p>11: Weak keeper enabled</p> |
| 9-8   | io_config_sr_clk | R/W  | 0h    | <p>2 bit to program clock IO Pads (DDR_DQS/DDR_DQSn) output slew rate.</p> <p>These connect as SR1, SR0 of the corresponding IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 7-5   | io_config_i_clk  | R/W  | 0h    | <p>3-bit configuration input to program clock IO pads (DDR_DQS/DDR_DQSn) output impedance.</p> <p>These connect as I2, I1, I0 of the corresponding buffer.</p> <p>See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>  |
| 4-3   | io_config_sr     | R/W  | 0h    | <p>2 bit to program data IO Pads output slew rate.</p> <p>These connect as SR1, SR0 of the corresponding IO buffer.</p> <p>See the DDR Slew Rate Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |
| 2-0   | io_config_i      | R/W  | 0h    | <p>3-bit configuration input to program data IO output impedance.</p> <p>These connect as I2, I1, I0 of the corresponding buffer.</p> <p>See the DDR Impedance Control Settings table in the Control Module Functional Description section for a definition of these bits.</p>   |

**Interconnects**

This chapter describes the interconnects of the device.

| Topic                          | Page       |
|--------------------------------|------------|
| <b>10.1 Introduction .....</b> | <b>858</b> |

## 10.1 Introduction

The system interconnect is based on a 2-level hierarchical architecture (L3, L4) driven by system performance. The L4 interconnect is based on a fully native OCP infrastructure, directly complying with the OCPIP2.2 reference standard.

### 10.1.1 Terminology

The following is a brief explanation of some terms used in this document:

**Initiator:** Module able to initiate read and write requests to the chip interconnect (typically: processors, DMA, etc.).

**Target:** Unlike an initiator, a target module cannot generate read/write requests to the chip interconnect, but it can respond to these requests. However, it may generate interrupts or a DMA request to the system (typically: peripherals, memory controllers). **Note:** A module can have several separate ports; therefore, a module can be an initiator and a target.

**Agent:** Each connection of one module to one interconnect is done using an agent, which is an adaptation (sometimes configurable) between the module and the interconnect. A target module is connected by a target agent (TA), and an initiator module is connected by an initiator agent (IA).

**Interconnect:** The decoding, routing, and arbitration logic that enable the connection between multiple initiator modules and multiple target modules connected on it.

**Register Target (RT):** Special TA used to access the interconnect internal configuration registers.

**Data-flow Signal:** Any signal that is part of a clearly identified transfer or data flow (typically: command, address, byte enables, etc.). Signal behavior is defined by the protocol semantics.

**Sideband Signal:** Any signal whose behavior is not associated to a precise transaction or data flow.

**Command Slot:** A command slot is a subset of the command list. It is the memory buffer for a single command. A total of 32 command slots exist.

**Out-of-band Error:** Any signal whose behavior is associated to a device error-reporting scheme, as opposed to in-band errors. Note: Interrupt requests and DMA requests are not routed by the interconnect in the device.

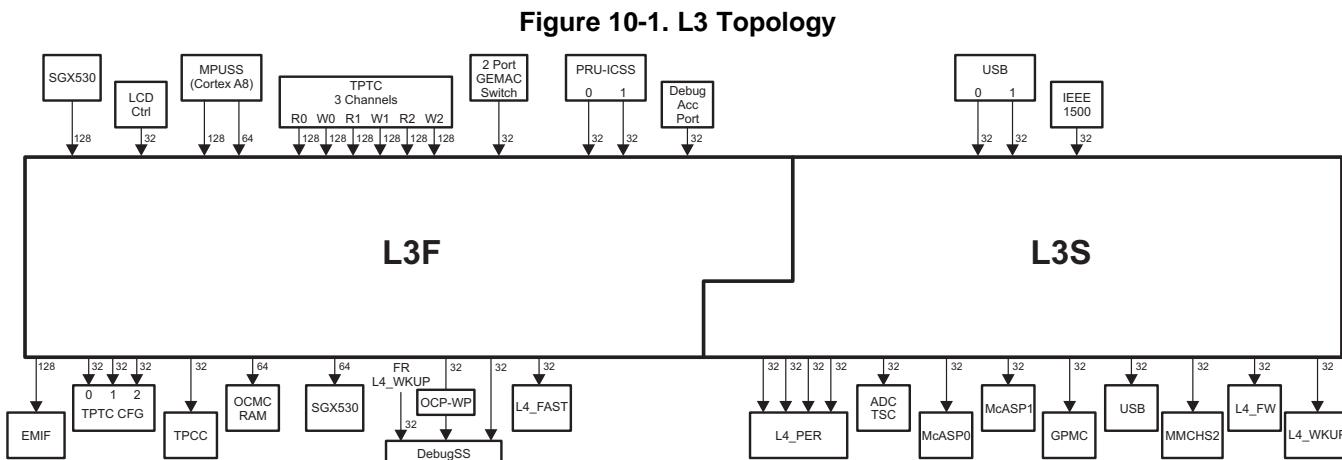
**ConnID:** Any transaction in the system interconnect is tagged by an in-band qualifier ConnID, which uniquely identifies the initiator at a given interconnect point. A ConnID is transmitted in band with the request and is used for error-logging mechanism.

### 10.1.2 L3 Interconnect

The L3 high-performance interconnect is based on a Network-On-Chip (NoC) interconnect infrastructure. The NoC uses an internal packet-based protocol for forward (read command, write command with data payload) and backward (read response with data payload, write response) transactions. All exposed interfaces of this NoC interconnect, both for Targets and Initiators; comply with the OCPIP2.2 reference standard.

#### 10.1.2.1 L3 Topology

The L3 topology is driven by performance requirements, bus types, and clocking structure. The main L3 paths are shown in [Figure 10-1](#). Arrows indicate the master/slave relationship not data flow. L3 is partitioned into two separate clock domains: L3F corresponds to L3 Fast clock domain and L3S corresponds to L3 Slow clock domain.



### 10.1.2.2 L3 Port Mapping

Each initiator and target core is connected to the L3 interconnect through a Network Interface Unit (NIU). The NIUs act as entry and exit points to the L3 Network on Chip – converting between the IP's OCP protocol and the NoC's internal protocol, and also include various programming registers. All ports are single threaded with tags used to enable pipelined transactions. The interconnect includes:

#### Initiator Ports:

- L3F
  - Cortex A8 MPUSS 128-bit initiator port0 and 64-bit initiator port1
  - SGX530 128-bit initiator port
  - 3 TPTC 128-bit read initiator ports
  - 3 TPTC 128-bit write initiator ports
  - LCD 32-bit initiator port
  - 2 PRU-ICSS1 32-bit initiator ports
  - 2 port Gigabit Ethernet Switch (2PGSW) 32-bit initiator port
  - Debug Subsystem 32-bit initiator port
- L3S
  - USB 32-bit CPPI DMA initiator port
  - USB 32-bit Queue Manager initiator port
  - P1500 32-bit initiator port

#### Target Ports:

- L3F
  - EMIF 128-bit target port
  - 3 TPTC CFG 32-bit target ports
  - TPCC CFG 32-bit target port
  - OCM RAM0 64-bit target port
  - DebugSS 32-bit target port
  - SGX530 64-bit target port
  - L4\_FAST 32-bit target port
- L3S
  - 4 L4\_PER peripheral 32-bit target ports
  - GPMC 32-bit target port
  - McASP0 32-bit target port

- McASP1 32-bit target port
- ADC\_TSC 32-bit target port
- USB 32-bit target port
- MMHCS2 32-bit target port
- L4\_WKUP wakeup 32-bit target port

### 10.1.2.3 Interconnect Requirements

The required L3 connections between bus masters and slave ports are shown in [Table 10-1](#). The L3 interconnect will return an address-hole error if any initiator attempts to access a target to which it has no connection.

**Table 10-1. L3 Master — Slave Connectivity**

| Master ID | Masters            | Slaves           |                  |                  |             |                  |   |                  |             |                                 |  |  |  |                  |                             |                            |                            |                            |                  |                                 |  |   |   |   |  |
|-----------|--------------------|------------------|------------------|------------------|-------------|------------------|---|------------------|-------------|---------------------------------|--|--|--|------------------|-----------------------------|----------------------------|----------------------------|----------------------------|------------------|---------------------------------|--|---|---|---|--|
|           |                    | E<br>M<br>I<br>F | O<br>C<br>M<br>C | T<br>P<br>T<br>C | T<br>P<br>C | S<br>G<br>X<br>5 | E<br>x<br>p<br>a<br>n<br>s<br>o<br>n<br>s<br>i<br>o<br>n<br>t | A<br>E<br>S<br>0 | S<br>H<br>A | L<br>4<br>F<br>a<br>s<br>t<br>0 | L<br>4<br>P<br>E<br>R<br>P<br>o<br>r<br>t<br>1 | L<br>4<br>P<br>E<br>R<br>P<br>o<br>r<br>t<br>2 | L<br>4<br>P<br>E<br>R<br>P<br>o<br>r<br>t<br>3 | G<br>P<br>M<br>C | A<br>D<br>C/<br>T<br>S<br>C | U<br>S<br>B<br>C<br>F<br>G | M<br>M<br>C<br>H<br>S<br>2 | L<br>4<br>W<br>K<br>U<br>P | L<br>4<br>F<br>W | D<br>e<br>b<br>u<br>g<br>S<br>S | N<br>O<br>C<br>R<br>e<br>g<br>i<br>s<br>t<br>e<br>r<br>s |   |   |   |  |
| 0x00      | MPUSS M1 (128-bit) | R                |                  |                  |             |                  |   |                  |             |                                 |  |  |  |                  |                             |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x00      | MPUSS M2 (64-bit)  | R                | R                | R                | R           | R                | R   | R                | R           | R                               | R  | R  | R  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x18      | TPTC0 RD           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x19      | TPTC0 WR           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R | R |  |
| 0x1A      | TPTC1 RD           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x1B      | TPTC1 WR           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x1C      | TPTC2 RD           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x1D      | TPTC2 WR           | R                | R                |                  | R           |                  | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R |   |  |
| 0x24      | LCD Controller     | R                | R                |                  |             |                  |   |                  |             |                                 |  |  |  |                  | R                           |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x0E      | PRU-ICSS (PRU0)    | R                | R                | R                | R           |                  | R   |                  |             | R                               |  | R  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R | R |  |
| 0x0F      | PRU-ICSS (PRU1)    | R                | R                | R                | R           |                  | R   |                  |             | R                               |  | R  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R | R |  |
| 0x30      | GEMAC              | R                | R                |                  |             |                  |   |                  |             |                                 |  |  |  |                  | R                           |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x20      | SGX530             | R                | R                |                  |             |                  |   |                  |             |                                 |  |  |  |                  | R                           |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x34      | USB0 DMA           | R                | R                |                  |             |                  |   |                  |             |                                 |  |  |  |                  |                             |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x35      | USB1 Queue Mgr     | R                | R                |                  |             |                  |   |                  |             |                                 |  |  |  |                  | R                           |                            |                            |                            |                  |                                 |  |   |   |   |  |
| 0x04      | EMU (DAP)          | R                | R                | R                | R           | R                | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R | R |  |
| 0x05      | IEEE1500           | R                | R                | R                | R           | R                | R   | R                | R           | R                               |  |  |  | R                | R                           | R                          | R                          | R                          | R                | R                               | R  | R | R | R |  |

### 10.1.2.4 ConnID Assignment

Each L3 initiator includes a unique 6-bit master connection identifier (MConnID) that is used to identify the source of a transfer request. Since AM335x contains more than 16 unique masters some masters will appear identical to target firewalls.

**Table 10-2. MConnID Assignment**

| Initiator          | 6-bit MConnID (Debug) | Instrumentation | Comment               |
|--------------------|-----------------------|-----------------|-----------------------|
| MPUSS M1 (128-bit) | 0x00                  | 0               | Connects only to EMIF |
| MPUSS M2 (64-bit)  | 0x01                  | SW              |                       |

**Table 10-2. MConnID Assignment (continued)**

| Initiator            | 6-bit MConnID (Debug) | Instrumentation | Comment                      |
|----------------------|-----------------------|-----------------|------------------------------|
| DAP                  | 0x04                  | SW              |                              |
| P1500                | 0x05                  | SW              |                              |
| PRU-ICSS (PRU0)      | 0x0E                  | SW              |                              |
| PRU-ICSS (PRU1)      | 0x0F                  | SW              |                              |
| Wakeup M3            | 0x14                  | SW              | Connects only to L4_WKUP     |
| TPTC0 Read           | 0x18                  | 0               |                              |
| TPTC0 Write          | 0x19                  | SW              | One WR port for data logging |
| TPTC1 Read           | 0x1A                  | 0               |                              |
| TPTC1 Write          | 0x1B                  | 0               |                              |
| TPTC2 Read           | 0x1C                  | 0               |                              |
| TPTC2 Write          | 0x1D                  | 0               |                              |
| SGX530               | 0x20                  | 0               |                              |
| OCP WP Traffic Probe | 0x20 <sup>(1)</sup>   | HW              | Direct connect to DebugSS    |
| OCP WP DMA Profiling | 0x21 <sup>(1)</sup>   | HW              | Direct connect to DebugSS    |
| OCP-WP Event Trace   | 0x22 <sup>(1)</sup>   | HW              | Direct connect to DebugSS    |
| LCD Ctrl             | 0x24                  | 0               |                              |
| GEMAC                | 0x30                  | 0               |                              |
| USB DMA              | 0x34                  | 0               |                              |
| USB QMGR             | 0x35                  | 0               |                              |
| Stat Collector 0     | 0x3C                  | HW              |                              |
| Stat Collector 1     | 0x3D                  | HW              |                              |
| Stat Collector 2     | 0x3E                  | HW              |                              |
| Stat Collector 3     | 0x3F                  | HW              |                              |

<sup>(1)</sup> These MConnIDs are generated within the OCP-WP module based on the H0, H1, and H2 configuration parameters.

---

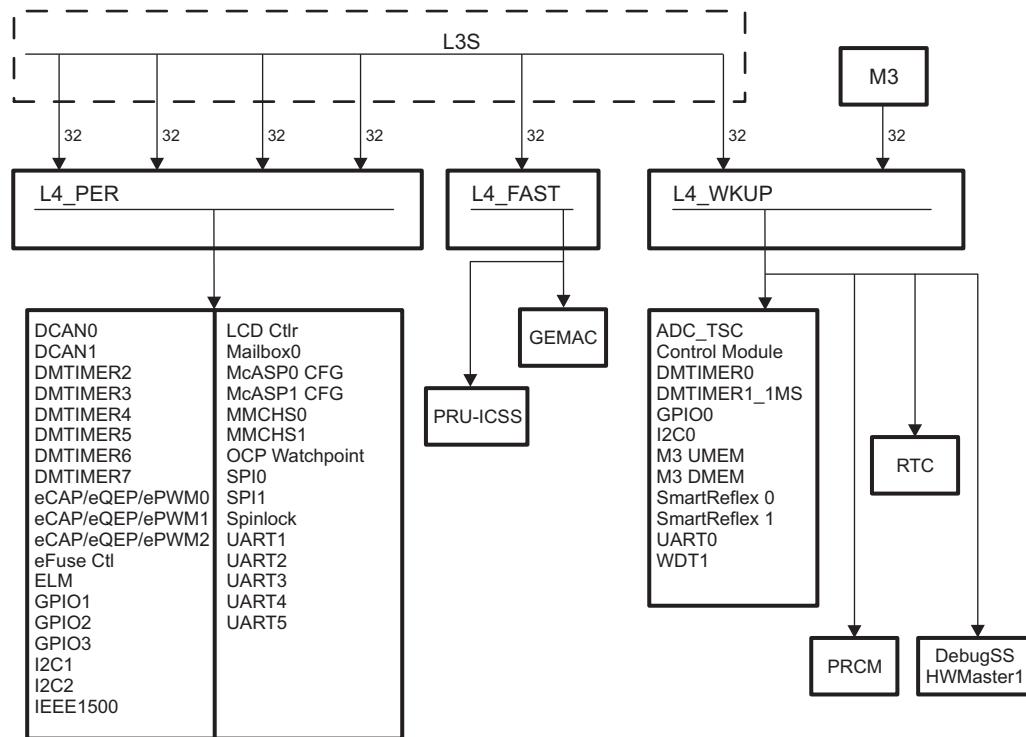
**NOTE:** Instrumentation refers to debug type. SW instrumentation means that the master can write data to be logged to the STM (similar to a printf()). HW indicates debug data captured automatically by hardware. A '0' entry indicates no debug capability.

---

### 10.1.3 L4 Interconnect

The L4 interconnect is a non-blocking peripheral interconnect that provides low latency access to a large number of low bandwidth, physically dispersed target cores. The L4 can handle incoming traffic from up to four initiators and can distribute those communication requests to and collect related responses from up to 63 targets.

AM335x provides three interfaces with L3 interconnect for High Speed Peripheral, Standard Peripheral, and Wakeup Peripherals. . [Figure 10-2](#) shows the L4 bus architecture and memory-mapped peripherals.

**Figure 10-2. L4 Topology**


## ***Enhanced Direct Memory Access (EDMA)***

This chapter describes the EDMA of the device.

| Topic                             | Page |
|-----------------------------------|------|
| 11.1 Introduction .....           | 864  |
| 11.2 Integration .....            | 867  |
| 11.3 Functional Description ..... | 870  |
| 11.4 EDMA3 Registers .....        | 933  |
| 11.5 Appendix A .....             | 1011 |

## 11.1 Introduction

The enhanced direct memory access (EDMA3) controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the device.

Typical usage includes, but is not limited to the following:

- Servicing software-driven paging transfers (e.g., transfers from external memory, such as DDR2 to internal device memory).
- Servicing event-driven peripherals, such as a serial port.
- Performing sorting or sub-frame extraction of various data structures.
- Offloading data transfers from the main device CPU(s).

The EDMA3 controller consists of two principal blocks:

- EDMA3 channel controller (EDMA3CC).
- EDMA3 transfer controller(s) (EDMA3TC).

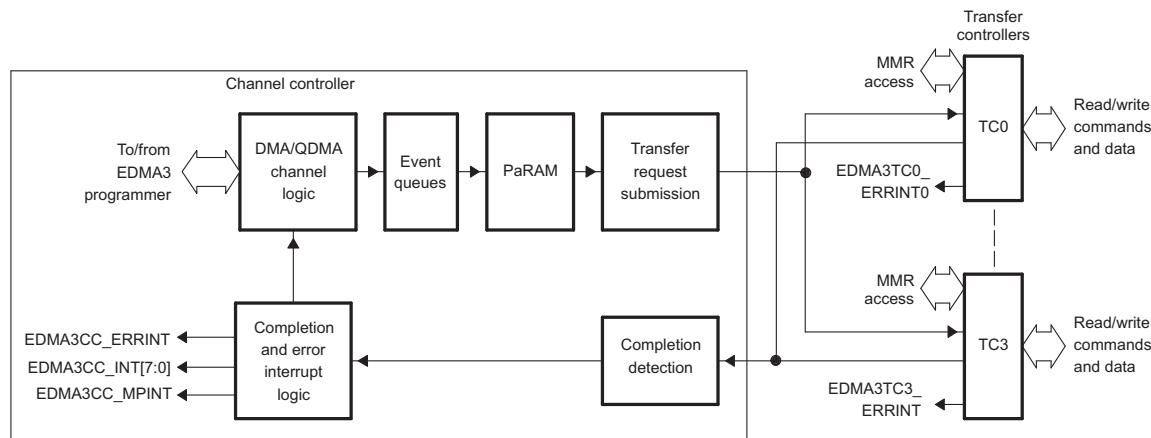
The EDMA3 channel controller serves as the user interface for the EDMA3 controller. The EDMA3CC includes parameter RAM (PaRAM), channel control registers, and interrupt control registers. The EDMA3CC serves to prioritize incoming software requests or events from peripherals and submits transfer requests (TRs) to the transfer controller.

The EDMA3 transfer controllers are slaves to the EDMA3 channel controller that is responsible for data movement. The transfer controller issues read/write commands to the source and destination addresses that are programmed for a given transfer. The operation is transparent to user.

### 11.1.1 EDMA3 Controller Block Diagram

Figure 11-1 shows a block diagram for the EDMA3 controller.

**Figure 11-1. EDMA3 Controller Block Diagram**



### 11.1.2 Third-Party Channel Controller (TPCC) Overview

#### 11.1.2.1 TPCC Features

The general features of the TPCC module are:

- Up to 64 DMA Channels
  - Channels triggered by:
    - Event Synchronization
    - Manual Synchronization (CPU write to 'Event Set Register')
    - Chain Synchronization (completion of one transfer chains to the next)
  - Parameterizable support for programmable DMA Channel to PaRAM mapping

- Up to 8 QDMA Channels
  - QDMA Channels are triggered automatically upon writing to PaRAM
  - Support for programmable QDMA Channel to PaRAM mapping
- Up to 64 Event Inputs
- Up to 8 Interrupt outputs for multi-core support
- Up to 256 PaRAM entries
  - Each PaRAM entry can be used as DMA Entry (up to 64), QDMA Entry (up to 8), or Link Entry (remaining)
- 8 Priority Levels for mapping CC/TC priority relative to priority of other masters in the system.
- Up to 8 Event Queues
- 16 Event Entries per Event Queue
- Supports three-transfer dimensions
  - A-synchronized transfers—one dimension serviced per event
  - AB-synchronized transfers—two dimensions serviced per event
  - Independent Indexes on Source and Destination
  - Does not support direct submission of 3D transfer to TC
  - Chaining feature allows 3D transfer based on single event
- Increment and FIFO transfer addressing modes (TC feature)
- Linking mechanism allows automatic PaRAM Entry update
- Transfer Completion Signaling between TC and CC for Chaining and Interrupt generation.
- Programmable assignment of Priority to TC channel.
- Proxied Memory Protection for TR submission
- Parameterizable support for Active Memory Protection for accesses to PaRAM and registers.
- Queue Watermarking
- Missed Event Detection
- Error and status recording to facilitate debug
- Single Clock domain for all interfaces
- Parameterizable number of Write Completion interfaces (up to 8) (set to number of TC Channels)
- AET Event generation

### 11.1.2.2 Unsupported TPCC Features

This device does not support AET event generation because output is not connected.

### 11.1.3 Third-Party Transfer Controller (TPTC) Overview

#### 11.1.3.1 TPTC Features

The TPTC module includes the following features:

- Up to eight independent channels
- External event control use model (TPCC)
- Read and Write Master ports per Channel 64- or 128-bit configuration.
- Parameterizable FIFO size
- Up to four in-flight Transfer Requests
- Proxied Memory protection for data transfers
- Programmable Priority levels (up to 8)
- Background programmation capability
- Supports 2-dimensional transfers with independent indexes on Source and Destination.

- Support for increment or FIFO-mode transfers
- Interrupt and error support
- Single clock domain for all interfaces

#### 11.1.3.2 Unsupported TPTC Features

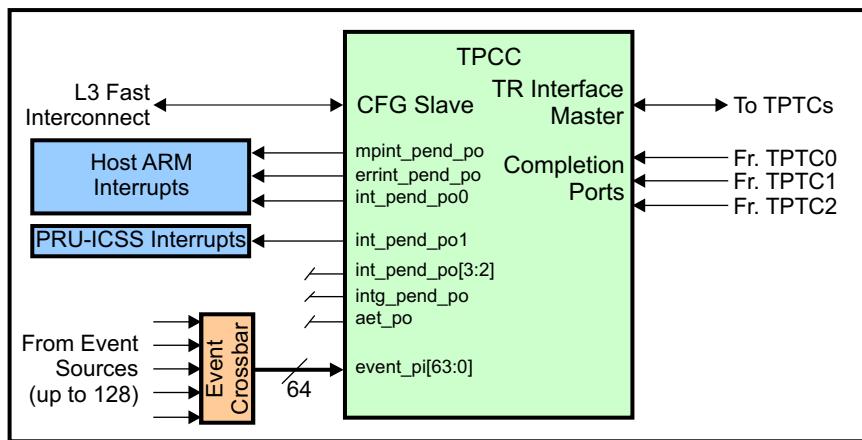
There are no unsupported TPTC features on this device.

## 11.2 Integration

### 11.2.1 Third-Party Channel Controller (TPCC) Integration

This device uses the TPCC peripheral to provide control over its third-party transfer channels (TPTCs). Figure 11-2 shows the integration of the TPCC module.

**Figure 11-2. TPCC Integration**



#### 11.2.1.1 TPCC Connectivity Attributes

The general connectivity attributes of the TPCC are summarized in Table 11-1.

**Table 11-1. TPCC Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power domain        | Peripheral Domain   |
| Clock domain        | PD_PER_L3_GCLK  |
| Reset signals       | PER_DOM_RST_N   |
| Idle/Wakeup signals | Smart Idle  |
| Interrupt request   | 4 Regional Completion Interrupts:<br>int_pend_po0 (EDMACOMPINT) – to MPU Subsystem<br>int_pend_po1 (tpcc_int_pend_po1) – to PRU-ICSS<br>Int_pend_po[3:2] - unused<br>Error Interrupt:<br>errint_po (EDMAERRINT) – to MPU Subsystem<br>Memory Protection Error Interrupt:<br>mpint_po (EDMAMPERR) – to MPU Subsystem |
| DMA request         | none  |
| Physical address    | L3 Fast slave port  |

#### 11.2.1.2 TPCC Clock and Reset Management

The TPCC operates from a single clock and runs at the L3\_Fast clock rate.

**Table 11-2. TPCC Clock Signals**

| Clock Signal                                | Max Freq | Reference / Source | Comments                    |
|---|----------|--------------------|-----------------------------|
| tpcc_clk_pi<br>Interface / Functional clock | 200 MHz  | CORE_CLKOUTM4      | pd_per_l3_gclk<br>From PRCM |

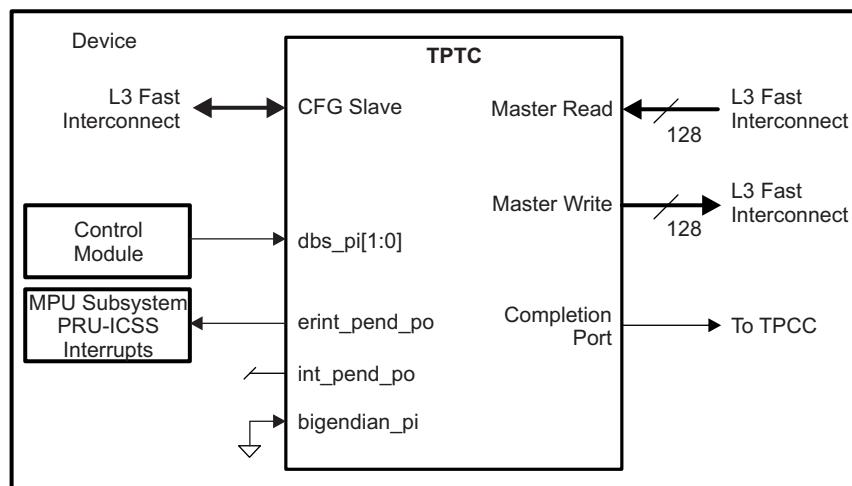
### 11.2.1.3 TPCC Pin List

The TPCC module does not include any external interface pins.

### 11.2.2 Third-Party Transfer Controller (TPTC) Integration

This device uses the three TPTC peripherals (TC0–TC2; TC3 is not supported) to perform EDMA transfers between slave peripherals. The submission of transfer requests to the TPTCs is controlled by the TPCC. [Figure 11-3](#) shows the integration of the TPTC modules

**Figure 11-3. TPTC Integration**



#### 11.2.2.1 TPTC Connectivity Attributes

The general connectivity attributes for the TPTCs are shown in [Table 11-3](#).

**Table 11-3. TPTC Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power domain        | Peripheral Domain  |
| Clock domain        | PD_PER_L3_GCLK   |
| Reset signals       | PER_DOM_RST_N  |
| Idle/Wakeup signals | Standby (2 ports)<br>Smart Idle  |
| Interrupt request   | Error interrupt per instance<br>erint_pend_po (TCERRINTx) – to MPU Subsystem and PRU-ICSS (tptc_erint_pend_po, TPTC0 only) |
| DMA request         | none   |
| Physical address    | L3 Fast slave port   |

#### 11.2.2.2 TPTC Clock and Reset Management

The TPTC operates from a single clock and runs at the L3\_Fast clock rate.

**Table 11-4. TPTC Clock Signals**

| Clock Signal                                | Max Freq | Reference / Source | Comments                    |
|---|----------|--------------------|-----------------------------|
| tptc_clk_pi<br>Interface / Functional clock | 200 MHz  | CORE_CLKOUTM4      | pd_per_l3_gclk<br>From PRCM |

### 11.2.2.3 TPTC Pin List

The TPTC module does not include any external interface pins.

## 11.3 Functional Description

This chapter discusses the architecture of the EDMA3 controller.

### 11.3.1 Functional Overview

#### 11.3.1.1 EDMA3 Channel Controller (EDMA3CC)

Figure 11-4 shows a functional block diagram of the EDMA3 channel controller (EDMA3CC).

The main blocks of the EDMA3CC are as follows:

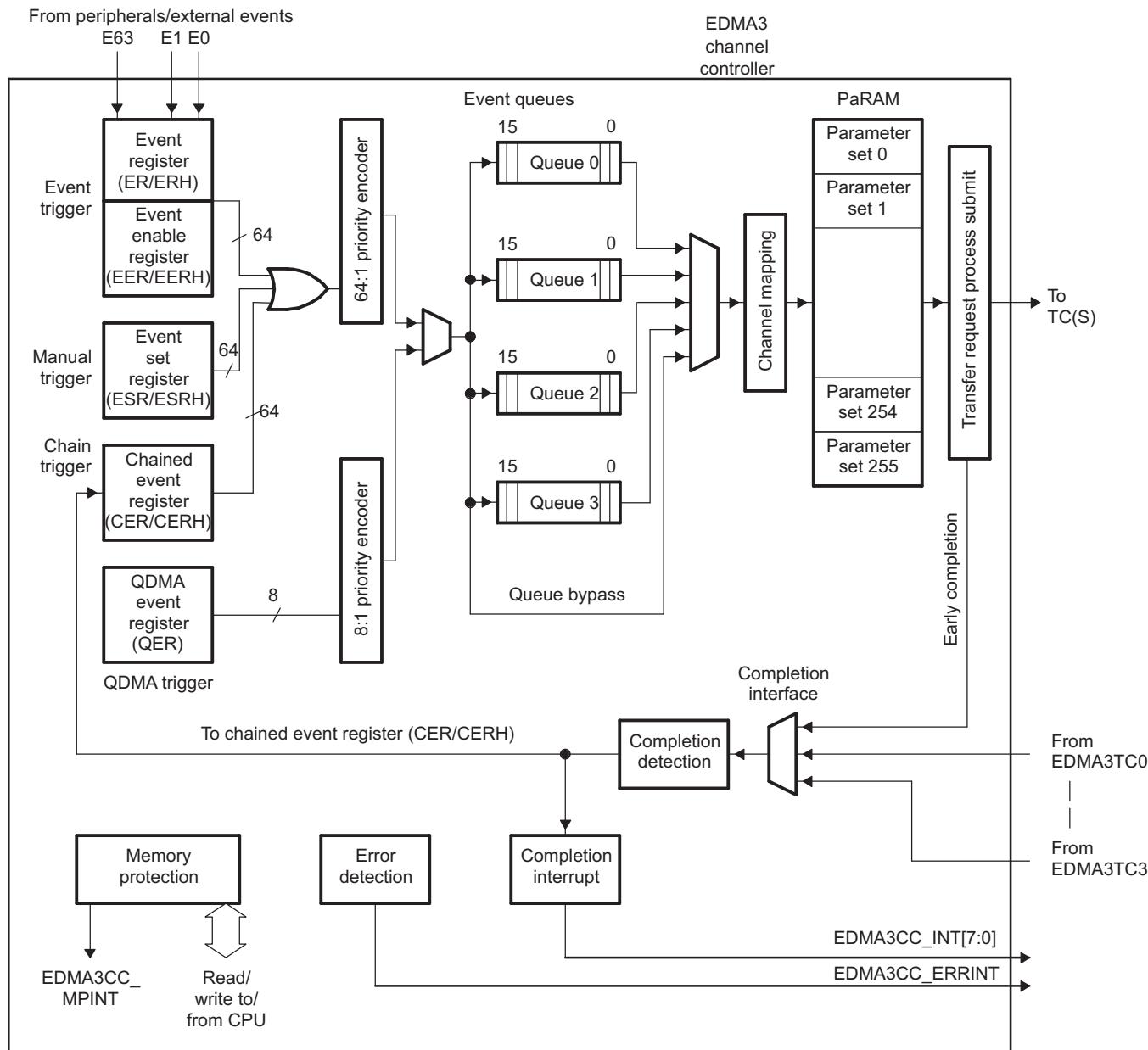
- Parameter RAM (PaRAM): The PaRAM maintains parameter sets for channel and reload parameter sets. You must write the PaRAM with the transfer context for the desired channels and link parameter sets. EDMA3CC processes sets based on a trigger event and submits a transfer request (TR) to the transfer controller.
- EDMA3 event and interrupt processing registers: Allows mapping of events to parameter sets, enable/disable events, enable/disable interrupt conditions, and clearing interrupts.
- Completion detection: The completion detect block detects completion of transfers by the EDMA3TC and/or slave peripherals. You can optionally use completion of transfers to chain trigger new transfers or to assert interrupts.
- Event queues: Event queues form the interface between the event detection logic and the transfer request submission logic.
- Memory protection registers: Memory protection registers define the accesses (privilege level and requestor(s)) that are allowed to access the DMA channel shadow region view(s) and regions of PaRAM.

Other functions include the following:

- Region registers: Region registers allow DMA resources (DMA channels and interrupts) to be assigned to unique regions that different EDMA3 programmers own (for example, ARM).
- Debug registers: Debug registers allow debug visibility by providing registers to read the queue status, controller status, and missed event status.

The EDMA3CC includes two channel types: DMA channels (64 channels) and QDMA channels (8 channels).

Each channel is associated with a given event queue/transfer controller and with a given PaRAM set. The main thing that differentiates a DMA channel from a QDMA channel is the method that the system uses to trigger transfers. See [Section 11.3.4](#).

**Figure 11-4. EDMA3 Channel Controller (EDMA3CC) Block Diagram**


A trigger event is needed to initiate a transfer. A trigger event may be due to an external event, manual write to the event set register, or chained event for DMA channels. QDMA channels auto-trigger when a write to the trigger word that you program occurs on the associated PaRAM set. All such trigger events are logged into appropriate registers upon recognition.

Once a trigger event is recognized, the appropriate event gets queued in the EDMA3CC event queue. The assignment of each DMA/QDMA channel to an event queue is programmable. Each queue is 16 events deep; therefore, you can queue up to 16 events (on a single queue) in the EDMA3CC at a time. Additional pending events that are mapped to a full queue are queued when the event queue space becomes available. See [Section 11.3.11](#).

If events on different channels are detected simultaneously, the events are queued based on a fixed priority arbitration scheme with the DMA channels being higher priority events than the QDMA channels. Among the two groups of channels, the lowest-numbered channel is the highest priority.

Each event in the event queue is processed in FIFO order. When the head of the queue is reached, the PaRAM associated with that channel is read to determine the transfer details. The TR submission logic evaluates the validity of the TR and is responsible for submitting a valid transfer request (TR) to the appropriate EDMA3TC (based on the event queue to the EDMA3TC association, Q0 goes to TC0 , Q1 goes to TC1, Q2 goes to TC2, and Q3 goes to TC3). For more information, refer to [Section 11.3.3](#).

The EDMA3TC receives the request and is responsible for data movement, as specified in the transfer request packet (TRP), other necessary tasks like buffering, and ensuring transfers are carried out in an optimal fashion wherever possible. For more information on EDMA3TC, refer to [Section 11.3.1.2](#).

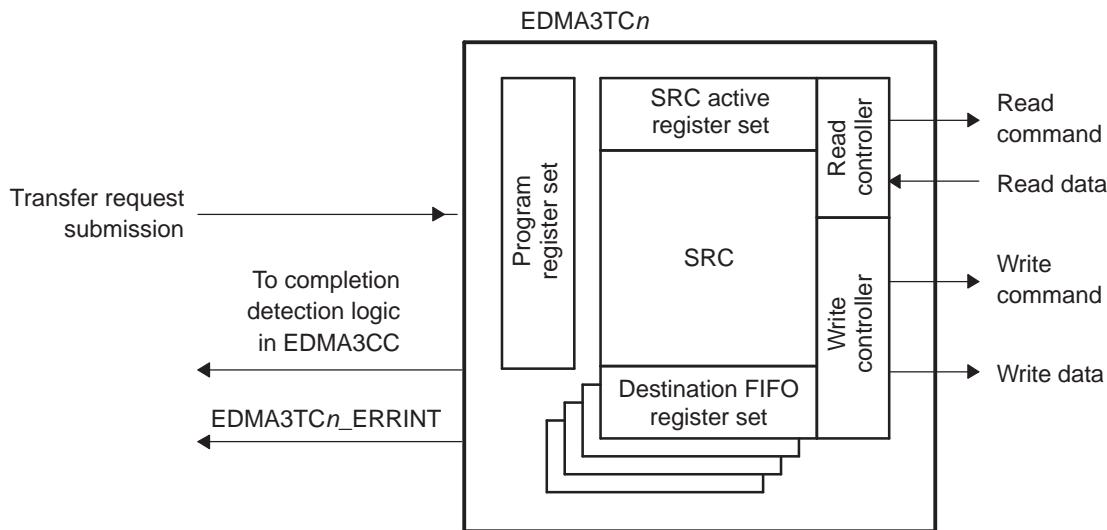
If you have decided to receive an interrupt or to chain to another channel on completion of the current transfer, the EDMA3TC signals completion to the EDMA3CC completion detection logic when the transfer is complete. You can alternately choose to trigger completion when a TR leaves the EDMA3CC boundary, rather than wait for all of the data transfers to complete. Based on the setting of the EDMA3CC interrupt registers, the completion interrupt generation logic is responsible for generating EDMA3CC completion interrupts to the CPU. For more information, refer to [Section 11.3.5](#).

Additionally, the EDMA3CC also has an error detection logic that causes an error interrupt generation on various error conditions (like missed events, exceeding event queue thresholds, etc.). For more information on error interrupts, refer to [Section 11.3.9.4](#).

### 11.3.1.2 EDMA3 Transfer Controller (EDMA3TC)

[Section 11.3.9.4](#) shows a functional block diagram of the EDMA3 transfer controller (EDMA3TC).

**Figure 11-5. EDMA3 Transfer Controller (EDMA3TC) Block Diagram**



The main blocks of the EDMA3TC are:

- DMA program register set: The DMA program register set stores the transfer requests received from the EDMA3 channel controller (EDMA3CC).
- DMA source active register set: The DMA source active register set stores the context for the DMA transfer request currently in progress in the read controller.
- Read controller: The read controller issues read commands to the source address.
- Destination FIFO register set: The destination (DST) FIFO register set stores the context for the DMA transfer request(s) currently in progress in the write controller.
- Write controller: The write controller issues write commands/write data to the destination slave.
- Data FIFO: The data FIFO exists for holding temporary in-flight data.
- Completion interface: The completion interface sends completion codes to the EDMA3CC when a transfer completes, and generates interrupts and chained events (also, see [Section 11.3.1.1](#) for more information on transfer completion reporting).

When the EDMA3TC is idle and receives its first TR, DMA program register set receives the TR, where it transitions to the DMA source active set and the destination FIFO register set immediately. The second TR (if pending from EDMA3CC) is loaded into the DMA program set, ensuring it can start as soon as possible when the active transfer completes. As soon as the current active set is exhausted, the TR is loaded from the DMA program register set into the DMA source active register set as well as to the appropriate entry in the destination FIFO register set.

The read controller issues read commands governed by the rules of command fragmentation and optimization. These are issued only when the data FIFO has space available for the data read. When sufficient data is in the data FIFO, the write controller starts issuing a write command again following the rules for command fragmentation and optimization. For more information on command fragmentation and optimization, refer to [Section 11.3.12.1.1](#).

Depending on the number of entries, the read controller can process up to two or four transfer requests ahead of the destination subject to the amount of free data FIFO.

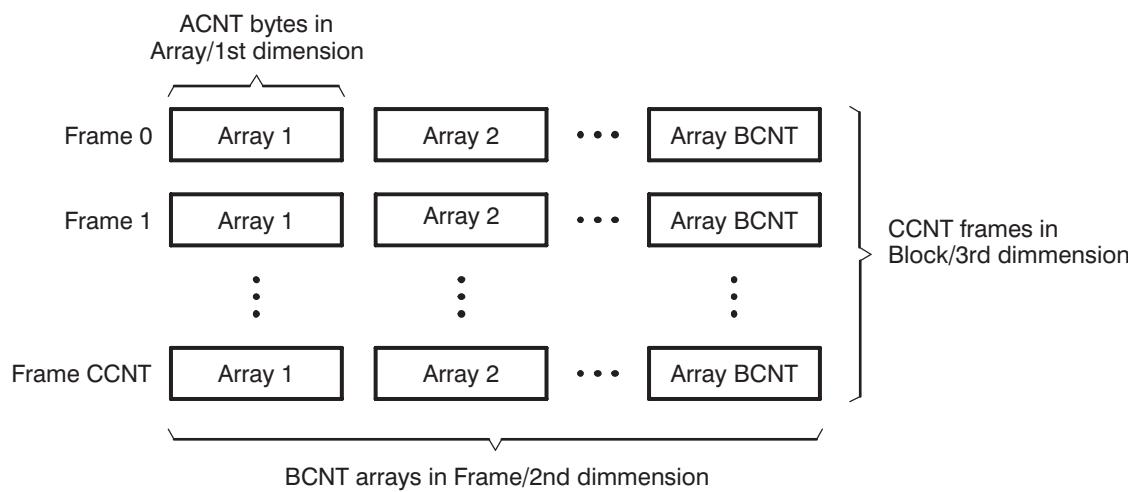
### 11.3.2 Types of EDMA3 Transfers

An EDMA3 transfer is always defined in terms of three dimensions. [Figure 11-6](#) shows the three dimensions used by EDMA3 transfers. These three dimensions are defined as:

- 1st Dimension or Array (A): The 1st dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The 2nd dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the 2nd dimension is separated from each other by an index programmed using SRCBIDX or DSTBIDX.
- 3rd Dimension or Block (C): The 3rd dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the 3rd dimension is separated from the previous by an index programmed using SRCCIDX or DSTCIDX.

Note that the reference point for the index depends on the synchronization type. The amount of data transferred upon receipt of a trigger/synchronization event is controlled by the synchronization types (SYNCDIM bit in OPT). Of the three dimensions, only two synchronization types are supported: A-synchronized transfers and AB-synchronized transfers.

**Figure 11-6. Definition of ACNT, BCNT, and CCNT**



### 11.3.2.1 A-Synchronized Transfers

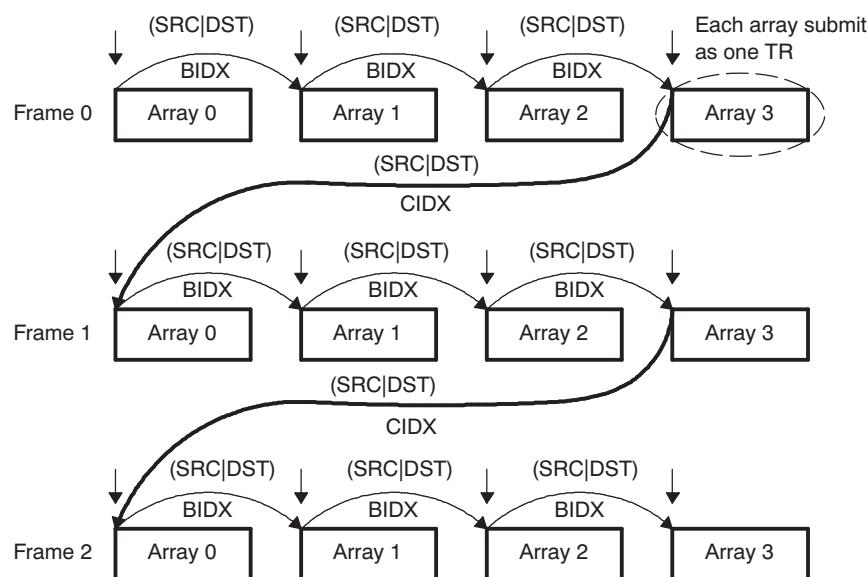
In an A-synchronized transfer, each EDMA3 sync event initiates the transfer of the 1st dimension of ACNT bytes, or one array of ACNT bytes. In other words, each event/TR packet conveys the transfer information for one array only. Thus, BCNT × CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX, as shown in [Figure 11-7](#), where the start address of Array N is equal to the start address of Array N – 1 plus source (SRC) or destination (DST) BIDX.

Frames are always separated by SRCCIDX and DSTCIDX. For A-synchronized transfers, after the frame is exhausted, the address is updated by adding SRCCIDX/DSTCIDX to the beginning address of the last array in the frame. As in [Figure 11-7](#), SRCCIDX/DSTCIDX is the difference between the start of Frame 0 Array 3 to the start of Frame 1 Array 0.

[Figure 11-7](#) shows an A-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of n (ACNT) bytes. In this example, a total of 12 sync events (BCNT × CCNT) exhaust a PaRAM set. See [Section 11.3.3.6](#) for details on parameter set updates.

**Figure 11-7. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**



### 11.3.2.2 AB-Synchronized Transfers

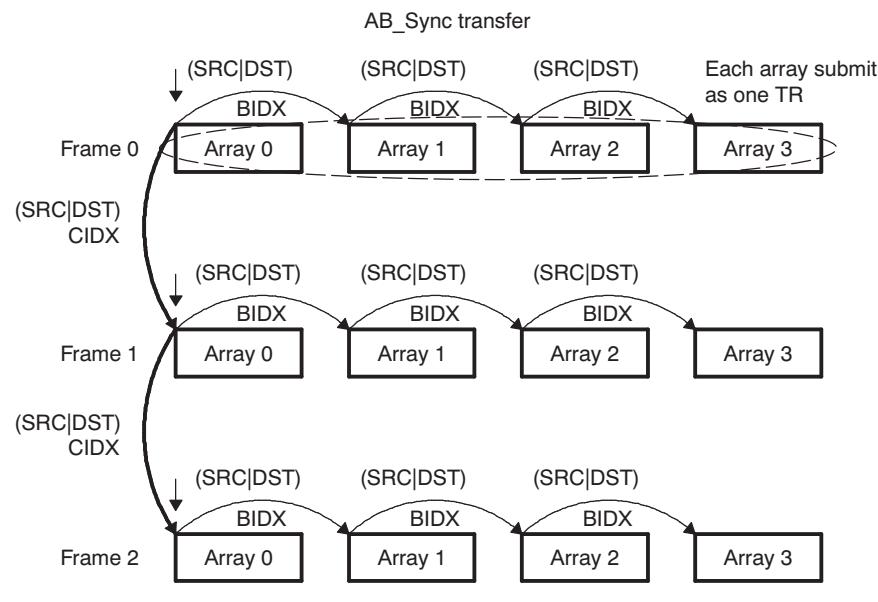
In a AB-synchronized transfer, each EDMA3 sync event initiates the transfer of 2 dimensions or one frame. In other words, each event/TR packet conveys information for one entire frame of BCNT arrays of ACNT bytes. Thus, CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX as shown in [Figure 11-8](#). Frames are always separated by SRCCIDX and DSTCIDX.

Note that for AB-synchronized transfers, after a TR for the frame is submitted, the address update is to add SRCCIDX/DSTCIDX to the beginning address of the beginning array in the frame. This is different from A-synchronized transfers where the address is updated by adding SRCCIDX/DSTCIDX to the start address of the last array in the frame. See [Section 11.3.3.6](#) for details on parameter set updates.

[Figure 11-8](#) shows an AB-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of  $n$  (ACNT) bytes. In this example, a total of 3 sync events (CCNT) exhaust a PaRAM set; that is, a total of 3 transfers of 4 arrays each completes the transfer.

**Figure 11-8. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**



**NOTE:** ABC-synchronized transfers are not directly supported. But can be logically achieved by chaining between multiple AB-synchronized transfers.

### 11.3.3 Parameter RAM (PaRAM)

The EDMA3 controller is a RAM-based architecture. The transfer context (source/destination addresses, count, indexes, etc.) for DMA or QDMA channels is programmed in a parameter RAM table within EDMA3CC, referred to as PaRAM. The PaRAM table is segmented into multiple PaRAM sets. Each PaRAM set includes eight four-byte PaRAM set entries (32-bytes total per PaRAM set), which includes typical DMA transfer parameters such as source address, destination address, transfer counts, indexes, options, etc.

The PaRAM structure supports flexible ping-pong, circular buffering, channel chaining, and auto-reloading (linking).

The contents of the PaRAM include the following:

- 256 PaRAM sets
- 64 channels that are direct mapped and can be used as link or QDMA sets if not used for DMA channels
- 64 channels remain for link or QDMA sets

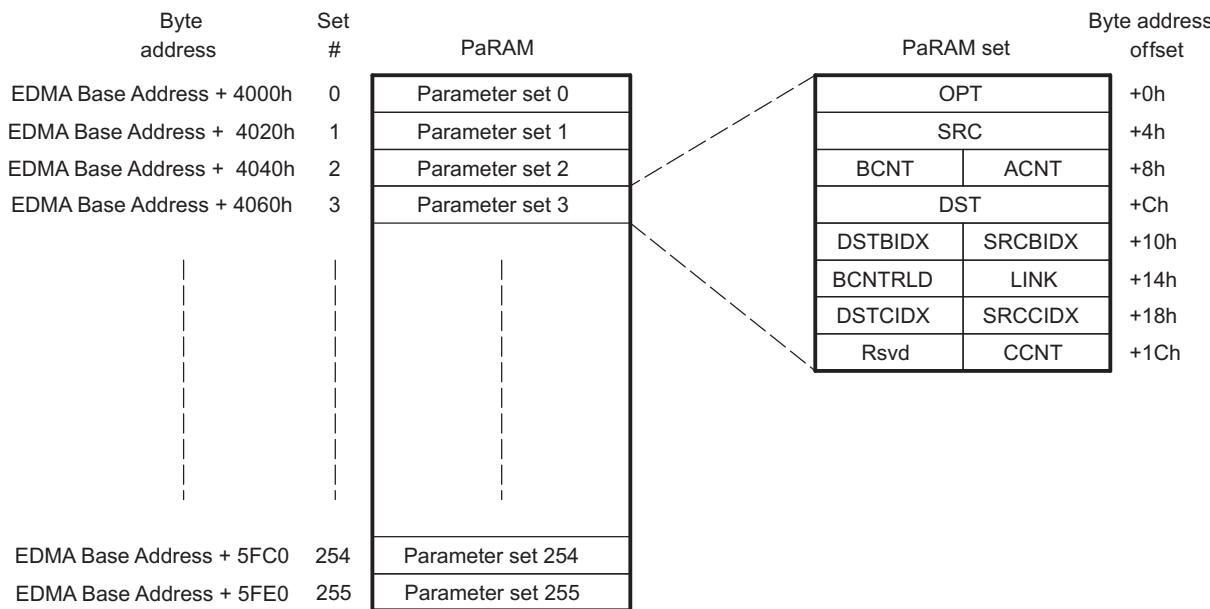
By default, all channels map to PaRAM set to 0. These should be remapped before use. For more information, see [Section 11.4.1.1.4](#) (DCHMAP registers) and [Section 11.4.1.1.5](#) (QCHMAP registers).

**Table 11-5. EDMA3 Parameter RAM Contents**

| PaRAM Set Number | Address  | Parameters    |
|------------------|--|---------------|
| 0                | EDMA Base Address + 4000h to EDMA Base Address + 401Fh | PaRAM set 0   |
| 1                | EDMA Base Address + 4020h to EDMA Base Address + 403Fh | PaRAM set 1   |
| 2                | EDMA Base Address + 4040h to EDMA Base Address + 405Fh | PaRAM set 2   |
| 3                | EDMA Base Address + 4060h to EDMA Base Address + 407Fh | PaRAM set 3   |
| 4                | EDMA Base Address + 4080h to EDMA Base Address + 409Fh | PaRAM set 4   |
| 5                | EDMA Base Address + 40A0h to EDMA Base Address + 40BFh | PaRAM set 5   |
| 6                | EDMA Base Address + 40C0h to EDMA Base Address + 40DFh | PaRAM set 6   |
| 7                | EDMA Base Address + 40E0h to EDMA Base Address + 40FFh | PaRAM set 7   |
| 8                | EDMA Base Address + 4100h to EDMA Base Address + 411Fh | PaRAM set 8   |
| 9                | EDMA Base Address + 4120h to EDMA Base Address + 413Fh | PaRAM set 9   |
| ...              | ...  | ...           |
| 63               | EDMA Base Address + 47E0h to EDMA Base Address + 47FFh | PaRAM set 63  |
| 64               | EDMA Base Address + 4800h to EDMA Base Address + 481Fh | PaRAM set 64  |
| 65               | EDMA Base Address + 4820h to EDMA Base Address + 483Fh | PaRAM set 65  |
| ...              | ...  | ...           |
| 254              | EDMA Base Address + 5FC0h to EDMA Base Address + 5FDKh | PaRAM set 254 |
| 255              | EDMA Base Address + 5FE0h to EDMA Base Address + 5FFFh | PaRAM set 255 |

### 11.3.3.1 PaRAM

Each parameter set of PaRAM is organized into eight 32-bit words or 32 bytes, as shown in [Figure 11-9](#) and described in [Table 11-6](#). Each PaRAM set consists of 16-bit and 32-bit parameters.

**Figure 11-9. PaRAM Set**


**Table 11-6. EDMA3 Channel Parameter Description**

| Offset Address<br>(bytes) | Acronym | Parameter                   | Description  |
|---------------------------|---------|-----------------------------|--|
| 0h                        | OPT     | Channel Options             | Transfer configuration options   |
| 4h                        | SRC     | Channel Source Address      | The byte address from which data is transferred  |
| 8h <sup>(1)</sup>         | ACNT    | Count for 1st Dimension     | Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535.   |
|                           | BCNT    | Count for 2nd Dimension     | Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535.  |
| Ch                        | DST     | Channel Destination Address | The byte address to which data is transferred  |
| 10h <sup>(1)</sup>        | SRCBIDX | Source BCNT Index           | Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from -32 768 and 32 767.  |
|                           | DSTBIDX | Destination BCNT Index      | Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from -32 768 and 32 767.   |
| 14h <sup>(1)</sup>        | LINK    | Link Address                | The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. A value of FFFFh specifies a null link.   |
|                           | BCNTRLD | BCNT Reload                 | The count value used to reload BCNT when BCNT decrements to 0 (TR is submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers.  |
| 18h <sup>(1)</sup>        | SRCCIDX | Source CCNT Index           | Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from -32 768 and 32 767.<br><br>A-synchronized transfers: The byte address offset from the beginning of the last source array in a frame to the beginning of the first source array in the next frame.<br><br>AB-synchronized transfers: The byte address offset from the beginning of the first source array in a frame to the beginning of the first source array in the next frame.                     |
|                           | DSTCIDX | Destination CCNT index      | Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from -32 768 and 32 767.<br><br>A-synchronized transfers: The byte address offset from the beginning of the last destination array in a frame to the beginning of the first destination array in the next frame.<br><br>AB-synchronized transfers: The byte address offset from the beginning of the first destination array in a frame to the beginning of the first destination array in the next frame. |
| 1Ch                       | CCNT    | Count for 3rd Dimension     | Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535.  |
|                           | RSVD    | Reserved                    | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |

<sup>(1)</sup> It is recommended to access the parameter set sets as 32-bit words whenever possible.

### 11.3.3.2 EDMA3 Channel PaRAM Set Entry Fields

#### 11.3.3.2.1 Channel Options Parameter (OPT)

The channel options parameter (OPT) is shown in [Figure 11-10](#) and described in [Table 11-7](#).

**Figure 11-10. Channel Options Parameter (OPT)**

| 31    | 30       | 28    | 27     | 24       | 23     | 22       | 21      | 20     | 19       | 18    | 17    | 16 |
|-------|----------|-------|--------|----------|--------|----------|---------|--------|----------|-------|-------|----|
| PRIV  | Reserved |       | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |       | TCC   |    |
| R-0   | R-0      |       | R-0    | R/W-0    | R/W-0  | R/W-0    | R/W-0   | R/W-0  | R/W-0    |       | R/W-0 |    |
| 15    | 12       | 11    | 10     | 8        | 7      |          |         | 4      | 3        | 2     | 1     | 0  |
| TCC   | TCCMOD E | FWID  |        | Reserved |        |          |         | STATIC | SYNCDIM  | DAM   | SAM   |    |
| R/W-0 | R/W-0    | R/W-0 |        | R/W-0    |        |          |         | R/W-0  | R/W-0    | R/W-0 | R/W-0 |    |

**Table 11-7. Channel Options Parameters (OPT) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31    | PRIV     | 0     | Privilege level (supervisor versus user) for the host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege value when any part of the PaRAM set is written.  |
|       |          | 1     | User level privilege.   |
|       |          | 1     | Supervisor level privilege.   |
| 30-28 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 27-24 | PRIVID   | 0-Fh  | Privilege identification for the external host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege identification value when any part of the PaRAM set is written.  |
| 23    | ITCCHEN  | 0     | Intermediate transfer completion chaining enable.   |
|       |          | 1     | Intermediate transfer complete chaining is disabled.  |
|       |          | 1     | Intermediate transfer complete chaining is enabled.   |
|       |          |       | When enabled, the chained event register (CER/CERH) bit is set on every intermediate chained transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified.   |
| 22    | TCCHEN   | 0     | Transfer complete chaining enable.  |
|       |          | 1     | Transfer complete chaining is disabled.   |
|       |          | 1     | Transfer complete chaining is enabled.  |
|       |          |       | When enabled, the chained event register (CER/CERH) bit is set on final chained transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified.   |
| 21    | ITCINTEN | 0     | Intermediate transfer completion interrupt enable.  |
|       |          | 1     | Intermediate transfer complete interrupt is disabled.   |
|       |          | 1     | Intermediate transfer complete interrupt is enabled.  |
|       |          |       | When enabled, the interrupt pending register (IPR / IPRH) bit is set on every intermediate transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. To generate a completion interrupt to the CPU, the corresponding IER[TCC] / IERH [TCC] bit must be set. |
| 20    | TCINTEN  | 0     | Transfer complete interrupt enable.   |
|       |          | 1     | Transfer complete interrupt is disabled.  |
|       |          | 1     | Transfer complete interrupt is enabled.   |
|       |          |       | When enabled, the interrupt pending register (IPR / IPRH) bit is set on transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. To generate a completion interrupt to the CPU, the corresponding IER[TCC] / IERH [TCC] bit must be set.   |
| 19-18 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |

**Table 11-7. Channel Options Parameters (OPT) Field Descriptions (continued)**

| Bit   | Field    | Value  | Description  |
|-------|----------|--|--|
| 17-12 | TCC      | 0-3Fh  | Transfer complete code. This 6-bit code sets the relevant bit in the chaining enable register (CER [TCC] /CERH [TCC]) for chaining or in the interrupt pending register (IPR [TCC] / IPRH [TCC]) for interrupts.   |
| 11    | TCCMODE  | 0<br>1   | Transfer complete code mode. Indicates the point at which a transfer is considered completed for chaining and interrupt generation.<br>0 Normal completion: A transfer is considered completed after the data has been transferred.<br>1 Early completion: A transfer is considered completed after the EDMA3CC submits a TR to the EDMA3TC. TC may still be transferring data when the interrupt/chain is triggered.  |
| 10-8  | FWID     | 0-7h<br>0<br>1h<br>2h<br>3h<br>4h<br>5h<br>6h-7h | FIFO Width. Applies if either SAM or DAM is set to constant addressing mode.<br>0 FIFO width is 8-bit.<br>1h FIFO width is 16-bit.<br>2h FIFO width is 32-bit.<br>3h FIFO width is 64-bit.<br>4h FIFO width is 128-bit.<br>5h FIFO width is 256-bit.<br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.                        |
| 7-4   | Reserved | 0  | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 3     | STATIC   | 0<br>1   | Static set.<br>0 Set is not static. The PaRAM set is updated or linked after a TR is submitted. A value of 0 should be used for DMA channels and for non-final transfers in a linked list of QDMA transfers.<br>1 Set is static. The PaRAM set is not updated or linked after a TR is submitted. A value of 1 should be used for isolated QDMA transfers or for the final transfer in a linked list of QDMA transfers. |
| 2     | SYNCDIM  | 0<br>1   | Transfer synchronization dimension.<br>0 A-synchronized. Each event triggers the transfer of a single array of ACNT bytes.<br>1 AB-synchronized. Each event triggers the transfer of BCNT arrays of ACNT bytes.  |
| 1     | DAM      | 0<br>1   | Destination address mode.<br>0 Increment (INCR) mode. Destination addressing within an array increments. Destination is not a FIFO.<br>1 Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.   |
| 0     | SAM      | 0<br>1   | Source address mode.<br>0 Increment (INCR) mode. Source addressing within an array increments. Source is not a FIFO.<br>1 Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.   |

### 11.3.3.2.2 Channel Source Address (SRC)

The 32-bit source address parameter specifies the starting byte address of the source. For SAM in increment mode, there are no alignment restrictions imposed by EDMA3. For SAM in constant addressing mode, you must program the source address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See [Section 11.3.12.3](#) for additional details.

### 11.3.3.2.3 Channel Destination Address (DST)

The 32-bit destination address parameter specifies the starting byte address of the destination. For DAM in increment mode, there are no alignment restrictions imposed by EDMA3. For DAM in constant addressing mode, you must program the destination address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See [Section 11.3.12.3](#) for additional details.

#### **11.3.3.2.4 Count for 1st Dimension (ACNT)**

ACNT represents the number of bytes within the 1st dimension of a transfer. ACNT is a 16-bit unsigned value with valid values between 0 and 65 535. Therefore, the maximum number of bytes in an array is 65 535 bytes (64K – 1 bytes). ACNT must be greater than or equal to 1 for a TR to be submitted to EDMA3TC. A transfer with ACNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

See [Section 11.3.3.5](#) and [Section 11.3.5.3](#) for details on dummy/null completion conditions.

#### **11.3.3.2.5 Count for 2nd Dimension (BCNT)**

BCNT is a 16-bit unsigned value that specifies the number of arrays of length ACNT. For normal operation, valid values for BCNT are between 1 and 65 535. Therefore, the maximum number of arrays in a frame is 65 535 (64K – 1 arrays). A transfer with BCNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

See [Section 11.3.3.5](#) and [Section 11.3.5.3](#) for details on dummy/null completion conditions.

#### **11.3.3.2.6 Count for 3rd Dimension (CCNT)**

CCNT is a 16-bit unsigned value that specifies the number of frames in a block. Valid values for CCNT are between 1 and 65 535. Therefore, the maximum number of frames in a block is 65 535 (64K – 1 frames). A transfer with CCNT equal to 0 is considered either a null or dummy transfer. A dummy or null transfer generates a completion code depending on the settings of the completion bit fields in OPT.

A CCNT value of 0 is considered either a null or dummy transfer. See [Section 11.3.3.5](#) and [Section 11.3.5.3](#) for details on dummy/null completion conditions.

#### **11.3.3.2.7 BCNT Reload (BCNTRLD)**

BCNTRLD is a 16-bit unsigned value used to reload the BCNT field once the last array in the 2nd dimension is transferred. This field is only used for A-synchronized transfers. In this case, the EDMA3CC decrements the BCNT value by 1 on each TR submission. When BCNT reaches 0, the EDMA3CC decrements CCNT and uses the BCNTRLD value to reinitialize the BCNT value.

For AB-synchronized transfers, the EDMA3CC submits the BCNT in the TR and the EDMA3TC decrements BCNT appropriately. For AB-synchronized transfers, BCNTRLD is not used.

#### **11.3.3.2.8 Source B Index (SRCBIDX)**

SRCBIDX is a 16-bit signed value (2s complement) used for source address modification between each array in the 2nd dimension. Valid values for SRCBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the source array to the beginning of the next source array. It applies to both A-synchronized and AB-synchronized transfers. Some examples:

- SRCBIDX = 0000h (0): no address offset from the beginning of an array to the beginning of the next array. All arrays are fixed to the same beginning address.
- SRCBIDX = 0003h (+3): the address offset from the beginning of an array to the beginning of the next array in a frame is 3 bytes. For example, if the current array begins at address 1000h, the next array begins at 1003h.
- SRCBIDX = FFFFh (-1): the address offset from the beginning of an array to the beginning of the next array in a frame is –1 byte. For example, if the current array begins at address 5054h, the next array begins at 5053h.

#### **11.3.3.2.9 Destination B Index (DSTBIDX)**

DSTBIDX is a 16-bit signed value (2s complement) used for destination address modification between each array in the 2nd dimension. Valid values for DSTBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the destination array to the beginning of the next destination array within the current frame. It applies to both A-synchronized and AB-synchronized transfers. See SRCBIDX for examples.

### 11.3.3.2.10 Source C Index (SRCCIDX)

SRCCIDX is a 16-bit signed value (2s complement) used for source address modification in the 3rd dimension. Valid values for SRCCIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by SRC address) to the beginning of the first source array in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when SRCCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 11-7), while the current array in an AB-synchronized transfer is the first array in the frame (Figure 11-8).

### 11.3.3.2.11 Destination C Index (DSTCIDX)

DSTCIDX is a 16-bit signed value (2s complement) used for destination address modification in the 3rd dimension. Valid values are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by DST address) to the beginning of the first destination array TR in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when DSTCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 11-7), while the current array in a AB-synchronized transfer is the first array in the frame (Figure 11-8).

### 11.3.3.2.12 Link Address (LINK)

The EDMA3CC provides a mechanism, called linking, to reload the current PaRAM set upon its natural termination (that is, after the count fields are decremented to 0) with a new PaRAM set. The 16-bit parameter LINK specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking.

You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0.

The EDMA3CC ignores the upper 2 bits of the LINK entry, allowing the programmer the flexibility of programming the link address as either an absolute/literal byte address or use the PaRAM-base-relative offset address. Therefore, if you make use of the literal address with a range from 4000h to 7FFFh, it will be treated as a PaRAM-base-relative value of 0000h to 3FFFh.

You should make sure to program the LINK field correctly, so that link update is requested from a PaRAM address that falls in the range of the available PaRAM addresses on the device.

A LINK value of FFFFh is referred to as a NULL link that should cause the EDMA3CC to perform an internal write of 0 to all entries of the current PaRAM set, except for the LINK field that is set to FFFFh. Also, see [Section 11.3.5](#) for details on terminating a transfer.

### 11.3.3.3 Null PaRAM Set

A null PaRAM set is defined as a PaRAM set where all count fields (ACNT, BCNT, and CCNT) are cleared to 0. If a PaRAM set associated with a channel is a NULL set, then when serviced by the EDMA3CC, the bit corresponding to the channel is set in the associated event missed register (EMR, EMRH, or QEMR). This bit remains set in the associated secondary event register (SER, SERH, or QSER). *This implies that any future events on the same channel are ignored by the EDMA3CC and you are required to clear the bit in SER, SERH, or QSER for the channel.* This is considered an error condition, since events are not expected on a channel that is configured as a null transfer. See [Section 11.4.1.6.8](#) and [Section 11.4.1.2.1](#) for more information on the SER and EMR registers, respectively.

### 11.3.3.4 Dummy PaRAM Set

A dummy PaRAM set is defined as a PaRAM set where at least one of the count fields (ACNT, BCNT, or CCNT) is cleared to 0 and at least one of the count fields is nonzero.

If a PaRAM set associated with a channel is a dummy set, then when serviced by the EDMA3CC, it will not set the bit corresponding to the channel (DMA/QDMA) in the event missed register (EMR, EMRH, or QEMR) and the secondary event register (SER, SERH, or QSER) bit gets cleared similar to a normal transfer. Future events on that channel are serviced. A dummy transfer is a legal transfer of 0 bytes. See [Section 11.4.1.6.8](#) and [Section 11.4.1.2.1](#) for more information on the SER and EMR registers, respectively.

### 11.3.3.5 Dummy Versus Null Transfer Comparison

There are some differences in the way the EDMA3CC logic treats a dummy versus a null transfer request. A null transfer request is an error condition, but a dummy transfer is a legal transfer of 0 bytes. A null transfer causes an error bit (*En*) in EMR to get set and the *En* bit in SER remains set, essentially preventing any further transfers on that channel without clearing the associated error registers.

[Table 11-8](#) summarizes the conditions and effects of null and dummy transfer requests.

**Table 11-8. Dummy and Null Transfer Request**

| Feature   | Null TR | Dummy TR |
|---|---------|----------|
| EMR/EMRH/QEMR is set                            | Yes     | No       |
| SER/SERH/QSER remains set                       | Yes     | No       |
| Link update (STATIC = 0 in OPT)                 | Yes     | Yes      |
| QER is set                                      | Yes     | Yes      |
| IPR/IPRH CER/CERH is set using early completion | Yes     | Yes      |

### 11.3.3.6 Parameter Set Updates

When a TR is submitted for a given DMA/QDMA channel and its corresponding PaRAM set, the EDMA3CC is responsible for updating the PaRAM set in anticipation of the next trigger event. For events that are not final, this includes address and count updates; for final events, this includes the link update.

The specific PaRAM set entries that are updated depend on the channel's synchronization type (A-synchronized or B-synchronized) and the current state of the PaRAM set. A B-update refers to the decrementing of BCNT in the case of A-synchronized transfers after the submission of successive TRs. A C-update refers to the decrementing of CCNT in the case of A-synchronized transfers after BCNT TRs for ACNT byte transfers have submitted. For AB-synchronized transfers, a C-update refers to the decrementing of CCNT after submission of every transfer request.

See [Table 11-9](#) for details and conditions on the parameter updates. A link update occurs when the PaRAM set is exhausted, as described in [Section 11.3.3.7](#).

After the TR is read from the PaRAM (and is in process of being submitted to EDMA3TC), the following fields are updated if needed:

- A-synchronized: BCNT, CCNT, SRC, DST.
- AB-synchronized: CCNT, SRC, DST.

The following fields are not updated (except for during linking, where all fields are overwritten by the link PaRAM set):

- A-synchronized: ACNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK.
- AB-synchronized: ACNT, BCNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK.

Note that PaRAM updates only pertain to the information that is needed to properly submit the next transfer request to the EDMA3TC. Updates that occur while data is moved within a transfer request are tracked within the transfer controller, and is detailed in [Section 11.3.12](#). For A-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes (BCNT = 1 and CCNT = 1). For AB-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes of BCNT arrays (CCNT = 1). The EDMA3TC is responsible for updating source and destination addresses within the array based on ACNT and FWID (in OPT). For AB-synchronized transfers, the EDMA3TC is also responsible to update source and destination addresses between arrays based on SRCBIDX and DSTBIDX.

[Table 11-9](#) shows the details of parameter updates that occur within EDMA3CC for A-synchronized and AB-synchronized transfers.

**Table 11-9. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set)**

| Condition:         | A-Synchronized Transfer |                       |                        | AB-Synchronized Transfer |            |                |
|--------------------|-------------------------|-----------------------|------------------------|--------------------------|------------|----------------|
|                    | B-Update                | C-Update              | Link Update            | B-Update                 | C-Update   | Link Update    |
|                    | BCNT > 1                | BCNT == 1 && CCNT > 1 | BCNT == 1 && CCNT == 1 | N/A                      | CCNT > 1   | CCNT == 1      |
| SRC                | += SRCBIDX              | += SRCCIDX            | = Link.SRC             | in EDMA3TC               | += SRCCIDX | = Link.SRC     |
| DST                | += DSTBIDX              | += DSTCIDX            | = Link.DST             | in EDMA3TC               | += DSTCIDX | = Link.DST     |
| ACNT               | None                    | None                  | = Link.ACNT            | None                     | None       | = Link.ACNT    |
| BCNT               | -= 1                    | = BCNTRLD             | = Link.BCNT            | in EDMA3TC               | N/A        | = Link.BCNT    |
| CCNT               | None                    | -= 1                  | = Link.CCNT            | in EDMA3TC               | -=1        | = Link.CCNT    |
| SRCBIDX            | None                    | None                  | = Link.SRCBIDX         | in EDMA3TC               | None       | = Link.SRCBIDX |
| DSTBIDX            | None                    | None                  | = Link.DSTBIDX         | None                     | None       | = Link.DSTBIDX |
| SRCCIDX            | None                    | None                  | = Link.SRCCIDX         | in EDMA3TC               | None       | = Link.SRCCIDX |
| DSTCIDX            | None                    | None                  | = Link.DSTCIDX         | None                     | None       | = Link.DSTCIDX |
| LINK               | None                    | None                  | = Link.LINK            | None                     | None       | = Link.LINK    |
| BCNTRLD            | None                    | None                  | = Link.BCNTRLD         | None                     | None       | = Link.BCNTRLD |
| OPT <sup>(1)</sup> | None                    | None                  | = LINK.OPT             | None                     | None       | = LINK.OPT     |

<sup>(1)</sup> In all cases, no updates occur if OPT STATIC == 1 for the current PaRAM set.

---

**NOTE:** The EDMA3CC includes no special hardware to detect when an indexed address update calculation overflows/underflows. The address update will wrap across boundaries as programmed by the user. You should ensure that no transfer is allowed to cross internal port boundaries between peripherals. A single TR must target a single source/destination slave endpoint.

---

### 11.3.3.7 Linking Transfers

The EDMA3CC provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive/continuous transfers with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed to by the 16-bit link address field of the current parameter set. Linking only occurs when the STATIC bit in OPT is cleared.

---

**NOTE:** You should always link a transfer (EDMA3 or QDMA) to another useful transfer. If you must terminate a transfer, then you should link the transfer to a NULL parameter set. See [Section 11.3.3.3](#).

---

The link update occurs after the current PaRAM set event parameters have been exhausted. An event's parameters are exhausted when the EDMA3 channel controller has submitted all of the transfers that are associated with the PaRAM set.

A link update occurs for null and dummy transfers depending on the state of the STATIC bit in OPT and the LINK field. In both cases (null or dummy), if the value of LINK is FFFFh, then a null PaRAM set (with all 0s and LINK set to FFFFh) is written to the current PaRAM set. Similarly, if LINK is set to a value other than FFFFh, then the appropriate PaRAM location that LINK points to is copied to the current PaRAM set.

Once the channel completion conditions are met for an event, the transfer parameters that are located at the link address are loaded into the current DMA or QDMA channel's associated parameter set. This indicates that the EDMA3CC reads the entire set (eight words) from the PaRAM set specified by LINK and writes all eight words to the PaRAM set that is associated with the current channel. [Figure 11-11](#) shows an example of a linked transfer.

Any PaRAM set in the PaRAM can be used as a link/reload parameter set. The PaRAM sets associated with peripheral synchronization events (see [Section 11.3.6](#)) should only be used for linking if the corresponding events are disabled.

If a PaRAM set location is defined as a QDMA channel PaRAM set (by QCHMAPn), then copying the link PaRAM set into the current QDMA channel PaRAM set is recognized as a trigger event. It is latched in QER because a write to the trigger word was performed. You can use this feature to create a linked list of transfers using a single QDMA channel and multiple PaRAM sets. See [Section 11.3.4.2](#).

Linking to itself replicates the behavior of auto-initialization, thus facilitating the use of circular buffering and repetitive transfers. After an EDMA3 channel exhausts its current PaRAM set, it reloads all of the parameter set entries from another PaRAM set, which is initialized with values that are identical to the original PaRAM set. [Figure 11-11](#) shows an example of a linked to self transfer. Here, the PaRAM set 255 has the link field pointing to the address of parameter set 255 (linked to self).

---

**NOTE:** If the STATIC bit in OPT is set for a PaRAM set, then link updates are not performed.

---

### 11.3.3.8 Constant Addressing Mode Transfers/Alignment Issues

If either SAM or DAM is set (constant addressing mode), then the source or destination address must be aligned to a 256-bit aligned address, respectively, and the corresponding BIDX should be an even multiple of 32 bytes (256 bits). The EDMA3CC does not recognize errors here, but the EDMA3TC asserts an error if this is not true. See [Section 11.3.12.3](#).

---

**NOTE:** The constant addressing (CONST) mode has limited applicability. The EDMA3 should be configured for the constant addressing mode (SAM/DAM = 1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the constant addressing mode. See the device-specific data manual and/or peripheral user's guide to verify if the constant addressing mode is supported. If the constant addressing mode is not supported, the similar logical transfer can be achieved using the increment (INCR) mode (SAM/DAM =0) by appropriately programming the count and indices values.

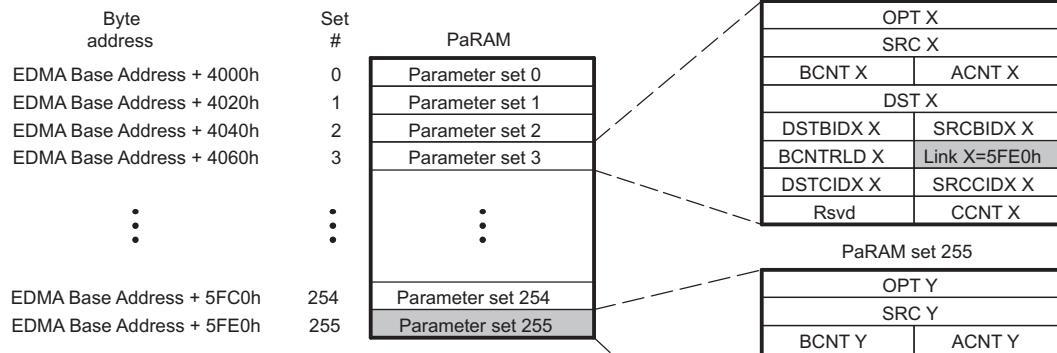
---

### 11.3.3.9 Element Size

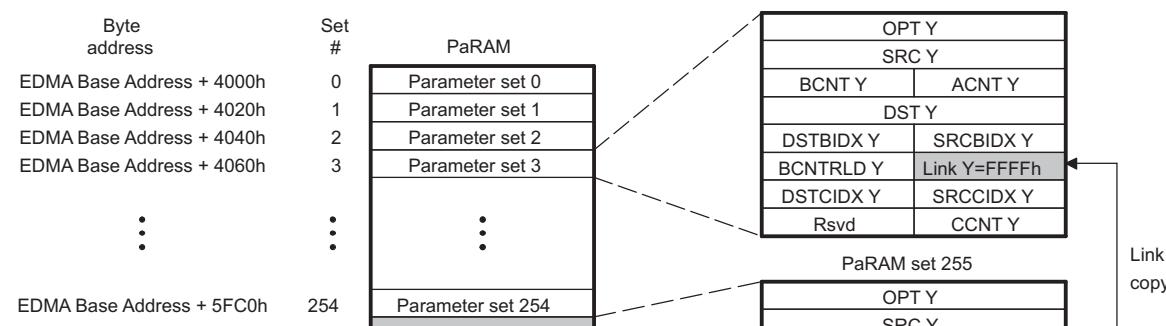
The EDMA3 controller does not use element-size and element-indexing. Instead, all transfers are defined in terms of all three dimensions: ACNT, BCNT, and CCNT. An element-indexed transfer is logically achieved by programming ACNT to the size of the element and BCNT to the number of elements that need to be transferred. For example, if you have 16-bit audio data and 256 audio samples that must be transferred to a serial port, you can only do this by programming the ACNT = 2 (2 bytes) and BCNT = 256.

**Figure 11-11. Linked Transfer**

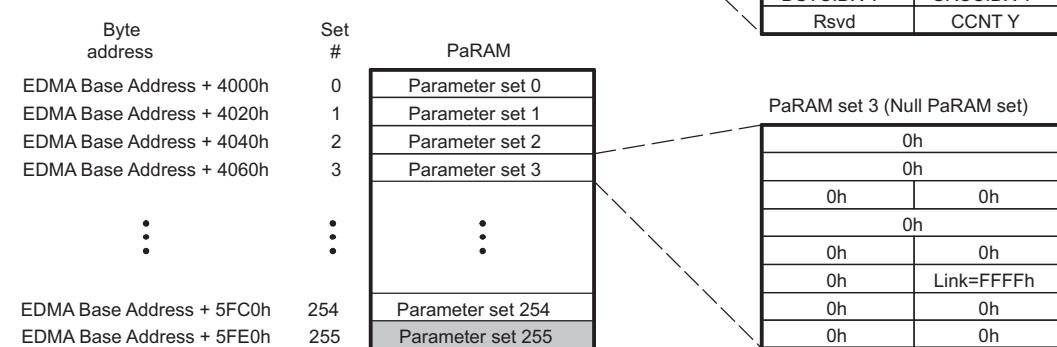
(a) At initialization



(b) After completion of PaRAM set 3 (link update)



(c) After completion of PaRAM set 51 (link to null set)



**Figure 11-12. Link-to-Self Transfer**

(a) At initialization

| Byte address              | Set # | PaRAM             |
|---------------------------|-------|-------------------|
| EDMA Base Address + 4000h | 0     | Parameter set 0   |
| EDMA Base Address + 4020h | 1     | Parameter set 1   |
| EDMA Base Address + 4040h | 2     | Parameter set 2   |
| EDMA Base Address + 4060h | 3     | Parameter set 3   |
| ⋮                         | ⋮     | ⋮                 |
| EDMA Base Address + 5FC0h | 254   | Parameter set 254 |
| EDMA Base Address + 5FE0h | 255   | Parameter set 255 |

PaRAM set 3

|           |            |
|-----------|------------|
| OPT X     |            |
| SRC X     |            |
| BCNT X    | ACNT X     |
| DST X     |            |
| DSTBIDX X | SRCBIDX X  |
| BCNTRLD X | Link=5FE0h |
| DSTCIDX X | SRCCIDX X  |
| Rsvd      | CCNT X     |

PaRAM set 255

|           |            |
|-----------|------------|
| OPT X     |            |
| SRC X     |            |
| BCNT X    | ACNT X     |
| DST X     |            |
| DSTBIDX Y | SRCBIDX X  |
| BCNTRLD X | Link=5FE0h |
| DSTCIDX X | SRCCIDX X  |
| Rsvd      | CCNT X     |

(b) After completion of PaRAM set 3  
(link update)

| Byte address              | Set # | PaRAM             |
|---------------------------|-------|-------------------|
| EDMA Base Address + 4000h | 0     | Parameter set 0   |
| EDMA Base Address + 4020h | 1     | Parameter set 1   |
| EDMA Base Address + 4040h | 2     | Parameter set 2   |
| EDMA Base Address + 4060h | 3     | Parameter set 3   |
| ⋮                         | ⋮     | ⋮                 |
| EDMA Base Address + 5FC0h | 254   | Parameter set 254 |
| EDMA Base Address + 5FE0h | 255   | Parameter set 255 |

PaRAM set 3

|           |            |
|-----------|------------|
| OPT X     |            |
| SRC X     |            |
| BCNT X    | ACNT X     |
| DST X     |            |
| DSTBIDX X | SRCBIDX X  |
| BCNTRLD X | Link=5FE0h |
| DSTCIDX X | SRCCIDX X  |
| Rsvd      | CCNT X     |

PaRAM set 255

Link update

(c) After completion of PaRAM set 127  
(link to self)

| Byte address              | Set # | PaRAM             |
|---------------------------|-------|-------------------|
| EDMA Base Address + 4000h | 0     | Parameter set 0   |
| EDMA Base Address + 4020h | 1     | Parameter set 1   |
| EDMA Base Address + 4040h | 2     | Parameter set 2   |
| EDMA Base Address + 4060h | 3     | Parameter set 3   |
| ⋮                         | ⋮     | ⋮                 |
| EDMA Base Address + 5FC0h | 254   | Parameter set 254 |
| EDMA Base Address + 5FE0h | 255   | Parameter set 255 |

PaRAM set 3

|           |            |
|-----------|------------|
| OPT X     |            |
| SRC X     |            |
| BCNT X    | ACNT X     |
| DST X     |            |
| DSTBIDX X | SRCBIDX X  |
| BCNTRLD X | Link=5FE0h |
| DSTCIDX X | SRCCIDX X  |
| Rsvd      | CCNT X     |

#### 11.3.4 Initiating a DMA Transfer

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. Transfers on DMA channels are initiated by three sources.

They are listed as follows:

- **Event-triggered transfer request** (this is the more typical usage of EDMA3): A peripheral, system, or externally-generated event triggers a transfer request.
- **Manually-triggered transfer request**: The CPU manually triggers a transfer by writing a 1 to the

corresponding bit in the event set register (ESR/ESRH).

- **Chain-triggered transfer request:** A transfer is triggered on the completion of another transfer or sub-transfer.

Transfers on QDMA channels are initiated by two sources. They are as follows:

- **Auto-triggered transfer request:** Writing to the programmed trigger word triggers a transfer.
- **Link-triggered transfer requests:** Writing to the trigger word triggers the transfer when linking occurs.

#### 11.3.4.1 DMA Channel

##### 11.3.4.1.1 Event-Triggered Transfer Request

When an event is asserted from a peripheral or device pins, it gets latched in the corresponding bit of the event register (ER.En = 1). If the corresponding event in the event enable register (EER) is enabled (EER.En = 1), then the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

If the PaRAM set is valid (not a NULL set), then a transfer request packet (TRP) is submitted to the EDMA3TC and the En bit in ER is cleared. At this point, a new event can be safely received by the EDMA3CC.

If the PaRAM set associated with the channel is a NULL set (see [Section 11.3.3.3](#)), then no transfer request (TR) is submitted and the corresponding En bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.En = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include cleaning the event missed error before re-triggering the DMA channel.

When an event is received, the corresponding event bit in the event register is set (ER.En = 1), regardless of the state of EER.En. If the event is disabled when an external event is received (ER.En = 1 and EER.En = 0), the ER.En bit remains set. If the event is subsequently enabled (EER.En = 1), then the pending event is processed by the EDMA3CC and the TR is processed/submitted, after which the ER.En bit is cleared.

If an event is being processed (prioritized or is in the event queue) and another sync event is received for the same channel prior to the original being cleared (ER.En != 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.En = 1).

See [Section 9.2.3, EDMA Event Multiplexing](#), for a description of how DMA events map to the EDMA event crossbar. See [Section 11.3.20, EDMA Events](#), for a table of direct and crossbar mapped EDMA events.

##### 11.3.4.1.2 Manually Triggered Transfer Request

The CPU or any EDMA programmer initiates a DMA transfer by writing to the event set register (ESR). Writing a 1 to an event bit in the ESR results in the event being prioritized/queued in the appropriate event queue, regardless of the state of the EER.En bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see [Section 11.3.3.3](#)), then no transfer request (TR) is submitted and the corresponding En bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.En = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include clearing the event missed error before re-triggering the DMA channel.

If an event is being processed (prioritized or is in the event queue) and the same channel is manually set by a write to the corresponding channel bit of the event set register (ESR.En = 1) prior to the original being cleared (ESR.En = 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.En = 1).

#### 11.3.4.1.3 Chain-Triggered Transfer Request

Chaining is a mechanism by which the completion of one transfer automatically sets the event for another channel. When a chained completion code is detected, the value of which is dictated by the transfer completion code (TCC[5:0] in OPT of the PaRAM set associated with the channel), it results in the corresponding bit in the chained event register (CER) to be set (CER.E[TCC] = 1).

Once a bit is set in CER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see [Section 11.3.3.3](#)), then no transfer request (TR) is submitted and the corresponding En bit in CER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.En = 1) to indicate that the event was discarded due to a null TR being serviced. In this case, the error condition must be cleared by you before the DMA channel can be re-triggered. Good programming practices might include clearing the event missed error before re-triggering the DMA channel.

If a chaining event is being processed (prioritized or queued) and another chained event is received for the same channel prior to the original being cleared (CER.En != 0), then the second chained event is registered as a missed event in the corresponding channel bit of the event missed register (EMR.En = 1).

---

**NOTE:** Chained event registers, event registers, and event set registers operate independently. An event (En) can be triggered by any of the trigger sources (event-triggered, manually-triggered, or chain-triggered).

---

#### 11.3.4.2 QDMA Channels

##### 11.3.4.2.1 Auto-triggered and Link-Triggered Transfer Request

QDMA-based transfer requests are issued when a QDMA event gets latched in the QDMA event register (QER.En = 1). A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when the following occurs:

- A CPU (or any EDMA3 programmer) write occurs to a PaRAM address that is defined as a QDMA channel trigger word (programmed in the QDMA channel mapping register (QCHMAP $n$ )) for the particular QDMA channel and the QDMA channel is enabled via the QDMA event enable register (QEER.En = 1).
- EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel (matches QCHMAP $n$  settings) and the corresponding channel is enabled via the QDMA event enable register (QEER.En = 1).

Once a bit is set in QER, the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If a bit is already set in QER (QER.En = 1) and a second QDMA event for the same QDMA channel occurs prior to the original being cleared, the second QDMA event gets captured in the QDMA event miss register (QEMR.En = 1).

#### 11.3.4.3 Comparison Between DMA and QDMA Channels

The primary difference between DMA and QDMA channels is the event/channel synchronization. QDMA events are either auto-triggered or link triggered. auto-triggering allows QDMA channels to be triggered by CPU(s) with a minimum number of linear writes to PaRAM. Link triggering allows a linked list of transfers to be executed, using a single QDMA PaRAM set and multiple link PaRAM sets.

A QDMA transfer is triggered when a CPU (or other EDMA3 programmer) writes to the trigger word of the QDMA channel parameter set (auto-triggered) or when the EDMA3CC performs a link update on a PaRAM set that has been mapped to a QDMA channel (link triggered). Note that for CPU triggered (manually triggered) DMA channels, in addition to writing to the PaRAM set, it is required to write to the event set register (ESR) to kick-off the transfer.

QDMA channels are typically for cases where a single event will accomplish a complete transfer since the CPU (or EDMA3 programmer) must reprogram some portion of the QDMA PaRAM set in order to re-trigger the channel. In other words, QDMA transfers are programmed with BCNT = CCNT = 1 for A-synchronized transfers, and CCNT = 1 for AB-synchronized transfers.

Additionally, since linking is also supported (if STATIC = 0 in OPT) for QDMA transfers, it allows you to initiate a linked list of QDMAs, so when EDMA3CC copies over a link PaRAM set (including the write to the trigger word), the current PaRAM set mapped to the QDMA channel will automatically be recognized as a valid QDMA event and initiate another set of transfers as specified by the linked set.

### 11.3.5 Completion of a DMA Transfer

A parameter set for a given channel is complete when the required number of transfer requests is submitted (based on receiving the number of synchronization events). The expected number of TRs for a non-null/non-dummy transfer is shown in [Table 11-10](#) for both synchronization types along with state of the PaRAM set prior to the final TR being submitted. When the counts (BCNT and/or CCNT) are this value, the next TR results in a:

- Final chaining or interrupt codes to be sent by the transfer controllers (instead of intermediate).
- Link updates (linking to either null or another valid link set).

**Table 11-10. Expected Number of Transfers for Non-Null Transfer**

| Sync Mode       | Counts at time 0     | Total # Transfers                     | Counts prior to final TR |
|-----------------|----------------------|---------------------------------------|--------------------------|
| A-synchronized  | ACNT<br>BCNT<br>CCNT | (BCNT × CCNT ) TRs of ACNT bytes each | BCNT == 1 && CCNT == 1   |
| AB-synchronized | ACNT<br>BCNT<br>CCNT | CCNT TRs for ACNT × BCNT bytes each   | CCNT == 1                |

You must program the PaRAM OPT field with a specific transfer completion code (TCC) along with the other OPT fields (TCCHEN, TCINTEN, ITCCHEN, and ITCINTEN bits) to indicate whether the completion code is to be used for generating a chained event or/and for generating an interrupt upon completion of a transfer.

The specific TCC value (6-bit binary value) programmed dictates which of the 64-bits in the chain event register (CER[TCC]) and/or interrupt pending register (IPR[TCC]) is set.

You can also selectively program whether the transfer controller sends back completion codes on completion of the final transfer request (TR) of a parameter set (TCCHEN or TCINTEN), for all but the final transfer request (TR) of a parameter set (ITCCHEN or ITCINTEN), or for all TRs of a parameter set (both). See [Section 11.3.8](#) for details on chaining (intermediate/final chaining) and [Section 11.3.9](#) for details on intermediate/final interrupt completion.

A completion detection interface exists between the EDMA3 channel controller and transfer controller(s). This interface sends back information from the transfer controller to the channel controller to indicate that a specific transfer is completed. Completion of a transfer is used for generating chained events and/or generating interrupts to the CPU(s).

All DMA/QDMA PaRAM sets must also specify a link address value. For repetitive transfers such as ping-pong buffers, the link address value should point to another predefined PaRAM set. Alternatively, a non-repetitive transfer should set the link address value to the null link value. The null link value is defined as FFFFh. See [Section 11.3.3.7](#) for more details.

**NOTE:** Any incoming events that are mapped to a null PaRAM set results in an error condition. The error condition should be cleared before the corresponding channel is used again. See [Section 11.3.3.5](#).

There are three ways the EDMA3CC gets updated/informed about a transfer completion: normal completion, early completion, and dummy/null completion. This applies to both chained events and completion interrupt generation.

#### 11.3.5.1 Normal Completion

In normal completion mode (TCCMODE = 0 in OPT), the transfer or sub-transfer is considered to be complete when the EDMA3 channel controller receives the completion codes from the EDMA3 transfer controller. In this mode, the completion code to the channel controller is posted by the transfer controller after it receives a signal from the destination peripheral. Normal completion is typically used to generate an interrupt to inform the CPU that a set of data is ready for processing.

#### 11.3.5.2 Early Completion

In early completion mode (TCCMODE = 1 in OPT), the transfer is considered to be complete when the EDMA3 channel controller submits the transfer request (TR) to the EDMA3 transfer controller. In this mode, the channel controller generates the completion code internally. Early completion is typically useful for chaining, as it allows subsequent transfers to be chained-triggered while the previous transfer is still in progress within the transfer controller, maximizing the overall throughput of the set of the transfers.

#### 11.3.5.3 Dummy or Null Completion

This is a variation of early completion. Dummy or null completion is associated with a dummy set ([Section 11.3.3.4](#)) or null set ([Section 11.3.3.3](#)). In both cases, the EDMA3 channel controller does not submit the associated transfer request to the EDMA3 transfer controller(s). However, if the set (dummy/null) has the OPT field programmed to return completion code (intermediate/final interrupt/chaining completion), then it will set the appropriate bits in the interrupt pending registers (IPR/IPRH) or chained event register (CER/CERH). The internal early completion path is used by the channel controller to return the completion codes internally (that is, EDMA3CC generates the completion code).

#### 11.3.6 Event, Channel, and PaRAM Mapping

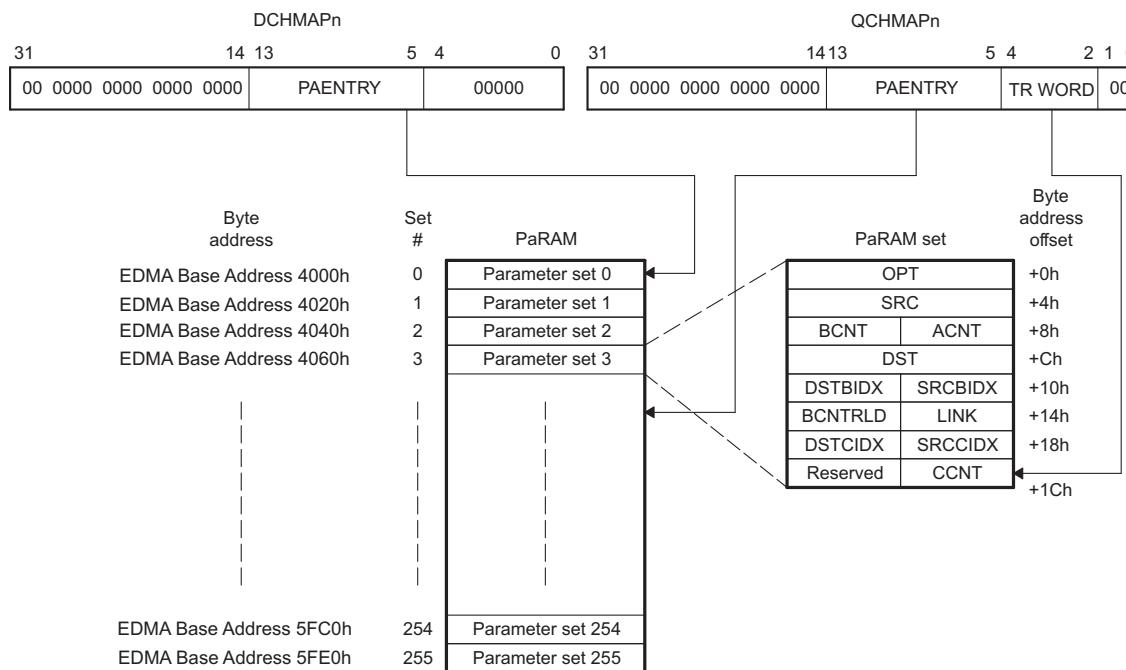
Several of the 64 DMA channels are tied to a specific hardware event, thus allowing events from device peripherals or external hardware to trigger transfers. A DMA channel typically requests a data transfer when it receives its event (apart from manually-triggered, chain-triggered, and other transfers). The amount of data transferred per synchronization event depends on the channel's configuration (ACNT, BCNT, CCNT, etc.) and the synchronization type (A-synchronized or AB-synchronized).

The association of an event to a channel is fixed, each DMA channel has one specific event associated with it. See [Section 9.2.3, EDMA Event Multiplexing](#), for a description of how DMA events map to the EDMA event crossbar. See [Section 11.3.20, EDMA Events](#), for a table of direct and crossbar mapped EDMA events.

In an application, if a channel does not use the associated synchronization event or if it does not have an associated synchronization event (unused), that channel can be used for manually-triggered or chained-triggered transfers, for linking/reloading, or as a QDMA channel.

##### 11.3.6.1 DMA Channel to PaRAM Mapping

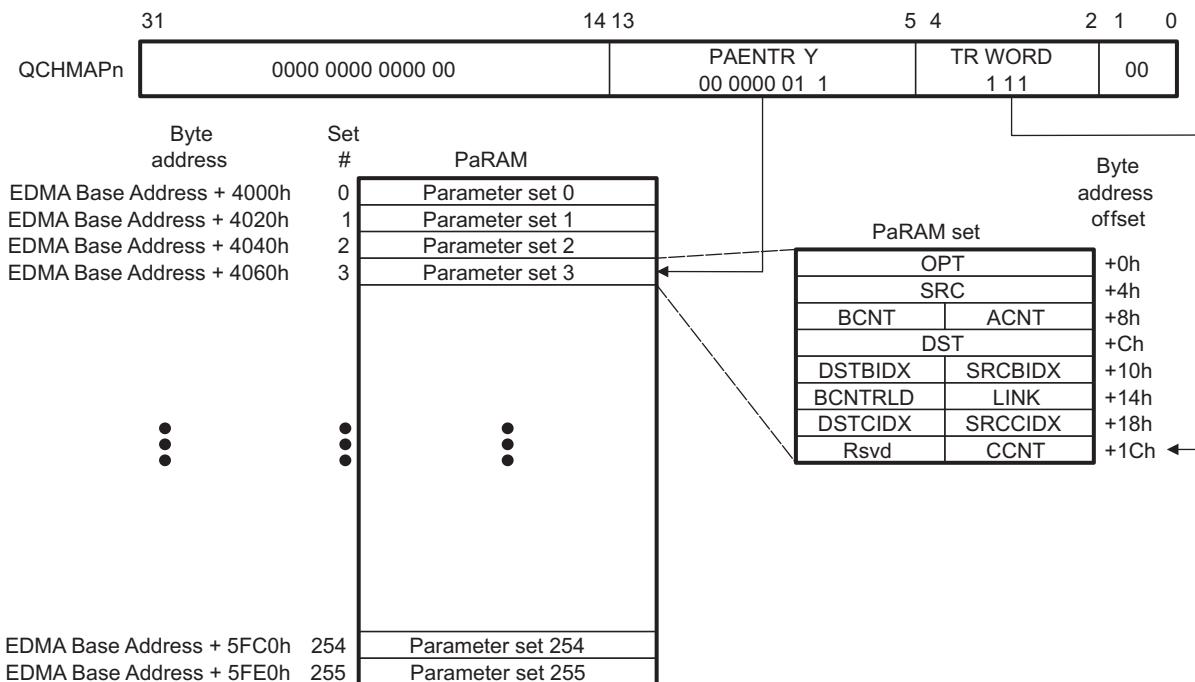
The mapping between the DMA channel numbers and the PaRAM sets is programmable (see [Table 11-5](#)). The DMA channel mapping registers (DCHMAPn) in the EDMA3CC provide programmability that allows the DMA channels to be mapped to any of the PaRAM sets in the PaRAM memory map. [Figure 11-13](#) illustrates the use of DCHMAP. There is one DCHMAP register per channel.

**Figure 11-13. DMA Channel and QDMA Channel to PaRAM Mapping**


### 11.3.6.2 QDMA Channel to PaRAM Mapping

The mapping between the QDMA channels and the PaRAM sets is programmable. The QDMA channel mapping register (QCHMAP) in the EDMA3CC allows you to map the QDMA channels to any of the PaRAM sets in the PaRAM memory map. [Figure 11-14](#) illustrates the use of QCHMAP.

Additionally, QCHMAP allows you to program the trigger word in the PaRAM set for the QDMA channel. A trigger word is one of the eight words in the PaRAM set. For a QDMA transfer to occur, a valid TR synchronization event for EDMA3CC is a write to the trigger word in the PaRAM set pointed to by QCHMAP for a particular QDMA channel. By default, QDMA channels are mapped to PaRAM set 0. You must appropriately re-map PaRAM set 0 before you use it.

**Figure 11-14. QDMA Channel to PaRAM Mapping**


### 11.3.7 EDMA3 Channel Controller Regions

The EDMA3 channel controller divides its address space into eight regions. Individual channel resources are assigned to a specific region, where each region is typically assigned to a specific EDMA programmer.

You can design the application software to use regions or to ignore them altogether. You can use active memory protection in conjunction with regions so that only a specific EDMA programmer (for example, privilege identification) or privilege level (for example, user vs. supervisor) is allowed access to a given region, and thus to a given DMA or QDMA channel. This allows robust system-level DMA code where each EDMA programmer only modifies the state of the assigned resources. Memory protection is described in [Section 11.3.10](#).

#### 11.3.7.1 Region Overview

The EDMA3 channel controller memory-mapped registers are divided in three main categories:

1. Global registers
2. Global region channel registers
3. Shadow region channel registers

The global registers are located at a single/fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information.

The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow *n* channel region address range(s). For example, the event enable register (EER) is visible at the global address of EDMA Base Address + 1020h or region addresses of EDMA Base Address + 2020h for region 0, EDMA Base Address + 2220h for region 1, ..., EDMA Base Address + 2E20h for region 7.

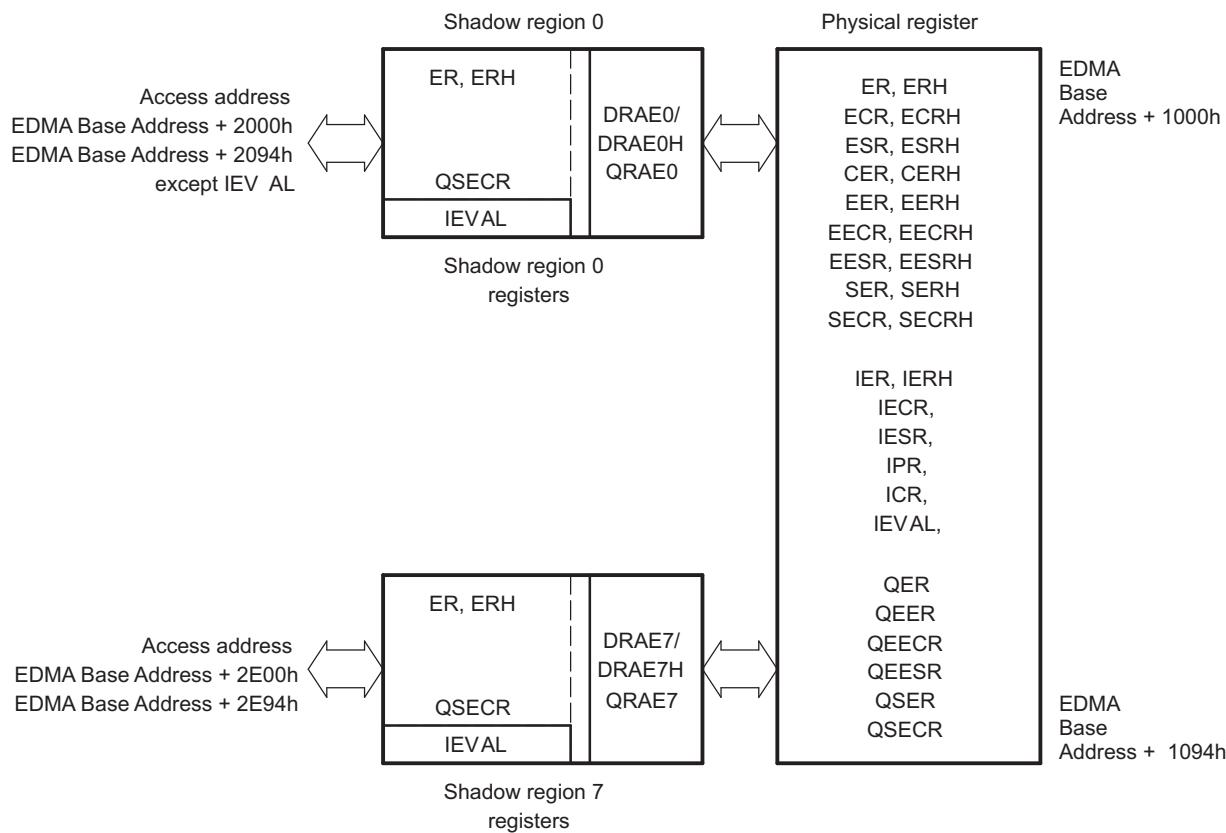
The DMA region access enable registers (DRAEm) and the QDMA region access enable registers (QRAEn) control the underlying control register bits that are accessible via the shadow region address space (except for IEVALn). [Table 11-11](#) lists the registers in the shadow region memory map. See the EDMA3CC memory map ([Table 11-25](#)) for the complete global and shadow region memory maps. [Figure 11-15](#) illustrates the conceptual view of the regions.

**Table 11-11. Shadow Region Registers**

| DRAEm | DRAEHm | QRAEn |
|-------|--------|-------|
| ER    | ERH    | QER   |
| ECR   | ECRH   | QEER  |
| ESR   | ESRH   | QEECR |
| CER   | CERH   | QEESR |
| EER   | EERH   |       |
| EECR  | EECRH  |       |
| EESR  | EESRH  |       |
| SER   | SERH   |       |
| SECR  | SECRH  |       |
| IER   | IERH   |       |
| IECR  | IECRH  |       |
| IESR  | IESRH  |       |
| IPR   | IPRH   |       |
| ICR   | ICRH   |       |

**Register not affected by DRAEDRAEH**

IEVAL

**Figure 11-15. Shadow Region Registers**

### 11.3.7.2 Channel Controller Regions

There are eight EDMA3 shadow regions (and associated memory maps). Associated with each shadow region are a set of registers defining which channels and interrupt completion codes belong to that region. These registers are user-programmed per region to assign ownership of the DMA/QDMA channels to a region.

- **DRAEm and DRAEHm:** One register pair exists for each of the shadow regions. The number of bits in each register pair matches the number of DMA channels (64 DMA channels). These registers need to be programmed to assign ownership of DMA channels and interrupt (or TCC codes) to the respective region. Accesses to DMA and interrupt registers via the shadow region address view are filtered through the DRAE/DRAEH pair. A value of 1 in the corresponding DRAE(H) bit implies that the corresponding DMA/interrupt channel is accessible; a value of 0 in the corresponding DRAE(H) bit forces writes to be discarded and returns a value of 0 for reads.
- **QRAEn:** One register exists for every region. The number of bits in each register matches the number of QDMA channels (4 QDMA channels). These registers must be programmed to assign ownership of QDMA channels to the respective region. To enable a channel in a shadow region using shadow region 0 QEER, the respective bit in QRAE must be set or writing into QEESR will not have the desired effect.
- **MPPAn and MPPAG:** One register exists for every region. This register defines the privilege level, requestor, and types of accesses allowed to a region's memory-mapped registers.

It is typical for an application to have a unique assignment of QDMA/DMA channels (and, therefore, a given bit position) to a given region.

The use of shadow regions allows for restricted access to EDMA3 resources (DMA channels, QDMA channels, TCC, interrupts) by tasks in a system by setting or clearing bits in the DRAE/ORAE registers. If exclusive access to any given channel / TCC code is required for a region, then only that region's DRAE/ORAE should have the associated bit set.

#### **Example 11-1. Resource Pool Division Across Two Regions**

This example illustrates a judicious resource pool division across two regions, assuming region 0 must be allocated 16 DMA channels (0-15) and 1 QDMA channel (0) and 32 TCC codes (0-15 and 48-63). Region 1 needs to be allocated 16 DMA channels (16-32) and the remaining 7 QDMA channels (1-7) and TCC codes (16-47). DRAE should be equal to the OR of the bits that are required for the DMA channels and the TCC codes:

```
Region 0: DRAEH, DRAE = 0xFFFF0000, 0x0000FFFF QRAE = 0x00000001
Region 1: DRAEH, DRAE = 0x0000FFFF, 0xFFFF0000 QRAE = 0x00000FE
```

#### **11.3.7.3 Region Interrupts**

In addition to the EDMA3CC global completion interrupt, there is an additional completion interrupt line that is associated with every shadow region. Along with the interrupt enable register (IER), DRAE acts as a secondary interrupt enable for the respective shadow region interrupts. See [Section 11.3.9](#) for more information.

#### **11.3.8 Chaining EDMA3 Channels**

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. The purpose is to allow you the ability to chain several events through one event occurrence.

Chaining is different from linking ([Section 11.3.3.7](#)). The EDMA3 link feature reloads the current channel parameter set with the linked parameter set. The EDMA3 chaining feature does not modify or update any channel parameter set; it provides a synchronization event to the chained channel (see [Section 11.3.4.1.3](#) for chain-triggered transfer requests).

Chaining is achieved at either final transfer completion or intermediate transfer completion, or both, of the current channel. Consider a channel  $m$  (DMA/QDMA) required to chain to channel  $n$ . Channel number  $n$  (0-63) needs to be programmed into the TCC bit of channel  $m$  channel options parameter (OPT) set.

- If final transfer completion chaining (TCCHEN = 1 in OPT) is enabled, the chain-triggered event occurs after the submission of the last transfer request of channel  $m$  is either submitted or completed (depending on early or normal completion).
- If intermediate transfer completion chaining (ITCCHEN = 1 in OPT) is enabled, the chain-triggered

event occurs after every transfer request, except the last of channel  $m$  is either submitted or completed (depending on early or normal completion).

- If both final and intermediate transfer completion chaining (TCCHEN = 1 and ITCCHEN = 1 in OPT) are enabled, then the chain-trigger event occurs after every transfer request is submitted or completed (depending on early or normal completion).

**Table 11-12** illustrates the number of chain event triggers occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 11-12. Chain Event Triggers**

| Options                 | (Number of chained event triggers on channel 30) |                                  |
|-------------------------|--|----------------------------------|
|                         | A-Synchronized                                   | AB-Synchronized                  |
| TCCHEN = 1, ITCCHEN = 0 | 1 (Owing to the last TR)                         | 1 (Owing to the last TR)         |
| TCCHEN = 0, ITCCHEN = 1 | 19 (Owing to all but the last TR)                | 4 (Owing to all but the last TR) |
| TCCHEN = 1, ITCCHEN = 1 | 20 (Owing to a total of 20 TRs)                  | 5 (Owing to a total of 5 TRs)    |

### 11.3.9 EDMA3 Interrupts

The EDMA3 interrupts are divided into 2 categories: transfer completion interrupts and error interrupts.

There are nine region interrupts, eight shadow regions and one global region. The transfer completion interrupts are listed in **Table 11-13**. The transfer completion interrupts and the error interrupts from the transfer controllers are all routed to the ARM interrupt controllers.

**Table 11-13. EDMA3 Transfer Completion Interrupts**

| Name         | Description   |
|--------------|---|
| EDMA3CC_INT0 | EDMA3CC Transfer Completion Interrupt Shadow Region 0 |
| EDMA3CC_INT1 | EDMA3CC Transfer Completion Interrupt Shadow Region 1 |
| EDMA3CC_INT2 | EDMA3CC Transfer Completion Interrupt Shadow Region 2 |
| EDMA3CC_INT3 | EDMA3CC Transfer Completion Interrupt Shadow Region 3 |
| EDMA3CC_INT4 | EDMA3CC Transfer Completion Interrupt Shadow Region 4 |
| EDMA3CC_INT5 | EDMA3CC Transfer Completion Interrupt Shadow Region 5 |
| EDMA3CC_INT6 | EDMA3CC Transfer Completion Interrupt Shadow Region 6 |
| EDMA3CC_INT7 | EDMA3CC Transfer Completion Interrupt Shadow Region 7 |

**Table 11-14. EDMA3 Error Interrupts**

| Name            | Description                         |
|-----------------|-------------------------------------|
| EDMA3CC_ERRINT  | EDMA3CC Error Interrupt             |
| EDMA3CC_MPINT   | EDMA3CC Memory Protection Interrupt |
| EDMA3TC0_ERRINT | TC0 Error Interrupt                 |
| EDMA3TC1_ERRINT | TC1 Error Interrupt                 |
| EDMA3TC2_ERRINT | TC2 Error Interrupt                 |
| EDMA3TC3_ERRINT | TC3 Error Interrupt                 |

#### 11.3.9.1 Transfer Completion Interrupts

The EDMA3CC is responsible for generating transfer completion interrupts to the CPU(s) (and other EDMA3 masters). The EDMA3 generates a single completion interrupt per shadow region, as well as one for the global region on behalf of all 64 channels. The various control registers and bit fields facilitate EDMA3 interrupt generation.

The software architecture should either use the global interrupt or the shadow interrupts, but not both.

The transfer completion code (TCC) value is directly mapped to the bits of the interrupt pending register (IPR/IPRH). For example, if TCC = 10 0001b, IPRH[1] is set after transfer completion, and results in interrupt generation to the CPU(s) if the completion interrupt is enabled for the CPU. See [Section 11.3.9.1.1](#) for details on enabling EDMA3 transfer completion interrupts.

When a completion code is returned (as a result of early or normal completions), the corresponding bit in IPR/IPRH is set if transfer completion interrupt (final/intermediate) is enabled in the channel options parameter (OPT) for a PaRAM set associated with the transfer.

**Table 11-15. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping**

| TCC Bits in OPT<br>(TCINTEN/ITCINTEN = 1) | IPR Bit Set | TCC Bits in OPT<br>(TCINTEN/ITCINTEN = 1) | IPRH Bit Set <sup>(1)</sup> |
|---|-------------|---|-----------------------------|
| 0   | IPR0        | 20h                                       | IPR32/IPRH0                 |
| 1   | IPR1        | 21h                                       | IPR33/IPRH1                 |
| 2h  | IPR2        | 22h                                       | IPR34/IPRH2                 |
| 3h  | IPR3        | 23h                                       | IPR35/IPRH3                 |
| 4h  | IPR4        | 24h                                       | IPR36/IPRH4                 |
| ...                                       | ...         | ...                                       | ...                         |
| 1Eh                                       | IPR30       | 3Eh                                       | IPR62/IPRH30                |
| 1Fh                                       | IPR31       | 3Fh                                       | IPR63/IPRH31                |

<sup>(1)</sup> Bit fields IPR[32-63] correspond to bits 0 to 31 in IPRH, respectively.

You can program the transfer completion code (TCC) to any value for a DMA/QDMA channel. A direct relation between the channel number and the transfer completion code value does not need to exist. This allows multiple channels having the same transfer completion code value to cause a CPU to execute the same interrupt service routine (ISR) for different channels.

If the channel is used in the context of a shadow region and you intend for the shadow region interrupt to be asserted, then ensure that the bit corresponding to the TCC code is enabled in IER/IERH and in the corresponding shadow region's DMA region access registers (DRAE/DRAEH).

You can enable Interrupt generation at either final transfer completion or intermediate transfer completion, or both. Consider channel *m* as an example.

- If the final transfer interrupt (TCCINT = 1 in OPT) is enabled, the interrupt occurs after the last transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If the intermediate transfer interrupt (ITCCINT = 1 in OPT) is enabled, the interrupt occurs after every transfer request, except the last TR of channel *m* is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion interrupts (TCCINT = 1, and ITCCINT = 1 in OPT) are enabled, then the interrupt occurs after every transfer request is submitted or completed (depending on early or normal completion).

[Table 11-16](#) shows the number of interrupts that occur in different synchronized scenarios. Consider channel 31, programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 11-16. Number of Interrupts**

| Options                   | A-Synchronized           | AB-Synchronized         |
|---------------------------|--------------------------|-------------------------|
| TCINTEN = 1, ITCINTEN = 0 | 1 (Last TR)              | 1 (Last TR)             |
| TCINTEN = 0, ITCINTEN = 1 | 19 (All but the last TR) | 4 (All but the last TR) |
| TCINTEN = 1, ITCINTEN = 1 | 20 (All TRs)             | 5 (All TRs)             |

### 11.3.9.1.1 Enabling Transfer Completion Interrupts

For the EDMA3 channel controller to assert a transfer completion to the external environment, the interrupts must be enabled in the EDMA3CC. This is in addition to setting up the TCINTEN and ITCINTEN bits in OPT of the associated PaRAM set.

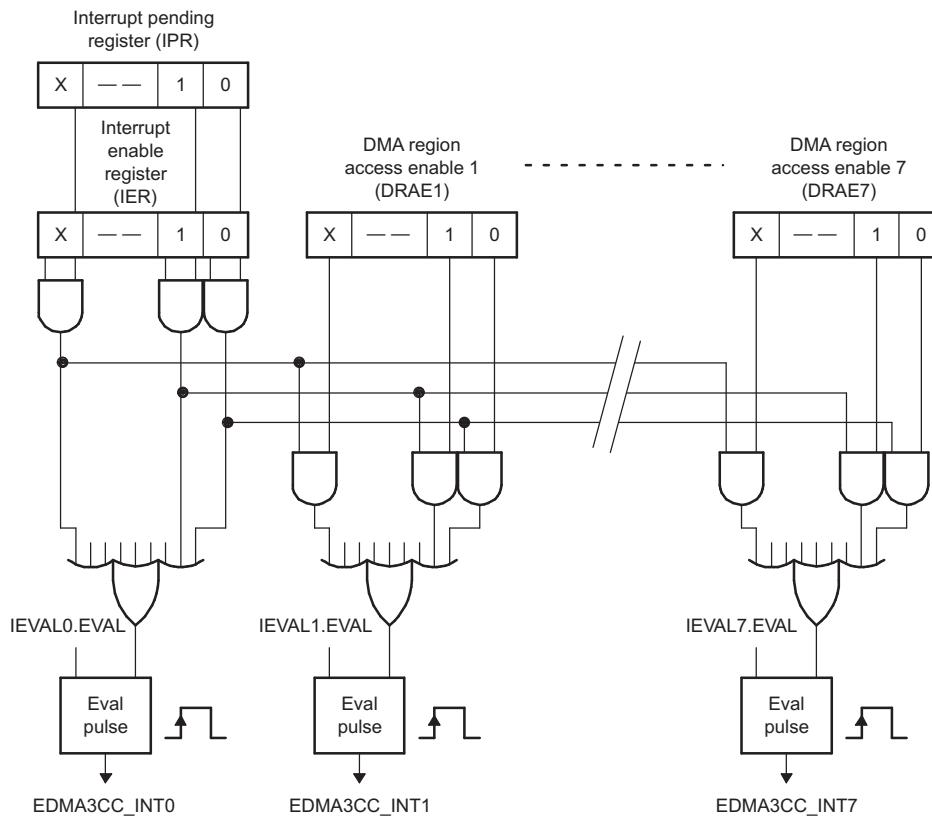
The EDMA3 channel controller has interrupt enable registers (IER/IERH) and each bit location in IER/IERH serves as a primary enable for the corresponding interrupt pending registers (IPR/IPRH).

All of the interrupt registers (IER, IESR, IEGR, and IPR) are either manipulated from the global DMA channel region, or by the DMA channel shadow regions. The shadow regions provide a view to the same set of physical registers that are in the global region.

The EDMA3 channel controller has a hierarchical completion interrupt scheme that uses a single set of interrupt pending registers (IPR/IPRH) and single set of interrupt enable registers (IER/IERH). The programmable DMA region access enable registers (DRAE/DRAEH) provides a second level of interrupt masking. The global region interrupt output is gated based on the enable mask that is provided by IER/IERH. see [Figure 11-16](#)

The region interrupt outputs are gated by IER and the specific DRAE/DRAEH associated with the region. See [Figure 11-16](#).

**Figure 11-16. Interrupt Diagram**



For the EDMA3CC to generate the transfer completion interrupts that are associated with each shadow region, the following conditions must be true:

- EDMA3CC\_INT0: (IPR.E0 & IER.E0 & DRAE0.E0) | (IPR.E1 & IER.E1 & DRAE0.E1) | ... | (IPR.E63 & IER.E63 & DRAE0.E63)
- EDMA3CC\_INT1: (IPR.E0 & IER.E0 & DRAE1.E0) | (IPR.E1 & IER.E1 & DRAE1.E1) | ... | (IPR.E63 & IER.E63 & DRAE1.E63)
- EDMA3CC\_INT2 : (IPR.E0 & IER.E0 & DRAE2.E0) | (IPR.E1 & IER.E1 & DRAE2.E1) | ... | (IPR.E63 & IER.E63 & DRAE2.E63)....

- Up to EDMA3CC\_INT7 : (IPR.E0 & IER.E0 & DRAE7.E0) | (IPR.E1 & IER.E1 & DRAE7.E1) | ...|(IPRH.E63 & IERH.E63 & DRAEH7.E63)

**NOTE:** The DRAE/DRAEH for all regions are expected to be set up at system initialization and to remain static for an extended period of time. The interrupt enable registers should be used for dynamic enable/disable of individual interrupts.

Because there is no relation between the TCC value and the DMA/QDMA channel, it is possible, for example, for DMA channel 0 to have the OPT.TCC = 63 in its associated PaRAM set. This would mean that if a transfer completion interrupt is enabled (OPT.TCINTEN or OPT.ITCINTEN is set), then based on the TCC value, IPRH.E63 is set up on completion. For proper channel operations and interrupt generation using the shadow region map, you must program the DRAE/DRAEH that is associated with the shadow region to have read/write access to both bit 0 (corresponding to channel 0) and bit 63 (corresponding to IPRH bit that is set upon completion).

#### 11.3.9.1.2 Clearing Transfer Completion Interrupts

Transfer completion interrupts that are latched to the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bit in the interrupt pending clear register (ICR/ICRH). For example, a write of 1 to ICR.E0 clears a pending interrupt in IPR.E0.

If an incoming transfer completion code (TCC) gets latched to a bit in IPR/IPRH, then additional bits that get set due to a subsequent transfer completion will not result in asserting the EDMA3CC completion interrupt. In order for the completion interrupt to be pulsed, the required transition is from a state where no enabled interrupts are set to a state where at least one enabled interrupt is set.

#### 11.3.9.2 EDMA3 Interrupt Servicing

Upon completion of a transfer (early or normal completion), the EDMA3 channel controller sets the appropriate bit in the interrupt pending registers (IPR/IPRH), as the transfer completion codes specify. If the completion interrupts are appropriately enabled, then the CPU enters the interrupt service routine (ISR) when the completion interrupt is asserted.

After servicing the interrupt, the ISR should clear the corresponding bit in IPR/IPRH, thereby enabling recognition of future interrupts. The EDMA3CC will only assert additional completion interrupts when all IPR/IPRH bits clear.

When one interrupt is serviced many other transfer completions may result in additional bits being set in IPR/IPRH, thereby resulting in additional interrupts. Each of the bits in IPR/IPRH may need different types of service; therefore, the ISR may check all pending interrupts and continue until all of the posted interrupts are serviced appropriately.

Examples of pseudo code for a CPU interrupt service routine for an EDMA3CC completion interrupt are shown in [Example 11-2](#) and [Example 11-3](#).

The ISR routine in [Example 11-2](#) is more exhaustive and incurs a higher latency.

#### Example 11-2. Interrupt Servicing

The pseudo code:

1. Reads the interrupt pending register (IPR/IPRH).
2. Performs the operations needed.
3. Writes to the interrupt pending clear register (ICR/ICRH) to clear the corresponding IPR/IPRH bit(s).
4. Reads IPR/IPRH again:
  - (a) If IPR/IPRH is not equal to 0, repeat from step 2 (implies occurrence of new event between step 2 to step 4).
  - (b) If IPR/IPRH is equal to 0, this should assure you that all of the enabled interrupts are inactive.

**Example 11-2. Interrupt Servicing (continued)**

**NOTE:** An event may occur during step 4 while the IPR/IPRH bits are read as 0 and the application is still in the interrupt service routine. If this happens, a new interrupt is recorded in the device interrupt controller and a new interrupt generates as soon as the application exits in the interrupt service routine.

Example 11-3 is less rigorous, with less burden on the software in polling for set interrupt bits, but can occasionally cause a race condition as mentioned above.

**Example 11-3. Interrupt Servicing**

If you want to leave any enabled and pending (possibly lower priority) interrupts; you must force the interrupt logic to reassert the interrupt pulse by setting the EVAL bit in the interrupt evaluation register (IEVAL).

The pseudo code is as follows:

1. Enters ISR.
2. Reads IPR/IPRH.
3. For the condition that is set in IPR/IPRH that you want to service, do the following:
  - (a) Service interrupt as the application requires.
  - (b) Clear the bit for serviced conditions (others may still be set, and other transfers may have resulted in returning the TCC to EDMA3CC after step 2).
4. Reads IPR/IPRH prior to exiting the ISR:
  - (a) If IPR/IPRH is equal to 0, then exit the ISR.
  - (b) If IPR/IPRH is not equal to 0, then set IEVAL so that upon exit of ISR, a new interrupt triggers if any enabled interrupts are still pending.

#### 11.3.9.3 Interrupt Evaluation Operations

The EDMA3CC has interrupt evaluate registers (IEVAL) that exist in the global region and in each shadow region. The registers in the shadow region are the only registers in the DMA channel shadow region memory map that are not affected by the settings for the DMA region access enable registers (DRAE/DRAEH). Writing a 1 to the EVAL bit in the registers that are associated with a particular shadow region results in pulsing the associated region interrupt (global or shadow), if any enabled interrupt (via IER/IERH) is still pending (IPR/IPRH). This register assures that the CPU does not miss the interrupts (or the EDMA3 master associated with the shadow region) if the software architecture chooses not to use all interrupts. See Example 11-3 for the use of IEVAL in the EDMA3 interrupt service routine (ISR).

Similarly an error evaluation register (EEVAL) exists in the global region. Writing a 1 to the EVAL bit in EEVAL causes the pulsing of the error interrupt if any pending errors are in EMR/EMRH, QEMR, or CCERR. See Section 11.3.9.4 for additional information regarding error interrupts.

**NOTE:** While using IEVAL for shadow region completion interrupts, you should make sure that the IEVAL operated upon is from that particular shadow region memory map.

#### 11.3.9.4 Error Interrupts

The EDMA3CC error registers provide the capability to differentiate error conditions (event missed, threshold exceed, etc.). Additionally, setting the error bits in these registers results in asserting the EDMA3CC error interrupt. If the EDMA3CC error interrupt is enabled in the device interrupt controller(s), then it allows the CPU(s) to handle the error conditions.

The EDMA3CC has a single error interrupt (EDMA3CC\_ERRINT) that is asserted for all EDMA3CC error conditions. There are four conditions that cause the error interrupt to pulse:

- DMA missed events: for all 64 DMA channels. DMA missed events are latched in the event missed registers (EMR/EMRH).
- QDMA missed events: for all 8 QDMA channels. QDMA missed events are latched in the QDMA event missed register (QEMR).
- Threshold exceed: for all event queues. These are latched in EDMA3CC error register (CCERR).
- TCC error: for outstanding transfer requests that are expected to return completion code (TCCHEN or TCINTEN bit in OPT is set to 1) exceeding the maximum limit of 63. This is also latched in the EDMA3CC error register (CCERR).

[Figure 11-17](#) illustrates the EDMA3CC error interrupt generation operation.

If any of the bits are set in the error registers due to any error condition, the EDMA3CC\_ERRINT is always asserted, as there are no enables for masking these error events. Similar to transfer completion interrupts (EDMA3CC\_INT), the error interrupt also only pulses when the error interrupt condition transitions from no errors being set to at least one error being set. If additional error events are latched prior to the original error bits clearing, the EDMA3CC does not generate additional interrupt pulses.

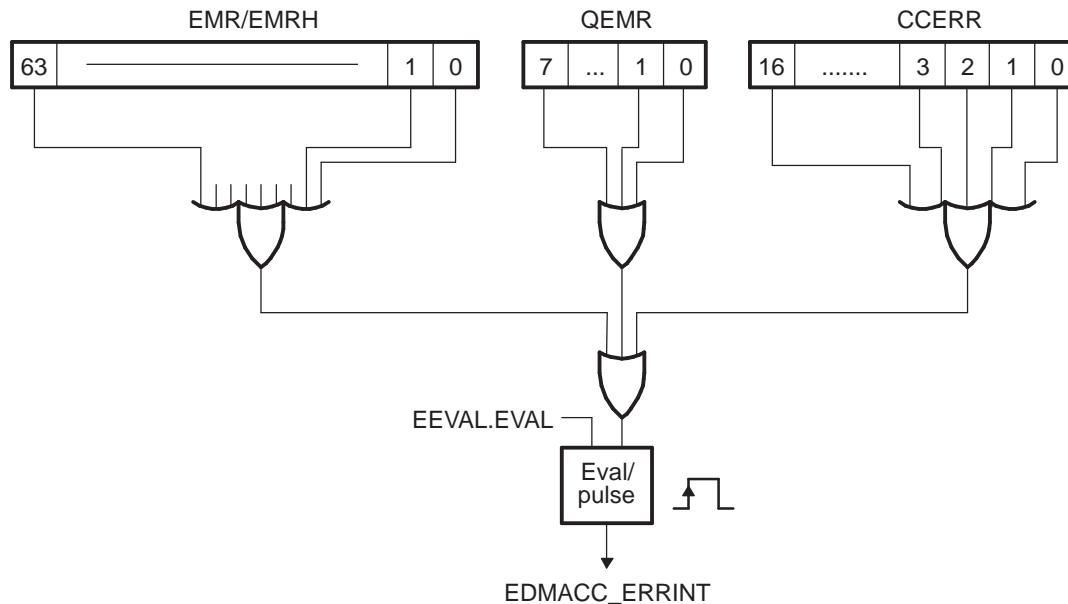
To reduce the burden on the software, there is an error evaluate register (EEVAL) that allows re-evaluation of pending set error events/bits, similar to the interrupt evaluate register (IEVAL). You can use this so that the CPU(s) does not miss any error events.

---

**NOTE:** It is good practice to enable the error interrupt in the device interrupt controller and to associate an interrupt service routine with it to address the various error conditions appropriately. Doing so puts less burden on the software (polling for error status); additionally, it provides a good debug mechanism for unexpected error conditions.

---

**Figure 11-17. Error Interrupt Operation**



### 11.3.10 Memory Protection

The EDMA3 channel controller supports two kinds of memory protection: active and proxy.

#### 11.3.10.1 Active Memory Protection

Active memory protection is a feature that allows or prevents read and write accesses (by any EDMA3 programmer) to the EDMA3CC registers (based on permission characteristics that you program). Active memory protection is achieved by a set of memory protection permissions attribute (MPPA) registers.

The EDMA3CC register map is divided into three categories:

- a global region.
- a global channel region.
- eight shadow regions.

Each shadow region consists of the respective shadow region registers and the associated PaRAM. For more detailed information regarding the contents of a shadow region, refer to section [Table 11-11](#).

Each of the eight shadow regions has an associated MPPA register (MPPAn) that defines the specific requestor(s) and types of requests that are allowed to the regions resources.

The global channel region is also protected with a memory-mapped register (MPPAG). The MPPAG applies to the global region and to the global channel region, except the other MPPA registers themselves. For more detailed information on the list of the registers in each region, refer to the register memory-map in section [Table 11-18](#).

See [Section 11.4.1.5.4](#) for the bit field descriptions of MPPAn. The MPPAn have a certain set of access rules.

[Table 11-17](#) shows the accesses that are allowed or not allowed to the MPPAG and MPPAn. The active memory protection uses the PRIV and PRIVID attributes of the EDMA programmer. The PRIV is the privilege level (i.e., user vs. supervisor). The PRIVID refers to a privilege ID with a number that is associated with an EDMA3 programmer. See the device-specific data manual for the PRIVIDs that are associated with potential EDMA3 programmers.

**Table 11-17. Allowed Accesses**

| Access | Supervisor | User |
|--------|------------|------|
| Read   | Yes        | Yes  |
| Write  | Yes        | No   |

[Table 11-18](#) describes the MPPA register mapping for the shadow regions (which includes shadow region registers and PaRAM addresses).

The region-based MPPA registers are used to protect accesses to the DMA shadow regions and the associated region PaRAM. Because there are eight regions, there are eight MPPA region registers (MPPA[0-7]).

**Table 11-18. MPPA Registers to Region Assignment**

| Register | Registers Protect | Address Range | PaRAM Protect <sup>(1)</sup> | Address Range |
|----------|-------------------|---------------|------------------------------|---------------|
| MPPAG    | Global Range      | 0000h-1FFCh   | N/A                          | N/A           |
| MPPA0    | DMA Shadow 0      | 2000h-21FCh   | 1st octant                   | 4000h-47FCh   |
| MPPA1    | DMA Shadow 1      | 2200h-23FCh   | 2nd octant                   | 4800h-4FFCh   |
| MPPA2    | DMA Shadow 2      | 2400h-25FCh   | 3rd octant                   | 5000h-57FCh   |
| MPPA3    | DMA Shadow 3      | 2600h-27FCh   | 4th octant                   | 5800h-5FFCh   |
| MPPA4    | DMA Shadow 4      | 2800h-29FCh   | 5th octant                   | 6000h-67FCh   |
| MPPA5    | DMA Shadow 5      | 2A00h-2BFCh   | 6th octant                   | 6800h-6FFCh   |
| MPPA6    | DMA Shadow 6      | 2C00h-2DFCh   | 7th octant                   | 7000h-77FCh   |
| MPPA7    | DMA Shadow 7      | 2E00h-2FFCh   | 8th octant                   | 7800h-7FFCh   |

<sup>(1)</sup> The PARAM region is divided into 8 regions referred to as an octant.

### Example Access denied.

Write access to shadow region 7's event enable set register (EESR):

1. The original value of the event enable register (EER) at address offset 0x1020 is 0x0.
2. The MPPA[7] is set to prevent user level accesses (UW = 0, UR = 0), but it allows supervisor level accesses (SW = 1, SR = 1) with a privilege ID of 0. (AID0 = 1).
3. An EDMA3 programmer with a privilege ID of 0 attempts to perform a user-level write of a value of 0xFF00FF00 to shadow region 7's event enable set register (EESR) at address offset 0x2E30. Note that the EER is a read-only register and the only way that you can write to it is by writing to the EESR. Also remember that there is only one physical register for EER, EESR, etc. and that the shadow regions only provide to the same physical set.
4. Since the MPPA[7] has UW = 0, though the privilege ID of the write access is set to 0, the access is not allowed and the EER is not written to.

**Table 11-19. Example Access Denied**

| Register                   | Value            | Description   |
|----------------------------|------------------|---|
| EER<br>(offset 0x1020)     | 0x0000 0000      | Value in EER to begin with.   |
| EESR<br>(offset 0x2E30)    | 0xFF00 FF00<br>↓ | Value attempted to be written to shadow region 7's EESR.<br>This is done by an EDMA3 programmer with a privilege level of User and Privilege ID of 0. |
| MPPA[7]<br>(offset 0x082C) | 0x0000 04B0      | Memory Protection Filter AID0 = 1, UW = 0, UR = 0, SW = 1, SR = 1.  |
| EER<br>(offset 0x1020)     | X                | Access Denied   |
|                            | 0x0000 0000      | Final value of EER  |

### Example Access Allowed

Write access to shadow region 7's event enable set register (EESR):

1. The original value of the event enable register (EER) at address offset 0x1020 is 0x0.
2. The MPPA[7] is set to allow user-level accesses (UW = 1, UR = 1) and supervisor-level accesses (SW = 1, SR = 1) with a privilege ID of 0. (AID0 = 1).
3. An EDMA3 programmer with a privilege ID of 0, attempts to perform a user-level write of a value of 0xABCD0123 to shadow region 7's event enable set register (EESR) at address offset 0x2E30. Note that the EER is a read-only register and the only way that you can write to it is by writing to the EESR. Also remember that there is only one physical register for EER, EESR, etc. and that the shadow regions only provide to the same physical set.
4. Since the MPPA[7] has UW = 1 and AID0 = 1, the user-level write access is allowed.
5. Remember that accesses to shadow region registers are masked by their respective DRAE register. In this example, the DRAE[7] is set of 0x9FF00FC2.
6. The value finally written to EER is 0x8BC00102.

**Table 11-20. Example Access Allowed**

| Register                   | Value       | Description  |
|----------------------------|-------------|--|
| EER<br>(offset 0x1020)     | 0x0000 0000 | Value in EER to begin with.  |
| EESR<br>(offset 0x2E30)    | 0xFF00 FF00 | Value attempted to be written to shadow region 7's EESR. This is done by an EDMA3 programmer with a privilege level of User and Privilege ID of 0. |
| MPPA[7]<br>(offset 0x082C) | 0x0000 04B3 | Memory Protection Filter AID = 1, UW = 1, UR = 1, SW = 1, SR = 1.  |
|                            | ↓           | Access allowed.  |
| DRAE[7]<br>(offset 0x0378) | 0x9FF0 0FC2 | DMA Region Access Enable Filter  |
| EESR<br>(offset 0x2E30)    | 0x8BC0 0102 | Value written to shadow region 7's EESR. This is done by an EDMA3 programmer with a privilege level of User and a Privilege ID of 0.               |
| EER<br>(offset 0x1020)     | 0xBC0 0102  | Final value of EER.  |

### 11.3.10.2 Proxy Memory Protection

Proxy memory protection allows an EDMA3 transfer programmed by a given EDMA3 programmer to have its permissions travel with the transfer through the EDMA3TC. The permissions travel along with the read transactions to the source and the write transactions to the destination endpoints. The PRIV bit and PRIVID bit in the channel options parameter (OPT) is set with the EDMA3 programmer's PRIV value and PRIVID values, respectively, when any part of the PaRAM set is written.

The PRIV is the privilege level (i.e., user vs. supervisor). The PRIVID refers to a privilege ID with a number that is associated with an EDMA3 programmer.

See the data manual for the PRIVIDs that are associated with potential EDMA3 programmers.

These options are part of the TR that are submitted to the transfer controller. The transfer controller uses the above values on their respective read and write command bus so that the target endpoints can perform memory protection checks based on these values.

Consider a parameter set that is programmed by a CPU in user privilege level for a simple transfer with the source buffer on an L2 page and the destination buffer on an L1D page. The PRIV is 0 for user-level and the CPU has a PRIVID of 0.

The PaRAM set is shown in [Figure 11-18](#).

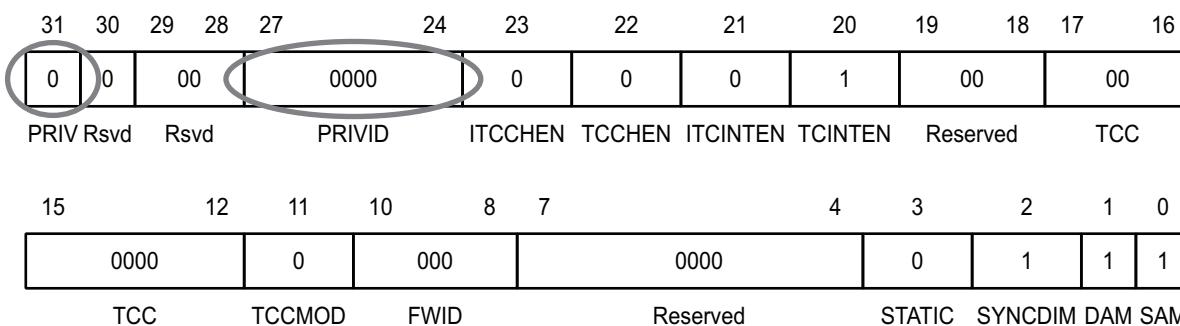
**Figure 11-18. PaRAM Set Content for Proxy Memory Protection Example**

(a) EDMA3 Parameters

| Parameter Contents |       | Parameter                         |                                |
|--------------------|-------|-----------------------------------|--------------------------------|
| 0010 0007h         |       | Channel Options Parameter (OPT)   |                                |
| 009F 0000h         |       | Channel Source Address (SRC)      |                                |
| 0001h              | 0004h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| 00F0 7800h         |       | Channel Destination Address (DST) |                                |
| 0001h              | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0000h              | FFFFh | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0001h              | 1000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h              | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

**Figure 11-19. Channel Options Parameter (OPT) Example**

(b) Channel Options Parameter (OPT) Content



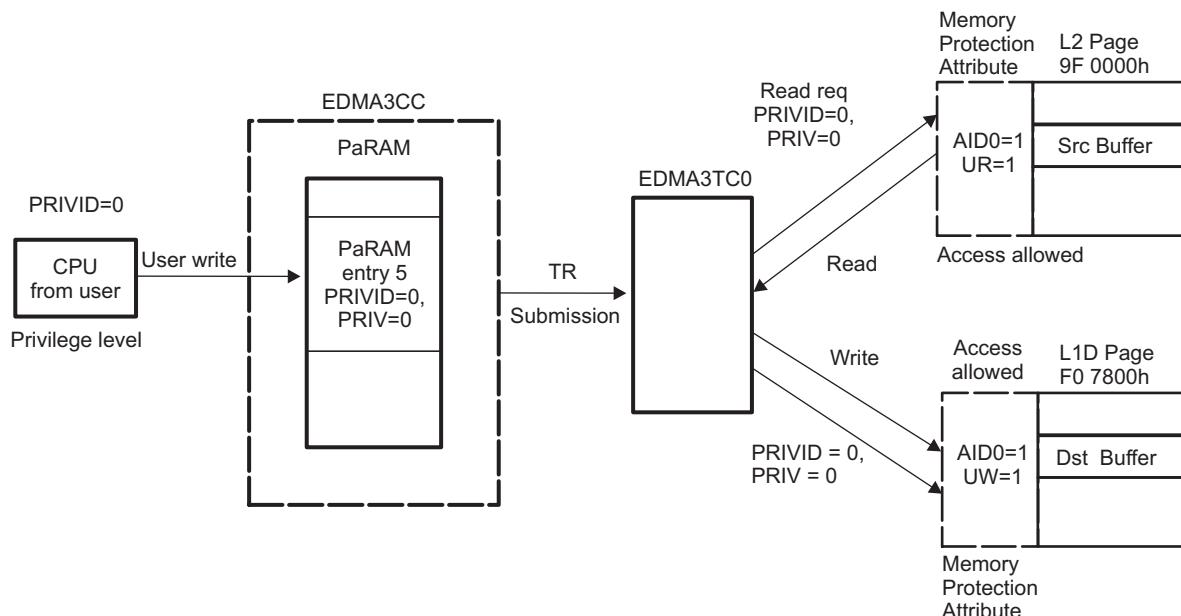
The PRIV and PRIVID information travels along with the read and write requests that are issued to the source and destination memories.

For example, if the access attributes that are associated with the L2 page with the source buffer only allow supervisor read, write accesses (SR,SW), the user-level read request above is refused. Similarly, if the access attributes that are associated with the L1D page with the destination buffer only allow supervisor read and write accesses (SR, SW), the user-level write request above is refused. For the transfer to succeed, the source and destination pages should have user-read and user-write permissions, respectively, along with allowing accesses from a PRIVID 0.

Because the programmer's privilege level and privilege identification travel with the read and write requests, EDMA3 acts as a proxy.

[Figure 11-20](#) illustrates the propagation of PRIV and PRIVID at the boundaries of all the interacting entities (CPU, EDMA3CC, EDMA3TC, and slave memories).

Figure 11-20. Proxy Memory Protection Example



### 11.3.11 Event Queue(s)

Event queues are a part of the EDMA3 channel controller. Event queues form the interface between the event detection logic in the EDMA3CC and the transfer request (TR) submission logic of the EDMA3CC. Each queue is 16 entries deep; thus, each event queue can queue a maximum of 16 events. If there are more than 16 events, then the events that cannot find a place in the event queue remain set in the associated event register and the CPU does not stall.

There are four event queues for the device: Queue0, Queue1, Queue2, and Queue3. Events in Queue0 result in submission of its associated transfer requests (TRs) to TC0. Similarly, transfer requests that are associated with events in Queue3 are submitted to TC3.

An event that wins prioritization against other DMA and/or QDMA pending events is placed at the tail of the appropriate event queue. Each event queue is serviced in FIFO order. Once the event reaches the head of its queue and the corresponding transfer controller is ready to receive another TR, the event is de-queued and the PaRAM set corresponding to the de-queued event is processed and submitted as a transfer request packet (TRP) to the associated EDMA3 transfer controller.

Queue0 has highest priority and Queue3 has the lowest priority, if Queue0 and Queue1 both have at least one event entry and if both TC0 and TC1 can accept transfer requests, then the event in Queue0 is de-queued first and its associated PaRAM set is processed and submitted as a transfer request (TR) to TC0.

See [Section 11.3.11.4](#) for system-level performance considerations. All of the event entries in all of the event queues are software readable (not writeable) by accessing the event entry registers (Q0E0, Q0E1,...Q1E15, etc.). Each event entry register characterizes the queued event in terms of the type of event (manual, event, chained or auto-triggered) and the event number. See [Section 11.4.1.4.1](#) for a description of the bit fields in the queue event entry registers.

#### 11.3.11.1 DMA/QDMA Channel to Event Queue Mapping

Each of the 64 DMA channels and eight QDMA channels are programmed independently to map to a specific queue, using the DMA queue number register (DMAQNUM) and the QDMA queue number register (QDMANUM). The mapping of DMA/QDMA channels is critical to achieving the desired performance level for the EDMA and most importantly, in meeting real-time deadlines. See [Section 11.3.11.4](#).

---

**NOTE:** If an event is ready to be queued and both the event queue and the EDMA3 transfer controller that is associated to the event queue are empty, then the event bypasses the event queue, and moves the PaRAM processing logic, and eventually to the transfer request submission logic for submission to the EDMA3TC. In this case, the event is not logged in the event queue status registers.

---

#### 11.3.11.2 Queue RAM Debug Visibility

There are four event queues and each queue has 16 entries. These 16 entries are managed in a circular FIFO manner. There is a queue status register (QSTAT) associated with each queue. These along with all of the 16 entries per queue can be read via registers QSTAT $n$  and QxEy, respectively.

These registers provide user visibility and may be helpful while debugging real-time issues (typically post-mortem), involving multiple events and event sources. The event queue entry register (QxEy) uniquely identifies the specific event type (event-triggered, manually-triggered, chain-triggered, and QDMA events) along with the event number (for all DMA/QDMA event channels) that are in the queue or have been de-queued (passed through the queue).

Each of the 16 entries in the event queue are read using the EDMA3CC memory-mapped register. By reading the event queue, you see the history of the last 16 TRs that have been processed by the EDMA3 on a given queue. This provides user/software visibility and is helpful for debugging real-time issues (typically post-mortem), involving multiple events and event sources.

The queue status register (QSTAT $n$ ) includes fields for the start pointer (STRTPTR) which provides the offset to the head entry of an event. It also includes a field called NUMVAL that provides the total number of valid entries residing in the event queue at a given instance of time. The STRTPTR may be used to index appropriately into the 16 event entries. NUMVAL number of entries starting from STRTPTR are indicative of events still queued in the respective queue. The remaining entry may be read to determine what's already de-queued and submitted to the associated transfer controller.

#### 11.3.11.3 Queue Resource Tracking

The EDMA3CC event queue includes watermarking/threshold logic that allows you to keep track of maximum usage of all event queues. This is useful for debugging real-time deadline violations that may result from head-of-line blocking on a given EDMA3 event queue.

You can program the maximum number of events that can queue up in an event queue by programming the threshold value (between 0 to 15) in the queue watermark threshold A register (QWMTHRA). The maximum queue usage is recorded actively in the watermark (WM) field of the queue status register (QSTAT $n$ ) that keeps getting updated based on a comparison of number of valid entries, which is also visible in the NUMVAL bit in QSTAT $n$  and the maximum number of entries (WM bit in QSTAT $n$ ).

If the queue usage is exceeded, this status is visible in the EDMA3CC registers: the QTHRXC $n$  bit in the channel controller error register (CCERR) and the THRXC $n$  bit in QSTAT $n$ , where  $n$  stands for the event queue number. Any bits that are set in CCERR also generate an EDMA3CC error interrupt.

#### 11.3.11.4 Performance Considerations

The main system bus infrastructure (L3) arbitrates bus requests from all of the masters (TCs, CPU(S), and other bus masters) to the shared slave resources (peripherals and memories).

The priorities of transfer requests (read and write commands) from the EDMA3 transfer controllers with respect to other masters within the system crossbar are programmed using the queue priority register (QUEPRI). QUEPRI programs the priority of the event queues (or indirectly, TC0-TC3, because QueueN transfer requests are submitted to TCN).

Therefore, the priority of unloading queues has a secondary affect compared to the priority of the transfers as they are executed by the EDMA3TC (dictated by the priority set using QUEPRI).

### 11.3.12 EDMA3 Transfer Controller (EDMA3TC)

The EDMA3 channel controller is the user-interface of the EDMA3 and the EDMA3 transfer controller (EDMA3TC) is the data movement engine of the EDMA3. The EDMA3CC submits transfer requests (TR) to the EDMA3TC and the EDMA3TC performs the data transfers dictated by the TR; thus, the EDMA3TC is a slave to the EDMA3CC.

#### 11.3.12.1 Architecture Details

##### 11.3.12.1.1 Command Fragmentation

The TC read and write controllers in conjunction with the source and destination register sets are responsible for issuing optimally-sized reads and writes to the slave endpoints. An optimally-sized command is defined by the transfer controller default burst size (DBS), which is defined in [Section 11.3.12.5](#).

The EDMA3TC attempts to issue the largest possible command size as limited by the DBS value or the ACNT/BCNT value of the TR. EDMA3TC obeys the following rules:

- The read/write controllers always issue commands less than or equal to the DBS value.
- The first command of a 1D transfer command always aligns the address of subsequent commands to the DBS value.

[Table 11-21](#) lists the TR segmentation rules that are followed by the EDMA3TC. In summary, if the ACNT value is larger than the DBS value, then the EDMA3TC breaks the ACNT array into DBS-sized commands to the source/destination addresses. Each BCNT number of arrays are then serviced in succession.

For BCNT arrays of ACNT bytes (that is, a 2D transfer), if the ACNT value is less than or equal to the DBS value, then the TR may be optimized into a 1D-transfer in order to maximize efficiency. The optimization takes place if the EDMA3TC recognizes that the 2D-transfer is organized as a single dimension (ACNT == BIDX) and the ACNT value is a power of 2.

[Table 11-21](#) lists conditions in which the optimizations are performed.

**Table 11-21. Read/Write Command Optimization Rules**

| ACNT ≤ DBS | ACNT is power of 2 | BIDX = ACNT | BCNT ≤ 1023 | SAM/DAM = Increment | Description   |
|------------|--------------------|-------------|-------------|---------------------|---------------|
| Yes        | Yes                | Yes         | Yes         | Yes                 | Optimized     |
| No         | x                  | x           | x           | x                   | Not Optimized |
| x          | No                 | x           | x           | x                   | Not Optimized |
| x          | x                  | No          | x           | x                   | Not Optimized |
| x          | x                  | x           | No          | x                   | Not Optimized |
| x          | x                  | x           | x           | No                  | Not Optimized |

##### 11.3.12.1.2 TR Pipelining

TR pipelining refers to the ability of the source active set to proceed ahead of the destination active set. Essentially, the reads for a given TR may already be in progress while the writes of a previous TR may not have completed.

The number of outstanding TRs is limited by the number of destination FIFO register entries.

TR pipelining is useful for maintaining throughput on back-to-back small TRs. It minimizes the startup overhead because reads start in the background of a previous TR writes.

#### **Example 11-4. Command Fragmentation (DBS = 64)**

The pseudo code:

- ACNT = 8, BCNT = 8, SRCBIDX = 8, DSTBIDX = 10, SRCADDR = 64, DSTADDR = 191

Read Controller: This is optimized from a 2D-transfer to a 1D-transfer such that the read side is equivalent to ACNT = 64, BCNT = 1.

Cmd0 = 64 byte

Write Controller: Because DSTBIDX != ACNT, it is not optimized.

Cmd0 = 8 byte, Cmd1 = 8 byte, Cmd2 = 8 byte, Cmd3 = 8 byte, Cmd4 = 8 byte, Cmd5 = 8 byte, Cmd6 = 8 byte, Cmd7 = 8 byte.

- ACNT=128, BCNT = 1, SRCADDR = 63, DSTADDR = 513

Read Controller: Read address is not aligned.

Cmd0 = 1 byte, (now the SRCADDR is aligned to 64 for the next command)

Cmd1 = 64 bytes

Cmd2 = 63 bytes

Write Controller: The write address is also not aligned.

Cmd0 = 63 bytes, (now the DSTADDR is aligned to 64 for the next command)

Cmd1 = 64 bytes

Cmd2 = 1 byte

#### **11.3.12.1.3 Performance Tuning**

By default, reads are as issued as fast as possible. In some cases, the reads issued by the EDMA3TC could fill the available command buffering for a slave, delaying other (potentially higher priority) masters from successfully submitting commands to that slave. The rate at which read commands are issued by the EDMA3TC is controlled by the RDRATE register. The RDRATE register defines the number of cycles that the EDMA3TC read controller waits before issuing subsequent commands for a given TR, thus minimizing the chance of the EDMA3TC consuming all available slave resources. The RDRATE value should be set to a relatively small value if the transfer controller is targeted for high priority transfers and to a higher value if the transfer controller is targeted for low priority transfers.

In contrast, the Write Interface does not have any performance tuning knobs because writes always have an interval between commands as write commands are submitted along with the associated write data.

#### **11.3.12.2 Memory Protection**

The transfer controller plays an important role in handling proxy memory protection. There are two access properties associated with a transfer: for instance, the privilege id (system-wide identification assigned to a master) of the master initiating the transfer, and the privilege level (user versus supervisor) used to program the transfer. This information is maintained in the PaRAM set when it is programmed in the channel controller. When a TR is submitted to the transfer controller, this information is made available to the EDMA3TC and used by the EDMA3TC while issuing read and write commands. The read or write commands have the same privilege identification, and privilege level as that programmed in the EDMA3 transfer in the channel controller.

#### **11.3.12.3 Error Generation**

Errors are generated if enabled under three conditions:

- EDMA3TC detection of an error signaled by the source or destination address.
- Attempt to read or write to an invalid address in the configuration memory map.
- Detection of a constant addressing mode TR violating the constant addressing mode transfer rules (the source/destination addresses and source/destination indexes must be aligned to 32 bytes).

Either or all error types may be disabled. If an error bit is set and enabled, the error interrupt for the concerned transfer controller is pulsed.

#### 11.3.12.4 Debug Features

The DMA program register set, DMA source active register set, and the destination FIFO register set are used to derive a brief history of TRs serviced through the transfer controller.

Additionally, the EDMA3TC status register (TCSTAT) has dedicated bit fields to indicate the ongoing activity within different parts of the transfer controller:

- The SRCACTV bit indicates whether the source active set is active.
- The DSTACTV bit indicates the number of TRs resident in the destination register active set at a given instance.
- The PROGBUSY bit indicates whether a valid TR is present in the DMA program set.

If the TRs are in progression, caution must be used and you must realize that there is a chance that the values read from the EDMA3TC status registers will be inconsistent since the EDMA3TC may change the values of these registers due to ongoing activities.

It is recommended that you ensure no additional submission of TRs to the EDMA3TC in order to facilitate ease of debug.

##### 11.3.12.4.1 Destination FIFO Register Pointer

The destination FIFO register pointer is implemented as a circular buffer with the start pointer being DFSTRTPTTR and a buffer depth of usually 2 or 4. The EDMA3TC maintains two important status details in TCSTAT that may be used during advanced debugging, if necessary. The DFSTRTPTTR is a start pointer, that is, the index to the head of the destination FIFO register. The DSTACTV is a counter for the number of valid (occupied) entries. These registers may be used to get a brief history of transfers.

Examples of some register field values and their interpretation:

- DFSTRTPTTR = 0 and DSTACTV = 0 implies that no TRs are stored in the destination FIFO register.
- DFSTRTPTTR = 1 and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 1 and the second pending TR is read from the destination FIFO register entry 2.
- DFSTRTPTTR = 3h and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 3 and the second pending TR is read from the destination FIFO register entry 0.

#### 11.3.12.5 EDMA3TC Configuration

[Table 11-22](#) provides the configuration of the individual EDMA3 transfer controllers present on the device. The DBS for each transfer controller is configurable using the TPTC\_CFG register in the chip configuration module.

**Table 11-22. EDMA3 Transfer Controller Configurations**

| Name        | TC0          | TC1          | TC2          | TC3          |
|-------------|--------------|--------------|--------------|--------------|
| FIFOSIZE    | 512 bytes    | 512 bytes    | 512 bytes    | 512 bytes    |
| BUSWIDTH    | 128 bits     | 128 bits     | 128 bits     | 128 bits     |
| DSTREGDEPTH | 4 entries    | 4 entries    | 4 entries    | 4 entries    |
| DBS         | Configurable | Configurable | Configurable | Configurable |

### 11.3.13 Event Dataflow

This section summarizes the data flow of a single event, from the time the event is latched to the channel controller to the time the transfer completion code is returned. The following steps list the sequence of EDMA3CC activity:

1. Event is asserted from an external source (peripheral or external interrupt). This also is similar for a manually-triggered, chained-triggered, or QDMA-triggered event. The event is latched into the ER.En/ERH.En (or CER.En/CERH.En, ESR.En /ESRH.En, QER.En) bit.
2. Once an event is prioritized and queued into the appropriate event queue, the SER.En/SERH.En (or QSER.En) bit is set to inform the event prioritization/processing logic to disregard this event since it is already in the queue. Alternatively, if the transfer controller and the event queue are empty, then the event bypasses the queue.
3. The EDMA3CC processing and the submission logic evaluates the appropriate PaRAM set and determines whether it is a non-null and non-dummy transfer request (TR).
4. The EDMA3CC clears the ER.En/ERH.En (or CER.En/CERH.En, ESR.En/ESRH.En, QER.En) bit and the SER.En/SERH.En bit as soon as it determines the TR is non-null. In the case of a null set, the SER.En/SERH.En bit remains set. It submits the non-null/non-dummy TR to the associated transfer controller. If the TR was programmed for early completion, the EDMA3CC immediately sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]-32).
5. If the TR was programmed for normal completion, the EDMA3CC sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]) when the EDMA3TC informs the EDMA3CC about completion of the transfer (returns transfer completion codes).
6. The EDMA3CC programs the associated EDMA3TCn's Program Register Set with the TR.
7. The TR is then passed to the Source Active set and the DST FIFO Register Set, if both the register sets are available.
8. The Read Controller processes the TR by issuing read commands to the source slave endpoint. The Read Data lands in the Data FIFO of the EDMA3TCn.
9. As soon as sufficient data is available, the Write Controller begins processing the TR by issuing write commands to the destination slave endpoint.
10. This continues until the TR completes and the EDMA3TCn then signals completion status to the EDMA3CC.

### 11.3.14 EDMA3 Prioritization

The EDMA3 controller has many implementation rules to deal with concurrent events/channels, transfers, etc. The following subsections detail various arbitration details whenever there might be occurrence of concurrent activity. [Figure 11-21](#) shows the different places EDMA3 priorities come into play.

#### 11.3.14.1 Channel Priority

The DMA event registers (ER and ERH) capture up to 64 events; likewise, the QDMA event register (QER) captures QDMA events for all QDMA channels; therefore, it is possible for events to occur simultaneously on the DMA/QDMA event inputs. For events arriving simultaneously, the event associated with the lowest channel number is prioritized for submission to the event queues (for DMA events, channel 0 has the highest priority and channel 63 has the lowest priority; similarly, for QDMA events, channel 0 has the highest priority and channel 7 has the lowest priority). This mechanism only sorts simultaneous events for submission to the event queues.

If a DMA and QDMA event occurs simultaneously, the DMA event always has prioritization against the QDMA event for submission to the event queues.

#### 11.3.14.2 Trigger Source Priority

If a DMA channel is associated with more than one trigger source (event trigger, manual trigger, and chain trigger), and if multiple events are set simultaneously for the same channel ( $ER.E_n = 1$ ,  $ESR.E_n = 1$ ,  $CER.E_n = 1$ ), then the EDMA3CC always services these events in the following priority order: event trigger (via ER) is higher priority than chain trigger (via CER) and chain trigger is higher priority than manual trigger (via ESR).

This implies that if for channel 0, both  $ER.E_0 = 1$  and  $CER.E_0 = 1$  at the same time, then the  $ER.E_0$  event is always queued before the  $CER.E_0$  event.

#### 11.3.14.3 Dequeue Priority

The priority of the associated transfer request (TR) is further mitigated by which event queue is being used for event submission (dictated by DMAQNUM and QDMAQNUM). For submission of a TR to the transfer request, events need to be de-queued from the event queues. Queue 0 has the highest dequeue priority and queue 3 the lowest.

#### 11.3.14.4 System (Transfer Controller) Priority

`INIT_PRIORITY_0` and `INIT_PRIORITY_1` registers in the chip configuration module are used to configure the EDMA TC's priority through the system bus infrastructure. Additionally, the priority settings for DDR memory accesses are defined in the dynamic memory manager (DMM).

---

**NOTE:** The default priority for all TCs is the same, 0 or highest priority relative to other masters. It is recommended that this priority be changed based on system level considerations, such as real-time deadlines for all masters including the priority of the transfer controllers with respect to each other.

---

### 11.3.15 EDMA3 Operating Frequency (Clock Control)

The EDMA3 channel controller and transfer controller are clocked from `PLL_L3 SYSCLK4`. The EDMA3 system runs at the L3 clock frequency.

### 11.3.16 Reset Considerations

A hardware reset resets the EDMA3 (EDMA3CC and EDMA3TC) and the EDMA3 configuration registers. The PaRAM memory contents are undefined after device reset and you should not rely on parameters to be reset to a known state. The PaRAM entry must be initialized to a desired value before it is used.

### 11.3.17 Power Management

The EDMA3 (EDMA3CC and EDMA3TC) can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the power reset clock management (PRCM). The PRCM acts as a master controller for power management for all peripherals on the device.

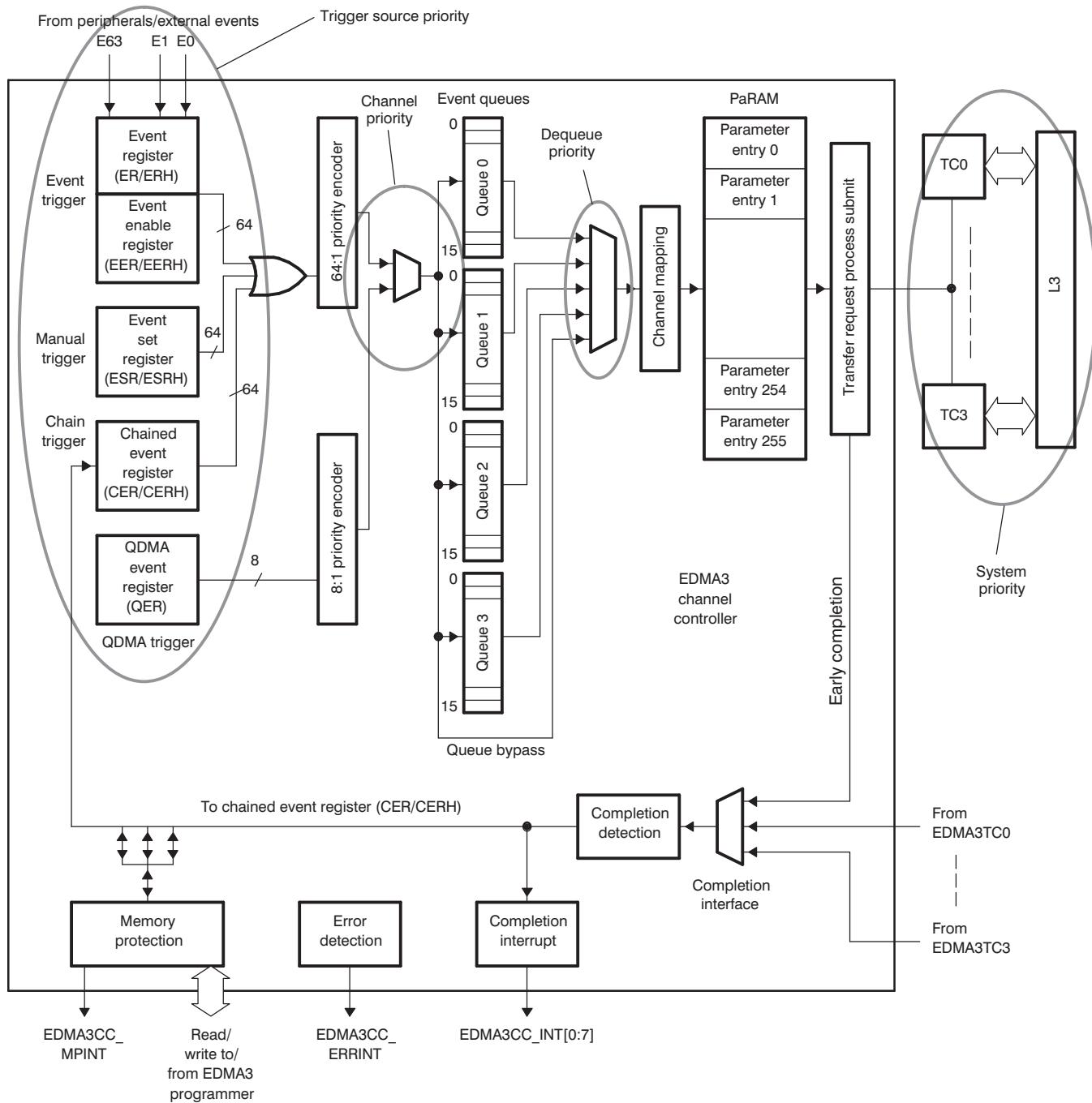
The EDMA3 controller can be idled on receiving a clock stop request from the PRCM. The requests to EDMA3CC and EDMA3TC are separate. In general, it should be verified that there are no pending activities in the EDMA3 controller.

### 11.3.18 Emulation Considerations

During debug when using the emulator, the CPU(s) may be halted on an execute packet boundary for single-stepping, benchmarking, profiling, or other debug purposes. During an emulation halt, the EDMA3 channel controller and transfer controller operations continue. Events continue to be latched and processed and transfer requests continue to be submitted and serviced.

Since EDMA3 is involved in servicing multiple master and slave peripherals, it is not feasible to have an independent behavior of the EDMA3 for emulation halts. EDMA3 functionality would be coupled with the peripherals it is servicing, which might have different behavior during emulation halts. For example, if a McASP is halted during an emulation access (FREE = 0 and SOFT = 0 or 1 in McASP registers), the McASP stops generating the McASP receive or transmit events (REVT or XEVT) to the EDMA. From the point of view of the McASP, the EDMA3 is suspended, but other peripherals (for example, a timer) still assert events and will be serviced by the EDMA.

**Figure 11-21. EDMA3 Prioritization**



### 11.3.19 EDMA Transfer Examples

The EDMA3 channel controller performs a variety of transfers depending on the parameter configuration. The following sections provide a description and PaRAM configuration for some typical use case scenarios.

#### 11.3.19.1 Block Move Example

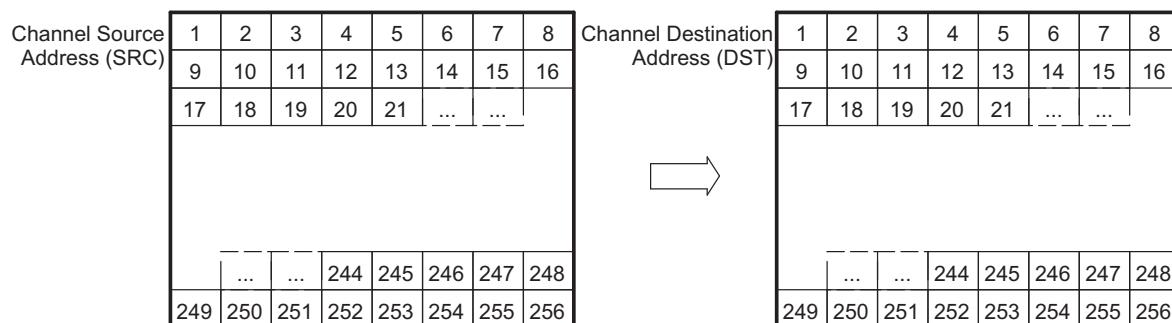
The most basic transfer performed by the EDMA3 is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external memory to internal L2 SRAM as shown in [Figure 11-22](#). [Figure 11-23](#) shows the parameters for this transfer.

The source address for the transfer is set to the start of the data block in external memory, and the destination address is set to the start of the data block in L2. If the data block is less than 64K bytes, the PaRAM configuration shown in [Figure 11-23](#) holds true with the synchronization type set to A-synchronized and indexes cleared to 0. If the amount of data is greater than 64K bytes, BCNT and the B-indexes need to be set appropriately with the synchronization type set to AB-synchronized. The STATIC bit in OPT is set to prevent linking.

This transfer example may also be set up using QDMA. For successive transfer submissions, of a similar nature, the number of cycles used to submit the transfer are fewer depending on the number of changing transfer parameters. You may program the QDMA trigger word to be the highest numbered offset in the PaRAM set that undergoes change.

**Figure 11-22. Block Move Example**



**Figure 11-23. Block Move Example PaRAM Configuration**

(a) EDMA Parameters

| Parameter Contents                |       | Parameter                         |  |
|-----------------------------------|-------|-----------------------------------|--|
| 0010 0008h                        |       | Channel Options Parameter (OPT)   |  |
| Channel Source Address (SRC)      |       | Channel Source Address (SRC)      |  |
| 0001h                             | 0100h | Count for 2nd Dimension (BCNT)    |  |
| Channel Destination Address (DST) |       | Count for 1st Dimension (ACNT)    |  |
| 0000h                             | 0000h | Channel Destination Address (DST) |  |
| 0000h                             | FFFFh | Destination BCNT Index (DSTBIDX)  |  |
| 0000h                             | 0000h | Source BCNT Index (SRCBIDX)       |  |
| 0000h                             | 0001h | BCNT Reload (BCNTRLD)             |  |
| Destination CCNT Index (DSTCIDX)  |       | Link Address (LINK)               |  |
| Reserved                          |       | Source CCNT Index (SRCCIDX)       |  |
|                                   |       | Count for 3rd Dimension (CCNT)    |  |

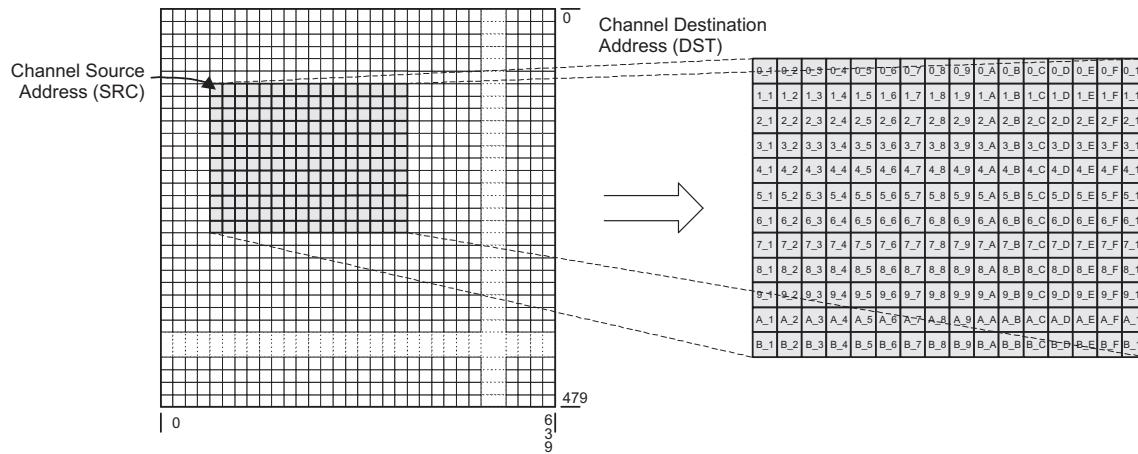
## (b) Channel Options Parameter (OPT) Content

|      |          |        |        |         |          |          |         |        |          |     |     |    |
|------|----------|--------|--------|---------|----------|----------|---------|--------|----------|-----|-----|----|
| 31   | 30       | 28     | 27     | 24      | 23       | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
| 0    | 000      |        | 0000   | 0       | 0        | 0        | 1       |        | 00       |     | 00  |    |
| PRIV | Reserved |        | PRIVID | ITCCHEN | TCCHEN   | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11     | 10     | 8       | 7        |          |         | 4      | 3        | 2   | 1   | 0  |
| 0000 |          | 0      | 000    |         | 0000     |          |         | 1      | 0        | 0   | 0   | 0  |
| TCC  |          | TCCMOD | FWID   |         | Reserved |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

**11.3.19.2 Subframe Extraction Example**

The EDMA3 can efficiently extract a small frame of data from a larger frame of data. By performing a 2D-to-1D transfer, the EDMA3 retrieves a portion of data for the CPU to process. In this example, a 640 × 480-pixel frame of video data is stored in external memory. Each pixel is represented by a 16-bit halfword. The CPU extracts a 16 × 12-pixel subframe of the image for processing. To facilitate more efficient processing time by the CPU, the EDMA3 places the subframe in internal L2 SRAM. [Figure 11-24](#) shows the transfer of a subframe from external memory to L2. [Figure 11-25](#) shows the parameters for this transfer.

The same PaRAM entry options are used for QDMA channels, as well as DMA channels. The STATIC bit in OPT is set to prevent linking. For successive transfers, only changed parameters need to be programmed before triggering the channel.

**Figure 11-24. Subframe Extraction Example****Figure 11-25. Subframe Extraction Example PaRAM Configuration**

## (a) EDMA Parameters

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 000Ch                        |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC)      |       | Channel Source Address (SRC)      |                                |
| 000Ch                             | 0020h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0020h                             | 0500h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0000h                             | FFFFh | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

## (b) Channel Options Parameter (OPT) Content

| 31   | 30       | 28   | 27     | 24       | 23     | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|------|--------|----------|--------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |      | 0000   | 0        | 0      | 0        | 1       |        | 00       |     | 00  |    |
| PRIV | Reserved |      | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11   | 10     | 8        | 7      |          |         | 4      | 3        | 2   | 1   | 0  |
| 0000 |          | 0    | 000    |          | 0000   |          |         | 1      | 1        | 0   | 0   |    |
| TCC  | TCCMOD   | FWID |        | Reserved |        |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

**11.3.19.3 Data Sorting Example**

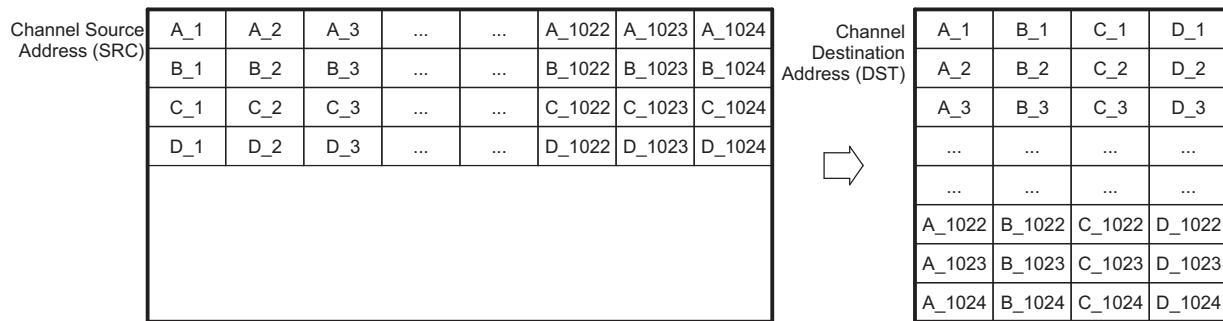
Many applications require the use of multiple data arrays; it is often desirable to have the arrays arranged such that the first elements of each array are adjacent, the second elements are adjacent, and so on. Often this is not how the data is presented to the device. Either data is transferred via a peripheral with the data arrays arriving one after the other or the arrays are located in memory with each array occupying a portion of contiguous memory spaces. For these instances, the EDMA3 can reorganize the data into the desired format. [Figure 11-26](#) shows the data sorting.

To determine the parameter set values, the following need to be considered:

- ACNT - Program this to be the size in bytes of an element.
- BCNT - Program this to be the number of elements in a frame.
- CCNT - Program this to be the number of frames.
- SRCBIDX - Program this to be the size of the element or ACNT.
- DSTBIDX - CCNT × ACNT
- SRCCDX - ACNT × BCNT
- DSTCIDX - ACNT

The synchronization type needs to be AB-synchronized and the STATIC bit is 0 to allow updates to the parameter set. It is advised to use normal EDMA3 channels for sorting.

It is not possible to sort this with a single trigger event. Instead, the channel can be programmed to be chained to itself. After BCNT elements get sorted, intermediate chaining could be used to trigger the channel again causing the transfer of the next BCNT elements and so on. [Figure 11-27](#) shows the parameter set programming for this transfer, assuming channel 0 and an element size of 4 bytes.

**Figure 11-26. Data Sorting Example**

**Figure 11-27. Data Sorting Example PaRAM Configuration**

(a) EDMA Parameters

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0090 0004h                        |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC)      |       | Channel Source Address (SRC)      |                                |
| 0400h                             | 0004h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0010h                             | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0000h                             | FFFFh | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0001h                             | 1000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0004h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) Channel Options Parameter (OPT) Content

| 31   | 30       | 28     | 27     | 24      | 23     | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|--------|--------|---------|--------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |        | 0000   |         | 1      | 0        | 0       | 1      | 00       |     | 00  |    |
| PRIV | Reserved |        | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11     | 10     | 8       | 7      |          |         | 4      | 3        | 2   | 1   | 0  |
| 0000 |          | 0      | 000    |         |        | 0000     |         | 0      | 1        | 0   | 0   |    |
| TCC  |          | TCCMOD | FWID   |         |        | Reserved |         | STATIC | SYNCDIM  | DAM | SAM |    |

#### 11.3.19.4 Peripheral Servicing Example

The EDMA3 channel controller also services peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the EDMA3 channels, they can be configured to continuously service on-chip and off-chip peripherals throughout the device operation. Each event available to the EDMA3 has its own dedicated channel, and all channels operate simultaneously. The only requirements are to use the proper channel for a particular transfer and to enable the channel event in the event enable register (EER). When programming an EDMA3 channel to service a peripheral, it is necessary to know how data is to be presented to the processor. Data is always provided with some kind of synchronization event as either one element per event (non-bursting) or multiple elements per event (bursting).

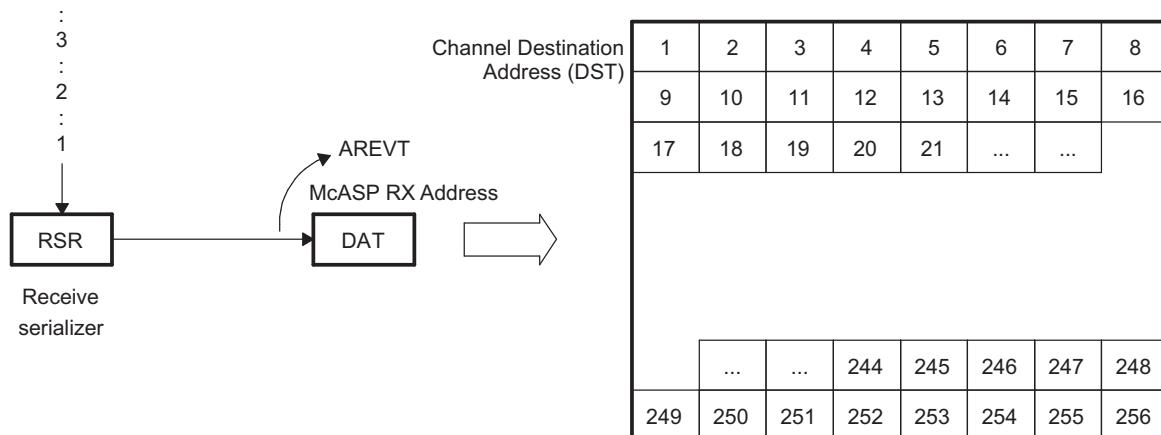
##### 11.3.19.4.1 Non-bursting Peripherals

Non-bursting peripherals include the on-chip multichannel audio serial port (McASP) and many external devices, such as codecs. Regardless of the peripheral, the EDMA3 channel configuration is the same.

The McASP transmit and receive data streams are treated independently by the EDMA3. The transmit and receive data streams can have completely different counts, data sizes, and formats. [Figure 11-28](#) shows servicing incoming McASP data.

To transfer the incoming data stream to its proper location in DDR memory, the EDMA3 channel must be set up for a 1D-to-1D transfer with A-synchronization. Because an event (AREVT) is generated for every word as it arrives, it is necessary to have the EDMA3 issue the transfer request for each element individually. [Figure 11-29](#) shows the parameters for this transfer. The source address of the EDMA3 channel is set to the data port address(DAT) for McASP, and the destination address is set to the start of the data block in DDR. Because the address of serializer buffer is fixed, the source B index is cleared to 0 (no modification) and the destination B index is set to 01b (increment).

Based on the premise that serial data is typically a high priority, the EDMA3 channel should be programmed to be on queue 0.

**Figure 11-28. Servicing Incoming McASP Data Example**

**Figure 11-29. Servicing Incoming McASP Data Example PaRAM Configuration**

(a) EDMA Parameters

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 0000h                        |       | Channel Options Parameter (OPT)   |                                |
| McASP RX Address                  |       | Channel Source Address (SRC)      |                                |
| 0100h                             | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0000h                             | FFFFh | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0004h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) Channel Options Parameter (OPT) Content

| 31   | 30       | 28   | 27     | 24      | 23     | 22       | 21      | 20     | 19       | 18 | 17  | 16  |
|------|----------|------|--------|---------|--------|----------|---------|--------|----------|----|-----|-----|
| 0    | 000      |      | 0000   | 0       | 0      | 0        | 1       | 00     |          | 00 |     | 00  |
| PRIV | Reserved |      | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |    | TCC |     |
| 15   | 12       | 11   | 10     | 8       | 7      |          |         | 4      | 3        | 2  | 1   | 0   |
| 0000 | 0        | 000  |        | 0000    |        | Reserved |         | 0      | 0        | 0  | DAM | SAM |
| TCC  | TCCMOD   | FWID |        |         |        |          |         | STATIC | SYNCDIM  |    |     |     |

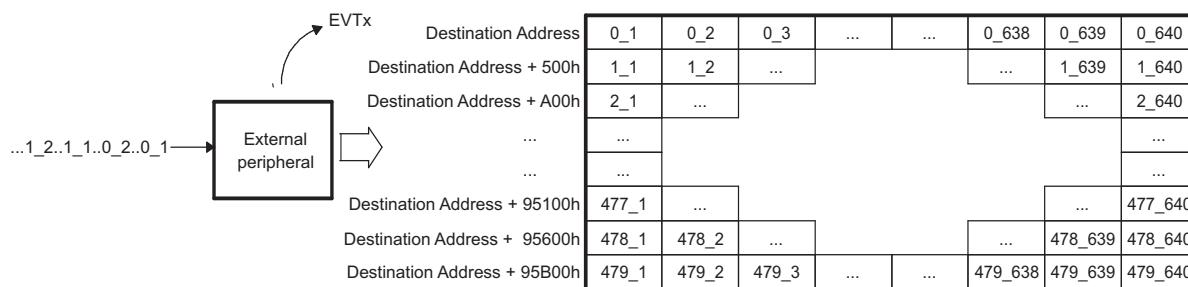
### 11.3.19.4.2 Bursting Peripherals

Higher bandwidth applications require that multiple data elements be presented to the processor core for every synchronization event. This frame of data can either be from multiple sources that are working simultaneously or from a single high-throughput peripheral that streams data to/from the processor.

In this example, a port is receiving a video frame from a camera and presenting it to the processor one array at a time. The video image is  $640 \times 480$  pixels, with each pixel represented by a 16-bit element. The image is to be stored in external memory. [Figure 11-30](#) shows this example.

To transfer data from an external peripheral to an external buffer one array at a time based on  $EVT_n$ , channel  $n$  must be configured. Due to the nature of the data (a video frame made up of arrays of pixels) the destination is essentially a 2D entity. [Figure 11-31](#) shows the parameters to service the incoming data with a 1D-to-2D transfer using AB-synchronization. The source address is set to the location of the video framer peripheral, and the destination address is set to the start of the data buffer. Because the input address is static, the SRCBIDX is 0 (no modification to the source address). The destination is made up of arrays of contiguous, linear elements; therefore, the DSTBIDX is set to pixel size, 2 bytes. ANCT is equal to the pixel size, 2 bytes. BCNT is set to the number of pixels in an array, 640. CCNT is equal to the total number of arrays in the block, 480. SRCCIDX is 0 because the source address undergoes no increment. The DSTCIDX is equal to the difference between the starting addresses of each array. Because a pixel is 16 bits (2 bytes), DSTCIDX is equal to  $640 \times 2$ .

**Figure 11-30. Servicing Peripheral Burst Example**



**Figure 11-31. Servicing Peripheral Burst Example PaRAM Configuration**

(a) EDMA Parameters

| Parameter Contents          |       | Parameter                         |                                |
|-----------------------------|-------|-----------------------------------|--------------------------------|
| 0010 0004h                  |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address      |       | Channel Source Address (SRC)      |                                |
| 0280h                       | 0002h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address |       | Channel Destination Address (DST) |                                |
| 0002h                       | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0000h                       | FFFFh | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0500h                       | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                       | 01E0h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) Channel Options Parameter (OPT) Content

| 31   | 30       | 28     | 27     | 24      | 23       | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|--------|--------|---------|----------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |        | 0000   | 0       | 0        | 0        | 1       | 00     |          | 00  |     | 00 |
| PRIV | Reserved |        | PRIVID | ITCCHEN | TCCHEN   | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11     | 10     | 8       | 7        |          |         | 4      | 3        | 2   | 1   | 0  |
| 0000 |          | 0      | 000    |         | 0000     |          |         | 0      | 1        | 0   | 0   |    |
| TCC  |          | TCCMOD | FWID   |         | Reserved |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

### 11.3.19.4.3 Continuous Operation

Configuring an EDMA3 channel to receive a single frame of data is useful, and is applicable to some systems. A majority of the time, however, data is going to be continuously transmitted and received throughout the entire operation of the processor. In this case, it is necessary to implement some form of linking such that the EDMA3 channels continuously reload the necessary parameter sets. In this example, McASP is configured to transmit and receive data on a T1 array. To simplify the example, only two channels are active for both transmit and receive data streams. Each channel receives packets of 128 elements. The packets are transferred from the serial port to internal memory and from internal memory to the serial port, as shown [Figure 11-32](#).

The McASP generates AREVT for every element received and generates AXEVT for every element transmitted. To service the data streams, the DMA channels associated with the McASP must be setup for 1D-to-1D transfers with A-synchronization.

[Figure 11-33](#) shows the parameter entries for the channel for these transfers. To service the McASP continuously, the channels must be linked to a duplicate PaRAM set in the PaRAM. After all frames have been transferred, the EDMA3 channels reload and continue. [Figure 11-34](#) shows the reload parameters for the channel.

#### 11.3.19.4.3.1 Receive Channel

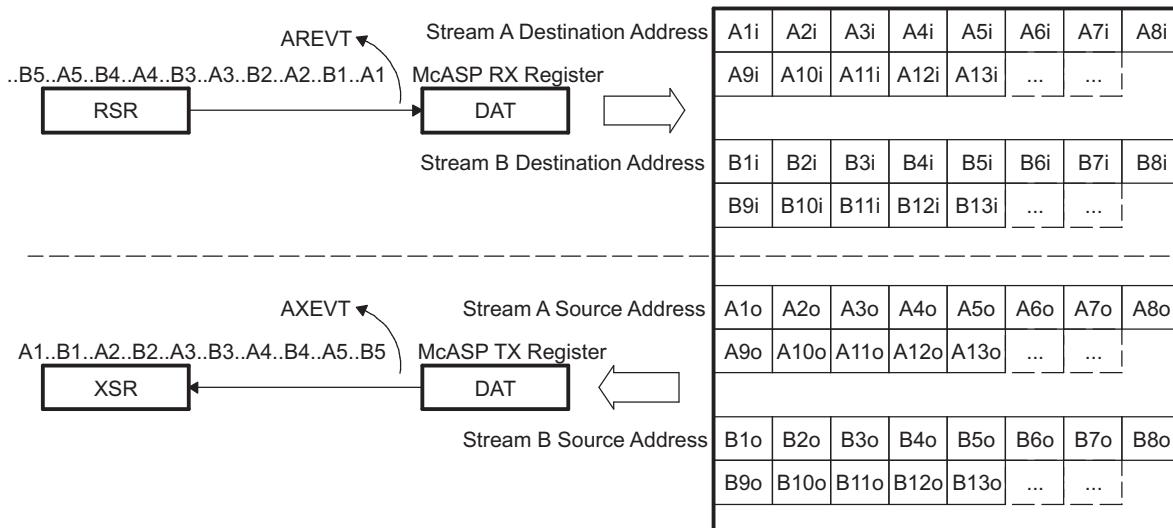
EDMA3 channel 15 services the incoming data stream of McASP. The source address is set to that of the receive serializer buffer, and the destination address is set to the first element of the data block. Because there are two data channels being serviced, A and B, they are to be located separately within the L2 SRAM.

To facilitate continuous operation, a copy of the PaRAM set for the channel is placed in PaRAM set 64. The LINK option is set and the link address is provided in the PaRAM set. Upon exhausting the channel 15 parameter set, the parameters located at the link address are loaded into the channel 15 parameter set and operation continues. This function continues throughout device operation until halted by the CPU.

#### 11.3.19.4.3.2 Transmit Channel

EDMA3 channel 12 services the outgoing data stream of McASP. In this case the destination address needs no update, hence, the parameter set changes accordingly. Linking is also used to allow continuous operation by the EDMA3 channel, with duplicate PaRAM set entries at PaRAM set 65.

**Figure 11-32. Servicing Continuous McASP Data Example**



**Figure 11-33. Servicing Continuous McASP Data Example PaRAM Configuration**

(a) EDMA Parameters for Receive Channel (PaRAM Set 15) being Linked to PaRAM Set 64

| Parameter Contents                |       |       |       |       |       |       |       | Parameter                         |  |  |  |                                |  |  |  |
|-----------------------------------|-------|-------|-------|-------|-------|-------|-------|-----------------------------------|--|--|--|--------------------------------|--|--|--|
| 0010 0000h                        |       |       |       |       |       |       |       | Channel Options Parameter (OPT)   |  |  |  |                                |  |  |  |
| McASP RX Register                 |       |       |       |       |       |       |       | Channel Source Address (SRC)      |  |  |  |                                |  |  |  |
| 0080h                             | 0001h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Count for 2nd Dimension (BCNT)    |  |  |  | Count for 1st Dimension (ACNT) |  |  |  |
| Channel Destination Address (DST) |       |       |       |       |       |       |       | Channel Destination Address (DST) |  |  |  |                                |  |  |  |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  |  |  |  | Source BCNT Index (SRCBIDX)    |  |  |  |
| 0080h                             | 4800h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | BCNT Reload (BCNTRLD)             |  |  |  | Link Address (LINK)            |  |  |  |
| 0000h                             | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Destination CCNT Index (DSTCIDX)  |  |  |  | Source CCNT Index (SRCCIDX)    |  |  |  |
| 0000h                             | FFFFh | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Reserved                          |  |  |  | Count for 3rd Dimension (CCNT) |  |  |  |

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 15)

| 31   | 30       | 28  | 27     | 24      | 23     | 22       | 21      | 20 | 19       | 18      | 17  | 16  |     |        |      |
|------|----------|-----|--------|---------|--------|----------|---------|----|----------|---------|-----|-----|-----|--------|------|
| 0    | 000      |     | 0000   | 0       | 0      | 0        | 1       | 00 | 00       | 00      | 00  | 00  |     |        |      |
| PRIV | Reserved |     | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN |    | Reserved |         |     |     | TCC |        |      |
| 15   | 12       | 11  | 10     | 8       | 7      |          |         | 4  | 3        | 2       | 1   | 0   |     |        |      |
| 0000 | 0        | 000 |        | 0000    |        |          |         | 0  | 0        | 0       | 0   | 0   | TCC | TCCMOD | FWID |
|      |          |     |        |         |        | Reserved |         |    | STATIC   | SYNCDIM | DAM | SAM |     |        |      |

(c) EDMA Parameters for Transmit Channel (PaRAM Set 12) being Linked to PaRAM Set 65

| Parameter Contents           |       |       |       |       |       |       |       | Parameter                         |  |  |  |                                |  |  |  |
|------------------------------|-------|-------|-------|-------|-------|-------|-------|-----------------------------------|--|--|--|--------------------------------|--|--|--|
| 0010 1000h                   |       |       |       |       |       |       |       | Channel Options Parameter (OPT)   |  |  |  |                                |  |  |  |
| Channel Source Address (SRC) |       |       |       |       |       |       |       | Channel Source Address (SRC)      |  |  |  |                                |  |  |  |
| 0080h                        | 0001h | 0000h | 0001h | 0000h | 0000h | 0000h | 0000h | Count for 2nd Dimension (BCNT)    |  |  |  | Count for 1st Dimension (ACNT) |  |  |  |
| McASP TX Register            |       |       |       |       |       |       |       | Channel Destination Address (DST) |  |  |  |                                |  |  |  |
| 0000h                        | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Destination BCNT Index (DSTBIDX)  |  |  |  | Source BCNT Index (SRCBIDX)    |  |  |  |
| 0080h                        | 4860h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | BCNT Reload (BCNTRLD)             |  |  |  | Link Address (LINK)            |  |  |  |
| 0000h                        | FFFFh | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Destination CCNT Index (DSTCIDX)  |  |  |  | Source CCNT Index (SRCCIDX)    |  |  |  |
| 0000h                        | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | 0000h | Reserved                          |  |  |  | Count for 3rd Dimension (CCNT) |  |  |  |

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 12)

| 31   | 30       | 28  | 27     | 24      | 23     | 22       | 21      | 20 | 19       | 18      | 17  | 16  |     |        |      |
|------|----------|-----|--------|---------|--------|----------|---------|----|----------|---------|-----|-----|-----|--------|------|
| 0    | 000      |     | 0000   | 0       | 0      | 0        | 1       | 00 | 00       | 00      | 00  | 00  |     |        |      |
| PRIV | Reserved |     | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN |    | Reserved |         |     |     | TCC |        |      |
| 15   | 12       | 11  | 10     | 8       | 7      |          |         | 4  | 3        | 2       | 1   | 0   |     |        |      |
| 0001 | 0        | 000 |        | 0000    |        |          |         | 0  | 0        | 0       | 0   | 0   | TCC | TCCMOD | FWID |
|      |          |     |        |         |        | Reserved |         |    | STATIC   | SYNCDIM | DAM | SAM |     |        |      |

**Figure 11-34. Servicing Continuous McASP Data Example Reload PaRAM Configuration**

(a) EDMA Reload Parameters (PaRAM Set 64) for Receive Channel

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 0000h                        |       | Channel Options Parameter (OPT)   |                                |
| McASP RX Register                 |       | Channel Source Address (SRC)      |                                |
| 0080h                             | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                             | 4800h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | FFFFh | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 64)

| 31   | 30       | 28   | 27     | 24       | 23     | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|------|--------|----------|--------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |      | 0000   | 0        | 0      | 0        | 1       | 00     |          | 00  |     | 00 |
| PRIV | Reserved |      | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11   | 10     | 8        | 7      |          |         | 4      | 3        | 2   | 1   | 0  |
| 0000 |          | 0    | 000    |          | 0000   |          |         | 0      | 0        | 0   | 0   | 0  |
| TCC  | TCCMOD   | FWID |        | Reserved |        |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

(c) EDMA Reload Parameters (PaRAM Set 65) for Transmit Channel

| Parameter Contents           |       | Parameter                         |                                |
|------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 1000h                   |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC) |       | Channel Source Address (SRC)      |                                |
| 0080h                        | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| McASP TX Register            |       | Channel Destination Address (DST) |                                |
| 0000h                        | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                        | 4860h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                        | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                        | FFFFh | Reserved                          | Count for 3rd Dimension (CCNT) |

(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 65)

| 31   | 30       | 28   | 27     | 24       | 23     | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|------|--------|----------|--------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |      | 0000   | 0        | 0      | 0        | 1       | 00     |          | 00  |     | 00 |
| PRIV | Reserved |      | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11   | 10     | 8        | 7      |          |         | 4      | 3        | 2   | 1   | 0  |
| 0001 |          | 0    | 000    |          | 0000   |          |         | 0      | 0        | 0   | 0   | 0  |
| TCC  | TCCMOD   | FWID |        | Reserved |        |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

#### 11.3.19.4.4 Ping-Pong Buffering

Although the previous configuration allows the EDMA3 to service a peripheral continuously, it presents a number of restrictions to the CPU. Because the input and output buffers are continuously being filled/emptied, the CPU must match the pace of the EDMA3 very closely to process the data. The EDMA3 receive data must always be placed in memory before the CPU accesses it, and the CPU must provide the output data before the EDMA3 transfers it. Though not impossible, this is an unnecessary challenge. It is particularly difficult in a 2-level cache scheme.

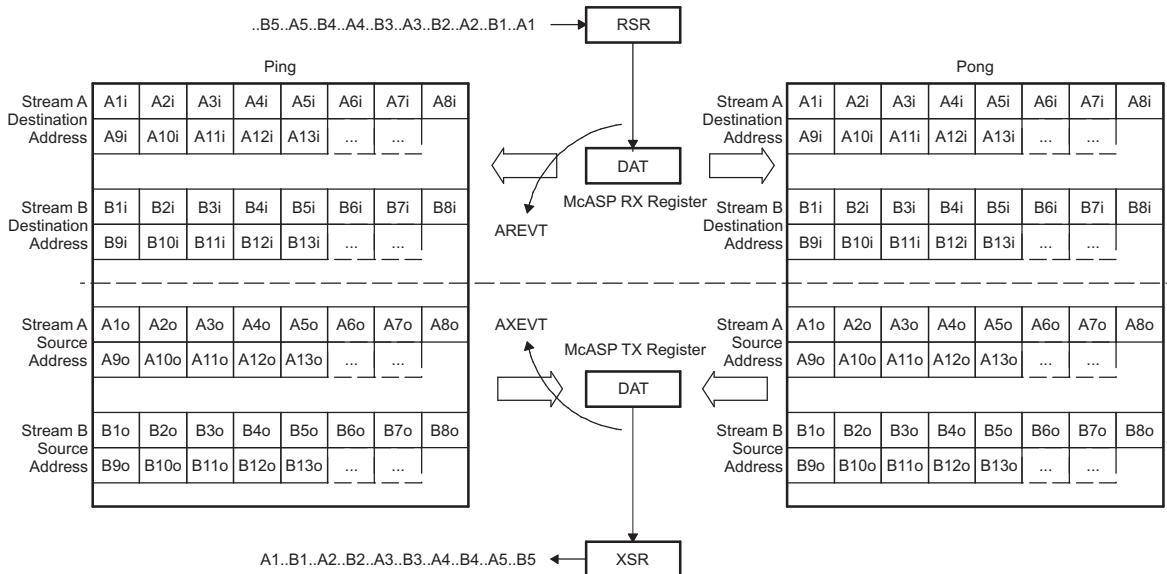
Ping-pong buffering is a simple technique that allows the CPU activity to be distanced from the EDMA3 activity. This means that there are multiple (usually two) sets of data buffers for all incoming and outgoing data streams. While the EDMA3 transfers the data into and out of the ping buffers, the CPU manipulates the data in the pong buffers. When both CPU and EDMA3 activity completes, they switch. The EDMA3 then writes over the old input data and transfers the new output data. [Figure 11-35](#) shows the ping-pong scheme for this example.

To change the continuous operation example, such that a ping-pong buffering scheme is used, the EDMA3 channels need only a moderate change. Instead of one parameter set, there are two; one for transferring data to/from the ping buffers and one for transferring data to/from the pong buffers. As soon as one transfer completes, the channel loads the PaRAM set for the other and the data transfers continue. [Figure 11-36](#) shows the EDMA3 channel configuration required.

Each channel has two parameter sets, ping and pong. The EDMA3 channel is initially loaded with the ping parameters ([Figure 11-36](#)). The link address for the ping set is set to the PaRAM offset of the pong parameter set ([Figure 11-37](#)). The link address for the pong set is set to the PaRAM offset of the ping parameter set ([Figure 11-38](#)). The channel options, count values, and index values are all identical between the ping and pong parameters for each channel. The only differences are the link address provided and the address of the data buffer.

#### 11.3.19.4.4.1 Synchronization with the CPU

To utilize the ping-pong buffering technique, the system must signal the CPU when to begin to access the new data set. After the CPU finishes processing an input buffer (ping), it waits for the EDMA3 to complete before switching to the alternate (pong) buffer. In this example, both channels provide their channel numbers as their report word and set the TCINTEN bit to generate an interrupt after completion. When channel 15 fills an input buffer, the E15 bit in the interrupt pending register (IPR) is set; when channel 12 empties an output buffer, the E12 bit in IPR is set. The CPU must manually clear these bits. With the channel parameters set, the CPU polls IPR to determine when to switch. The EDMA3 and CPU could alternatively be configured such that the channel completion interrupts the CPU. By doing this, the CPU could service a background task while waiting for the EDMA3 to complete.

**Figure 11-35. Ping-Pong Buffering for McASP Data Example**

**Figure 11-36. Ping-Pong Buffering for McASP Example PaRAM Configuration**

(a) EDMA Parameters for Channel 15 (Using PaRAM Set 15 Linked to Pong Set 64)

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 D000h                        |       | Channel Options Parameter (OPT)   |                                |
| McASP RX Register                 |       | Channel Source Address (SRC)      |                                |
| 0080h                             | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                             | 4800h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) Channel Options Parameter (OPT) Content for Channel 15

| 31   | 30       | 28   | 27     | 24       | 23     | 22       | 21      | 20     | 19       | 18  | 17  | 16 |
|------|----------|------|--------|----------|--------|----------|---------|--------|----------|-----|-----|----|
| 0    | 000      |      | 0000   |          | 0      | 0        | 0       | 1      | 00       |     | 00  |    |
| PRIV | Reserved |      | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |        | Reserved |     | TCC |    |
| 15   | 12       | 11   | 10     | 8        | 7      |          |         | 4      | 3        | 2   | 1   | 0  |
| 1101 |          | 0    | 000    |          | 0000   |          |         | 0      | 0        | 0   | 0   | 0  |
| TCC  | TCCMOD   | FWID |        | Reserved |        |          |         | STATIC | SYNCDIM  | DAM | SAM |    |

(c) EDMA Parameters for Channel 12 (Using PaRAM Set 12 Linked to Pong Set 66)

| Parameter Contents           |       | Parameter                         |                                |
|------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 C000h                   |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC) |       | Channel Source Address (SRC)      |                                |
| 0080h                        | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| McASP TX Register            |       | Channel Destination Address (DST) |                                |
| 0000h                        | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                        | 4840h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                        | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                        | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

(d) Channel Options Parameter (OPT) Content for Channel 12

| 31   | 30       | 28   | 27     | 24       | 23     | 22       | 21      | 20 | 19       | 18 | 17  | 16  |
|------|----------|------|--------|----------|--------|----------|---------|----|----------|----|-----|-----|
| 0    | 000      |      | 0000   | 0        | 0      | 0        | 1       | 00 |          | 00 |     | 00  |
| PRIV | Reserved |      | PRIVID | ITCCHEN  | TCCHEN | ITCINTEN | TCINTEN |    | Reserved |    | TCC |     |
| 15   | 12       | 11   | 10     | 8        | 7      |          |         | 4  | 3        | 2  | 1   | 0   |
| 1100 | 0        | 000  |        | 0000     |        | 0000     |         | 0  | 0        | 0  | DAM | SAM |
| TCC  | TCCMOD   | FWID |        | Reserved |        | STATIC   | SYNCDIM |    |          |    |     |     |

**Figure 11-37. Ping-Pong Buffering for McASP Example Pong PaRAM Configuration**

(a) EDMA Pong Parameters for Channel 15 at Set 64 Linked to Set 65

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 D000h                        |       | Channel Options Parameter (OPT)   |                                |
| McASP RX Register                 |       | Channel Source Address (SRC)      |                                |
| 0080h                             | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                             | 4820h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) EDMA Pong Parameters for Channel 12 at Set 66 Linked to Set 67

| Parameter Contents           |       | Parameter                         |                                |
|------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 C000h                   |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC) |       | Channel Source Address (SRC)      |                                |
| 0080h                        | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| McASP TX Register            |       | Channel Destination Address (DST) |                                |
| 0000h                        | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                        | 4860h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                        | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                        | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

**Figure 11-38. Ping-Pong Buffering for McASP Example Ping PaRAM Configuration**

(a) EDMA Ping Parameters for Channel 15 at Set 65 Linked to Set 64

| Parameter Contents                |       | Parameter                         |                                |
|-----------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 D000h                        |       | Channel Options Parameter (OPT)   |                                |
| McASP RX Register                 |       | Channel Source Address (SRC)      |                                |
| 0080h                             | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| Channel Destination Address (DST) |       | Channel Destination Address (DST) |                                |
| 0001h                             | 0000h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                             | 4800h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                             | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                             | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

(b) EDMA Ping Parameters for Channel 12 at Set 67 Linked to Set 66

| Parameter Contents           |       | Parameter                         |                                |
|------------------------------|-------|-----------------------------------|--------------------------------|
| 0010 C000h                   |       | Channel Options Parameter (OPT)   |                                |
| Channel Source Address (SRC) |       | Channel Source Address (SRC)      |                                |
| 0080h                        | 0001h | Count for 2nd Dimension (BCNT)    | Count for 1st Dimension (ACNT) |
| McASP TX Register            |       | Channel Destination Address (DST) |                                |
| 0000h                        | 0001h | Destination BCNT Index (DSTBIDX)  | Source BCNT Index (SRCBIDX)    |
| 0080h                        | 4840h | BCNT Reload (BCNTRLD)             | Link Address (LINK)            |
| 0000h                        | 0000h | Destination CCNT Index (DSTCIDX)  | Source CCNT Index (SRCCIDX)    |
| 0000h                        | 0001h | Reserved                          | Count for 3rd Dimension (CCNT) |

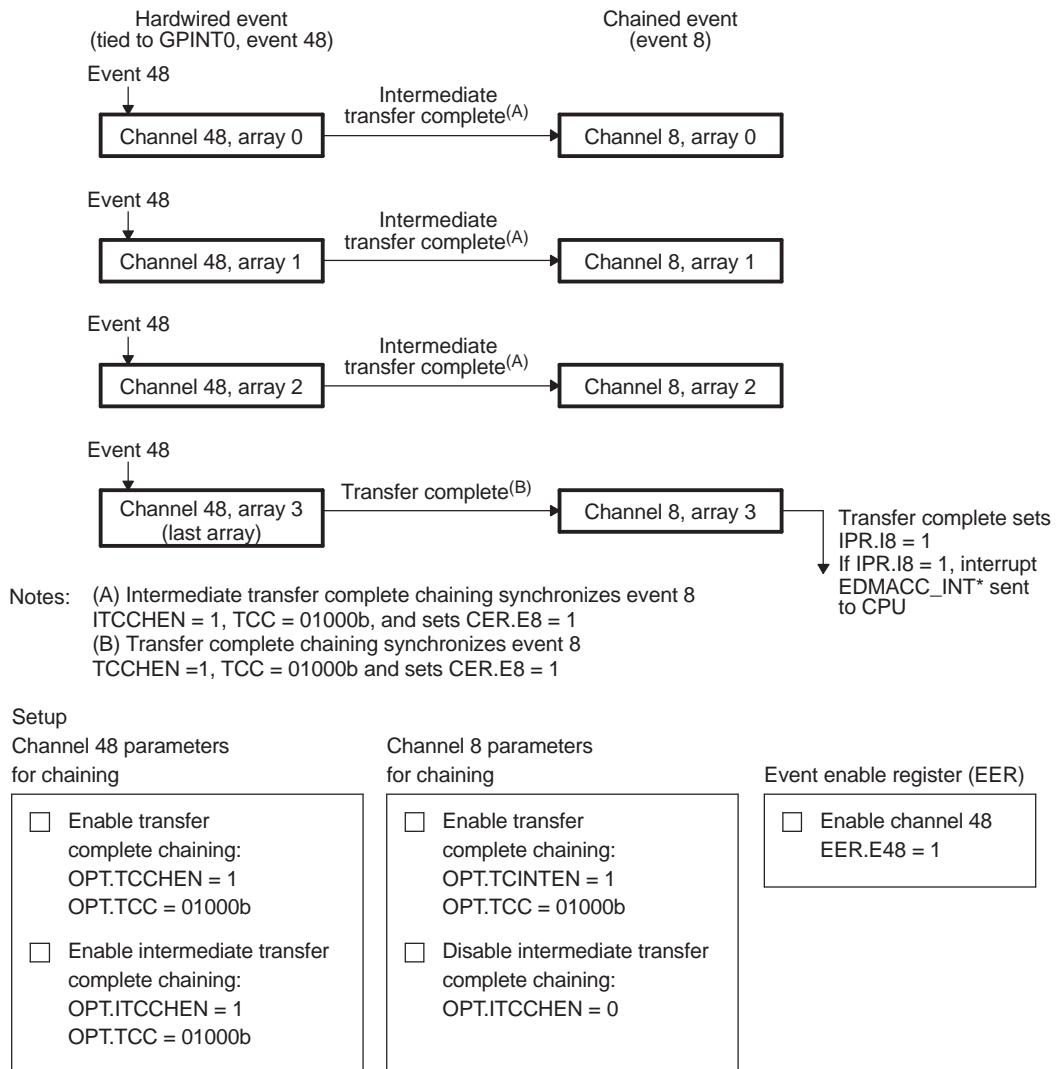
#### 11.3.19.4.5 Transfer Chaining Examples

The following examples explain the intermediate transfer complete chaining function.

##### 11.3.19.4.5.1 Servicing Input/Output FIFOs with a Single Event

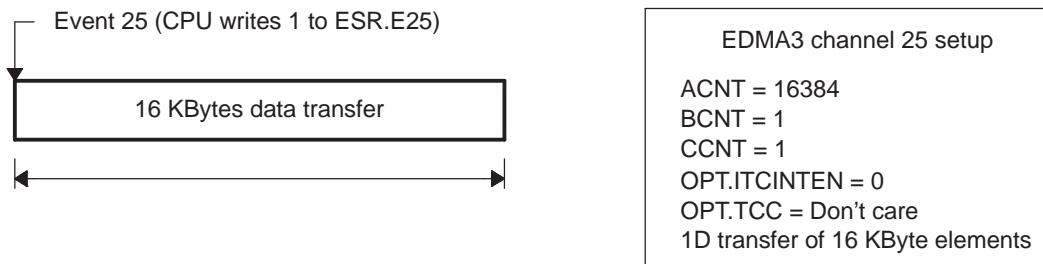
Many systems require the use of a pair of external FIFOs that must be serviced at the same rate. One FIFO buffers data input, and the other buffers data output. The EDMA3 channels that service these FIFOs can be set up for AB-synchronized transfers. While each FIFO is serviced with a different set of parameters, both can be signaled from a single event. For example, an external interrupt pin can be tied to the status flags of one of the FIFOs. When this event arrives, the EDMA3 needs to perform servicing for both the input and output streams. Without the intermediate transfer complete chaining feature this would require two events, and thus two external interrupt pins. The intermediate transfer complete chaining feature allows the use of a single external event (for example, a GPIO event). [Figure 11-39](#) shows the EDMA3 setup and illustration for this example.

A GPIO event (in this case, GPINT0) triggers an array transfer. Upon completion of each intermediate array transfer of channel 48, intermediate transfer complete chaining sets the E8 bit (specified by TCC of 8) in the chained event register (CER) and provides a synchronization event to channel 8. Upon completion of the last array transfer of channel 48, transfer complete chaining—not intermediate transfer complete chaining—sets the E8 bit in CER (specified by TCCMODE:TCC) and provides a synchronization event to channel 8. The completion of channel 8 sets the I8 bit (specified by TCCMODE:TCC) in the interrupt pending register (IPR), which can generate an interrupt to the CPU, if the I8 bit in the interrupt enable register (IER) is set.

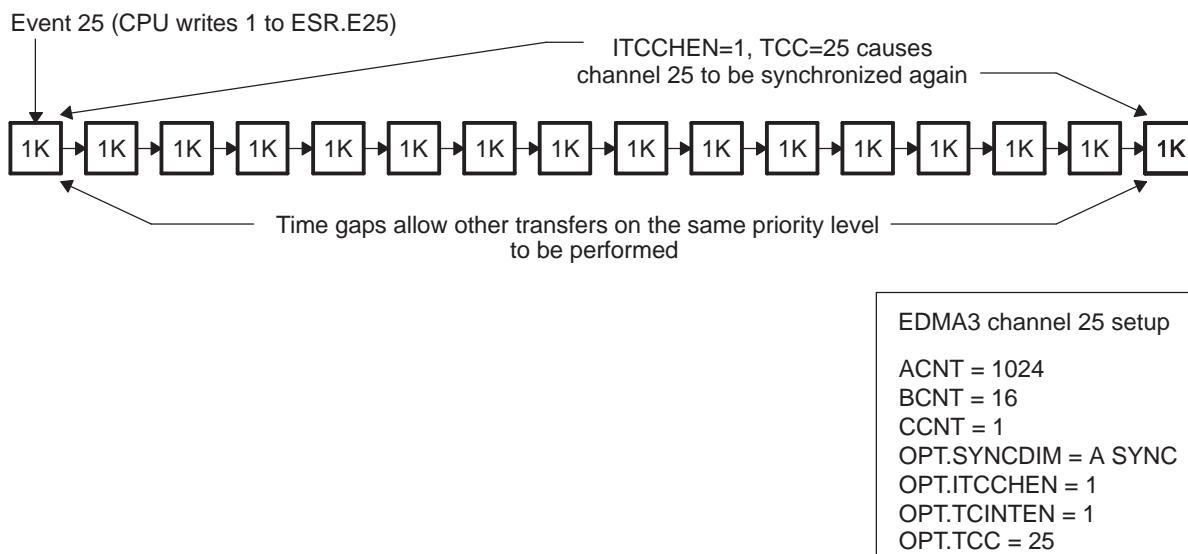
**Figure 11-39. Intermediate Transfer Completion Chaining Example**


#### 11.3.19.4.5.2 Breaking Up Large Transfers with Intermediate Chaining

Another feature of intermediate transfer chaining (ITCCHEN) is for breaking up large transfers. A large transfer may lock out other transfers of the same priority level for the duration of the transfer. For example, a large transfer on queue 0 from the internal memory to the external memory using the EMIF may starve other EDMA3 transfers on the same queue. In addition, this large high-priority transfer may prevent the EMIF for a long duration to service other lower priority transfers. When a large transfer is considered to be high priority, it should be split into multiple smaller transfers. [Figure 11-40](#) shows the EDMA3 setup and illustration of an example single large block transfer.

**Figure 11-40. Single Large Block Transfer Example**


The intermediate transfer chaining enable (ITCCHEN) provides a method to break up a large transfer into smaller transfers. For example, to move a single large block of memory (16K bytes), the EDMA3 performs an A-synchronized transfer. The element count is set to a reasonable value, where reasonable derives from the amount of time it would take to move this smaller amount of data. Assume 1 Kbyte is a reasonable small transfer in this example. The EDMA3 is set up to transfer 16 arrays of 1 Kbyte elements, for a total of 16K byte elements. The TCC field in the channel options parameter (OPT) is set to the same value as the channel number and ITCCHEN are set. In this example, EDMA3 channel 25 is used and TCC is also set to 25. The TCINTEN may also be set to trigger interrupt 25 when the last 1 Kbyte array is transferred. The CPU starts the EDMA3 transfer by writing to the appropriate bit of the event set register (ESR.E25). The EDMA3 transfers the first 1 Kbyte array. Upon completion of the first array, intermediate transfer complete code chaining generates a synchronization event to channel 25, a value specified by the TCC field. This intermediate transfer completion chaining event causes EDMA3 channel 25 to transfer the next 1 Kbyte array. This process continues until the transfer parameters are exhausted, at which point the EDMA3 has completed the 16K byte transfer. This method breaks up a large transfer into smaller packets, thus providing natural time slices in the transfer such that other events may be processed. [Figure 11-41](#) shows the EDMA3 setup and illustration of the broken up smaller packet transfers.

**Figure 11-41. Smaller Packet Data Transfers Example**


### 11.3.20 EDMA Events

**Table 11-23. Direct Mapped**

| Event Number | Event Name                 | Source Module |
|--------------|----------------------------|---------------|
| 0            | pr1_host[7] <sup>(1)</sup> | PRU-ICSS      |
| 1            | pr1_host[6] <sup>(1)</sup> | PRU-ICSS      |
| 2            | SDTXEVT1                   | MMCHS1        |
| 3            | SDRXEVT1                   | MMCHS1        |
| 4            | Reserved                   | Reserved      |
| 5            | Reserved                   | Reserved      |
| 6            | Reserved                   | Reserved      |
| 7            | Reserved                   | Reserved      |
| 8            | AXEVT0                     | McASP0        |
| 9            | AREVT0                     | McASP0        |
| 10           | AXEVT1                     | McASP1        |
| 11           | AREVT1                     | McASP1        |
| 12           | Open                       | Open          |
| 13           | Open                       | Open          |
| 14           | ePWMEVT0                   | ePWM 0        |
| 15           | ePWMEVT1                   | ePWM 1        |
| 16           | SPIXEVT0                   | McSPI0        |
| 17           | SPIREVT0                   | McSPI0        |
| 18           | SPIXEVT1                   | McSPI0        |
| 19           | SPIREVT1                   | McSPI0        |
| 20           | Open                       | Open          |
| 21           | Open                       | Open          |
| 22           | GPIOEVT0                   | GPIO0         |
| 23           | GPIOEVT1                   | GPIO1         |
| 24           | SDTXEVT0                   | MMCHS0        |
| 25           | SDRXEVT0                   | MMCHS0        |
| 26           | UTXEVTO                    | UART0         |
| 27           | URXEVTO                    | UART0         |
| 28           | UTXEVET1                   | UART1         |
| 29           | URXEVET1                   | UART1         |
| 30           | UTXEVET2                   | UART2         |
| 31           | URXEVET2                   | UART2         |
| 32           | Open                       | Open          |
| 33           | Open                       | Open          |
| 34           | Open                       | Open          |
| 35           | Open                       | Open          |
| 36           | Open                       | Open          |
| 37           | Open                       | Open          |
| 38           | eCAPEVT0                   | eCAP 0        |
| 39           | eCAPEVT1                   | eCAP 1        |
| 40           | CAN_IF1DMA                 | DCAN 0        |
| 41           | CAN_IF2DMA                 | DCAN 0        |
| 42           | SPIXEVT0                   | McSPI1        |
| 43           | SPIREVT0                   | McSPI1        |
| 44           | SPIXEVT1                   | McSPI1        |

<sup>(1)</sup> pr1\_host\_intr[0:7] corresponds to Host-2 to Host-9 of the PRU-ICSS interrupt controller.

**Table 11-23. Direct Mapped (continued)**

| Event Number | Event Name    | Source Module |
|--------------|---------------|---------------|
| 45           | SPIREVT1      | McSPI1        |
| 46           | eQEPEVT0      | eQEP 0        |
| 47           | CAN_IF3DMA    | DCAN 0        |
| 48           | TINT4         | Timer 4       |
| 49           | TINT5         | Timer 5       |
| 50           | TINT6         | Timer 6       |
| 51           | TINT7         | Timer 7       |
| 52           | GPMCEVT       | GPMC          |
| 53           | tsc_adc_FIFO0 | ADC/TSC       |
| 54           | Open          |               |
| 55           | Open          |               |
| 56           | eQEPEVT1      | eQEP 1        |
| 57           | tsc_adc_FIFO1 | ADC/TSC       |
| 58           | I2CTXEVT0     | I2C0          |
| 59           | I2CRXEVT0     | I2C0          |
| 60           | I2CTXEVT1     | I2C1          |
| 61           | I2CRXEVT1     | I2C1          |
| 62           | eCAPEVT2      | eCAP 2        |
| 63           | eHRPWM2       | eHRPWM 2      |

**Table 11-24. Crossbar Mapped**

| Event Number | Event Name | Source Module |
|--------------|------------|---------------|
| 1            | SDTXEVT2   | MMCHS2        |
| 2            | SDRXEVT2   | MMCHS2        |
| 3            | I2CTXEVT2  | I2C2          |
| 4            | I2CRXEVT2  | I2C2          |
| 5            | Open       | Open          |
| 6            | Open       | Open          |
| 7            | UTXEV3     | UART3         |
| 8            | URXEV3     | UART3         |
| 9            | UTXEV4     | UART4         |
| 10           | URXEV4     | UART4         |
| 11           | UTXEV5     | UART5         |
| 12           | URXEV5     | UART5         |
| 13           | CAN_IF1DMA | DCAN 1        |
| 14           | CAN_IF2DMA | DCAN 1        |
| 15           | CAN_IF3DMA | DCAN 1        |
| 16           | Open       | Open          |
| 17           | Open       | Open          |
| 18           | Open       | Open          |
| 19           | Open       | Open          |
| 20           | Open       | Open          |
| 21           | Open       | Open          |
| 22           | TINT0      | Timer 0       |
| 23           |            |               |
| 24           | TINT2      | Timer 2       |
| 25           | TINT3      | Timer 3       |

**Table 11-24. Crossbar Mapped (continued)**

| Event Number | Event Name           | Source Module                   |
|--------------|----------------------|---------------------------------|
| 26           | Open                 | Open                            |
| 27           | Open                 | Open                            |
| 28           | pi_x_dma_event_intr0 | External pin (XDMA_EVENT_INTR0) |
| 29           | pi_x_dma_event_intr1 | External pin (XDMA_EVENT_INTR1) |
| 30           | pi_x_dma_event_intr2 | External pin (XDMA_EVENT_INTR2) |
| 31           | eQEPEVT2             | eQEP 2                          |
| 32           | GPIOEVT2             | GPIO2                           |
| 33           | Open                 |                                 |
| 34           | Open                 |                                 |
| 35           | Open                 |                                 |
| 36           | Open                 |                                 |
| 37           | Open                 |                                 |
| 38           | Open                 |                                 |
| 39           | Open                 |                                 |
| 40           | Open                 |                                 |
| 41           | Open                 |                                 |
| 42           | Open                 |                                 |
| 43           | Open                 |                                 |
| 44           | Open                 |                                 |
| 45           | Open                 |                                 |
| 46           | Open                 |                                 |
| 47           | Open                 |                                 |
| 48           | Open                 |                                 |
| 49           | Open                 |                                 |
| 50           | Open                 |                                 |
| 51           | Open                 |                                 |
| 52           | Open                 |                                 |
| 53           | Open                 |                                 |
| 54           | Open                 |                                 |
| 55           | Open                 |                                 |
| 56           | Open                 |                                 |
| 57           | Open                 |                                 |
| 58           | Open                 |                                 |
| 59           | Open                 |                                 |
| 60           | Open                 |                                 |
| 61           | Open                 |                                 |
| 62           | Open                 |                                 |
| 63           | Open                 |                                 |

## 11.4 EDMA3 Registers

### 11.4.1 EDMA3 Channel Controller Registers

Table 11-25 lists the memory-mapped registers for the EDMA3 channel controller (EDMACC). All other register offset addresses not listed in Table 11-25 should be considered as reserved locations and the register contents should not be modified.

**Table 11-25. EDMACC Registers**

| Offset      | Acronym    | Register Description                                | Section                            |
|-------------|------------|---|------------------------------------|
| 00h         | PID        | Peripheral Identification Register                  | <a href="#">Section 11.4.1.1.1</a> |
| 04h         | CCCFG      | EDMA3CC Configuration Register                      | <a href="#">Section 11.4.1.1.2</a> |
| 10h         | SYSCONFIG  | EDMA3CC System Configuration Register               | <a href="#">Section 11.4.1.1.3</a> |
| 0100h-01FCh | DCHMAP0-63 | DMA Channel 0-63 Mapping Registers                  | <a href="#">Section 11.4.1.1.4</a> |
| 0200h       | QCHMAP0    | QDMA Channel 0 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0204h       | QCHMAP1    | QDMA Channel 1 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0208h       | QCHMAP2    | QDMA Channel 2 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 020Ch       | QCHMAP3    | QDMA Channel 3 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0210h       | QCHMAP4    | QDMA Channel 4 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0214h       | QCHMAP5    | QDMA Channel 5 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0218h       | QCHMAP6    | QDMA Channel 6 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 021Ch       | QCHMAP7    | QDMA Channel 7 Mapping Register                     | <a href="#">Section 11.4.1.1.5</a> |
| 0240h       | DMAQNUM0   | DMA Queue Number Register 0                         | <a href="#">Section 11.4.1.1.6</a> |
| 0244h       | DMAQNUM1   | DMA Queue Number Register 1                         | <a href="#">Section 11.4.1.1.6</a> |
| 0248h       | DMAQNUM2   | DMA Queue Number Register 2                         | <a href="#">Section 11.4.1.1.6</a> |
| 024Ch       | DMAQNUM3   | DMA Queue Number Register 3                         | <a href="#">Section 11.4.1.1.6</a> |
| 0250h       | DMAQNUM4   | DMA Queue Number Register 4                         | <a href="#">Section 11.4.1.1.6</a> |
| 0254h       | DMAQNUM5   | DMA Queue Number Register 5                         | <a href="#">Section 11.4.1.1.6</a> |
| 0258h       | DMAQNUM6   | DMA Queue Number Register 6                         | <a href="#">Section 11.4.1.1.6</a> |
| 025Ch       | DMAQNUM7   | DMA Queue Number Register 7                         | <a href="#">Section 11.4.1.1.6</a> |
| 0260h       | QDMAQNUM   | QDMA Queue Number Register                          | <a href="#">Section 11.4.1.1.7</a> |
| 0284h       | QUEPRI     | Queue Priority Register                             | <a href="#">Section 11.4.1.1.8</a> |
| 0300h       | EMR        | Event Missed Register                               | <a href="#">Section 11.4.1.2.1</a> |
| 0304h       | EMRH       | Event Missed Register High                          | <a href="#">Section 11.4.1.2.1</a> |
| 0308h       | EMCR       | Event Missed Clear Register                         | <a href="#">Section 11.4.1.2.2</a> |
| 030Ch       | EMCRH      | Event Missed Clear Register High                    | <a href="#">Section 11.4.1.2.2</a> |
| 0310h       | QEMR       | QDMA Event Missed Register                          | <a href="#">Section 11.4.1.2.3</a> |
| 0314h       | QEMCR      | QDMA Event Missed Clear Register                    | <a href="#">Section 11.4.1.2.4</a> |
| 0318h       | CCERR      | EDMA3CC Error Register                              | <a href="#">Section 11.4.1.2.5</a> |
| 031Ch       | CCERRCLR   | EDMA3CC Error Clear Register                        | <a href="#">Section 11.4.1.2.6</a> |
| 0320h       | EEVAL      | Error Evaluate Register                             | <a href="#">Section 11.4.1.2.7</a> |
| 0340h       | DRAE0      | DMA Region Access Enable Register for Region 0      | <a href="#">Section 11.4.1.3.1</a> |
| 0344h       | DRAEH0     | DMA Region Access Enable Register High for Region 0 | <a href="#">Section 11.4.1.3.1</a> |
| 0348h       | DRAE1      | DMA Region Access Enable Register for Region 1      | <a href="#">Section 11.4.1.3.1</a> |
| 034Ch       | DRAEH1     | DMA Region Access Enable Register High for Region 1 | <a href="#">Section 11.4.1.3.1</a> |
| 0350h       | DRAE2      | DMA Region Access Enable Register for Region 2      | <a href="#">Section 11.4.1.3.1</a> |
| 0354h       | DRAEH2     | DMA Region Access Enable Register High for Region 2 | <a href="#">Section 11.4.1.3.1</a> |
| 0358h       | DRAE3      | DMA Region Access Enable Register for Region 3      | <a href="#">Section 11.4.1.3.1</a> |
| 035Ch       | DRAEH3     | DMA Region Access Enable Register High for Region 3 | <a href="#">Section 11.4.1.3.1</a> |
| 0360h       | DRAE4      | DMA Region Access Enable Register for Region 4      | <a href="#">Section 11.4.1.3.1</a> |
| 0364h       | DRAEH4     | DMA Region Access Enable Register High for Region 4 | <a href="#">Section 11.4.1.3.1</a> |

**Table 11-25. EDMACC Registers (continued)**

| Offset      | Acronym    | Register Description                                | Section                            |
|-------------|------------|---|------------------------------------|
| 0368h       | DRAE5      | DMA Region Access Enable Register for Region 5      | <a href="#">Section 11.4.1.3.1</a> |
| 036Ch       | DRAEH5     | DMA Region Access Enable Register High for Region 5 | <a href="#">Section 11.4.1.3.1</a> |
| 0370h       | DRAE6      | DMA Region Access Enable Register for Region 6      | <a href="#">Section 11.4.1.3.1</a> |
| 0374h       | DRAEH6     | DMA Region Access Enable Register High for Region 6 | <a href="#">Section 11.4.1.3.1</a> |
| 0378h       | DRAE7      | DMA Region Access Enable Register for Region 7      | <a href="#">Section 11.4.1.3.1</a> |
| 037Ch       | DRAEH7     | DMA Region Access Enable Register High for Region 7 | <a href="#">Section 11.4.1.3.1</a> |
| 0380h-039Ch | QRAE0-7    | QDMA Region Access Enable Registers for Region 0-7  | <a href="#">Section 11.4.1.3.2</a> |
| 0400h-04FCh | Q0E0-Q3E15 | Event Queue Entry Registers Q0E0-Q3E15              | <a href="#">Section 11.4.1.4.1</a> |
| 0600h-060Ch | QSTAT0-3   | Queue Status Registers 0-3                          | <a href="#">Section 11.4.1.4.2</a> |
| 0620h       | QWMTHRA    | Queue Watermark Threshold A Register                | <a href="#">Section 11.4.1.4.3</a> |
| 0640h       | CCSTAT     | EDMA3CC Status Register                             | <a href="#">Section 11.4.1.4.4</a> |
| 0800h       | MPFAR      | Memory Protection Fault Address Register            | <a href="#">Section 11.4.1.5.1</a> |
| 0804h       | MPFSR      | Memory Protection Fault Status Register             | <a href="#">Section 11.4.1.5.2</a> |
| 0808h       | MPFCR      | Memory Protection Fault Command Register            | <a href="#">Section 11.4.1.5.3</a> |
| 080Ch       | MPPAG      | Memory Protection Page Attribute Register Global    | <a href="#">Section 11.4.1.5.4</a> |
| 0810h-082Ch | MPPA0-7    | Memory Protection Page Attribute Registers 0-7      | <a href="#">Section 11.4.1.5.4</a> |
| 1000h       | ER         | Event Register                                      | <a href="#">Section 11.4.1.6.1</a> |
| 1004h       | ERH        | Event Register High                                 | <a href="#">Section 11.4.1.6.1</a> |
| 1008h       | ECR        | Event Clear Register                                | <a href="#">Section 11.4.1.6.2</a> |
| 100Ch       | ECRH       | Event Clear Register High                           | <a href="#">Section 11.4.1.6.2</a> |
| 1010h       | ESR        | Event Set Register                                  | <a href="#">Section 11.4.1.6.3</a> |
| 1014h       | ESRH       | Event Set Register High                             | <a href="#">Section 11.4.1.6.3</a> |
| 1018h       | CER        | Chained Event Register                              | <a href="#">Section 11.4.1.6.4</a> |
| 101Ch       | CERH       | Chained Event Register High                         | <a href="#">Section 11.4.1.6.4</a> |
| 1020h       | EER        | Event Enable Register                               | <a href="#">Section 11.4.1.6.5</a> |
| 1024h       | EERH       | Event Enable Register High                          | <a href="#">Section 11.4.1.6.5</a> |
| 1028h       | EECR       | Event Enable Clear Register                         | <a href="#">Section 11.4.1.6.6</a> |
| 102Ch       | EECRH      | Event Enable Clear Register High                    | <a href="#">Section 11.4.1.6.6</a> |
| 1030h       | EESR       | Event Enable Set Register                           | <a href="#">Section 11.4.1.6.7</a> |
| 1034h       | EESRH      | Event Enable Set Register High                      | <a href="#">Section 11.4.1.6.7</a> |
| 1038h       | SER        | Secondary Event Register                            | <a href="#">Section 11.4.1.6.8</a> |
| 103Ch       | SERH       | Secondary Event Register High                       | <a href="#">Section 11.4.1.6.8</a> |
| 1040h       | SECR       | Secondary Event Clear Register                      | <a href="#">Section 11.4.1.6.9</a> |
| 1044h       | SECRH      | Secondary Event Clear Register High                 | <a href="#">Section 11.4.1.6.9</a> |
| 1050h       | IER        | Interrupt Enable Register                           | <a href="#">Section 11.4.1.7.1</a> |
| 1054h       | IERH       | Interrupt Enable Register High                      | <a href="#">Section 11.4.1.7.1</a> |
| 1058h       | IECR       | Interrupt Enable Clear Register                     | <a href="#">Section 11.4.1.7.2</a> |
| 105Ch       | IECRH      | Interrupt Enable Clear Register High                | <a href="#">Section 11.4.1.7.2</a> |
| 1060h       | IESR       | Interrupt Enable Set Register                       | <a href="#">Section 11.4.1.7.3</a> |
| 1064h       | IESRH      | Interrupt Enable Set Register High                  | <a href="#">Section 11.4.1.7.3</a> |
| 1068h       | IPR        | Interrupt Pending Register                          | <a href="#">Section 11.4.1.7.4</a> |
| 106Ch       | IPRH       | Interrupt Pending Register High                     | <a href="#">Section 11.4.1.7.4</a> |
| 1070h       | ICR        | Interrupt Clear Register                            | <a href="#">Section 11.4.1.7.5</a> |
| 1074h       | ICRH       | Interrupt Clear Register High                       | <a href="#">Section 11.4.1.7.5</a> |
| 1078h       | IEVAL      | Interrupt Evaluate Register                         | <a href="#">Section 11.4.1.7.6</a> |
| 1080h       | QER        | QDMA Event Register                                 | <a href="#">Section 11.4.1.8.1</a> |
| 1084h       | QEER       | QDMA Event Enable Register                          | <a href="#">Section 11.4.1.8.2</a> |

**Table 11-25. EDMACC Registers (continued)**

| <b>Offset</b>                            | <b>Acronym</b> | <b>Register Description</b>             | <b>Section</b>                     |
|--|----------------|---|------------------------------------|
| 1088h                                    | QEECR          | QDMA Event Enable Clear Register        | <a href="#">Section 11.4.1.8.3</a> |
| 108Ch                                    | QEESR          | QDMA Event Enable Set Register          | <a href="#">Section 11.4.1.8.4</a> |
| 1090h                                    | QSER           | QDMA Secondary Event Register           | <a href="#">Section 11.4.1.8.5</a> |
| 1094h                                    | QSECR          | QDMA Secondary Event Clear Register     | <a href="#">Section 11.4.1.8.6</a> |
| <b>Shadow Region 0 Channel Registers</b> |                |   |                                    |
| 2000h                                    | ER             | Event Register                          |                                    |
| 2004h                                    | ERH            | Event Register High                     |                                    |
| 2008h                                    | ECR            | Event Clear Register                    |                                    |
| 200Ch                                    | ECRH           | Event Clear Register High               |                                    |
| 2010h                                    | ESR            | Event Set Register                      |                                    |
| 2014h                                    | ESRH           | Event Set Register High                 |                                    |
| 2018h                                    | CER            | Chained Event Register                  |                                    |
| 201Ch                                    | CERH           | Chained Event Register High             |                                    |
| 2020h                                    | EER            | Event Enable Register                   |                                    |
| 2024h                                    | EERH           | Event Enable Register High              |                                    |
| 2028h                                    | EECR           | Event Enable Clear Register             |                                    |
| 202Ch                                    | EECRH          | Event Enable Clear Register High        |                                    |
| 2030h                                    | EESR           | Event Enable Set Register               |                                    |
| 2034h                                    | EESRH          | Event Enable Set Register High          |                                    |
| 2038h                                    | SER            | Secondary Event Register                |                                    |
| 203Ch                                    | SERH           | Secondary Event Register High           |                                    |
| 2040h                                    | SECR           | Secondary Event Clear Register          |                                    |
| 2044h                                    | SECRH          | Secondary Event Clear Register High     |                                    |
| 2050h                                    | IER            | Interrupt Enable Register               |                                    |
| 2054h                                    | IERH           | Interrupt Enable Register High          |                                    |
| 2058h                                    | IECR           | Interrupt Enable Clear Register         |                                    |
| 205Ch                                    | IECRH          | Interrupt Enable Clear Register High    |                                    |
| 2060h                                    | IESR           | Interrupt Enable Set Register           |                                    |
| 2064h                                    | IESRH          | Interrupt Enable Set Register High      |                                    |
| 2068h                                    | IPR            | Interrupt Pending Register              |                                    |
| 206Ch                                    | IPRH           | Interrupt Pending Register High         |                                    |
| 2070h                                    | ICR            | Interrupt Clear Register                |                                    |
| 2074h                                    | ICRH           | Interrupt Clear Register High           |                                    |
| 2078h                                    | IEVAL          | Interrupt Evaluate Register             |                                    |
| 2080h                                    | QER            | QDMA Event Register                     |                                    |
| 2084h                                    | QEER           | QDMA Event Enable Register              |                                    |
| 2088h                                    | QEECR          | QDMA Event Enable Clear Register        |                                    |
| 208Ch                                    | QEESR          | QDMA Event Enable Set Register          |                                    |
| 2090h                                    | QSER           | QDMA Secondary Event Register           |                                    |
| 2094h                                    | QSECR          | QDMA Secondary Event Clear Register     |                                    |
| 2200h-2294h                              | -              | Shadow Region 1 Channel Registers       |                                    |
| 2400h-2494h                              | -              | Shadow Region 2 Channel Registers       |                                    |
| ...                                      |                | ...                                     |                                    |
| 2E00h-2E94h                              | -              | Shadow Channel Registers for MP Space 7 |                                    |

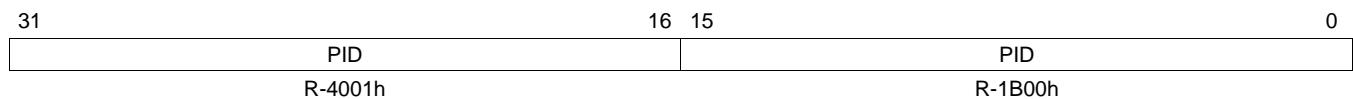
### 11.4.1.1 Global Registers

#### 11.4.1.1.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC.

The PID is shown in [Figure 11-42](#) and described in [Table 11-26](#).

**Figure 11-42. Peripheral ID Register (PID)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-26. Peripheral ID Register (PID) Field Descriptions**

| Bit  | Field | Value        | Description   |
|------|-------|--------------|---|
| 31-0 | PID   | 0xFFFF FFFFh | Peripheral identifier uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC. |

#### 11.4.1.1.2 EDMA3CC Configuration Register (CCCFG)

The EDMA3CC configuration register (CCCFG) provides the features/resources for the EDMA3CC in a particular device.

The CCCFG is shown in [Figure 11-43](#) and described in [Table 11-27](#).

**Figure 11-43. EDMA3CC Configuration Register (CCCFG)**

|          |             |          |    |          |           |     |     |    |
|----------|-------------|----------|----|----------|-----------|-----|-----|----|
| 31       | Reserved    |          |    |          |           | 26  | 25  | 24 |
|          | R-x         |          |    |          |           | R-1 | R-1 |    |
| 23       | 22          | 21       | 20 | 19       | 18        |     |     | 16 |
| Reserved |             | NUM_REGN |    | Reserved | NUM_EVQUE |     |     |    |
| R-0      |             | R-2h     |    | R-x      | R-2h      |     |     |    |
| 15       | 14          |          | 12 | 11       | 10        |     |     | 8  |
| Reserved | NUM_PAENTRY |          |    | Reserved | NUM_INTCH |     |     |    |
|          | R-x         |          |    | R-x      | R-4h      |     |     |    |
| 7        | 6           |          | 4  | 3        | 2         |     |     | 0  |
| Reserved | NUM_QDMACH  |          |    | Reserved | NUM_DMACH |     |     |    |
|          | R-x         |          |    | R-x      | R-5h      |     |     |    |

LEGEND: R = Read only; -n = value after reset; -x = value is indeterminate after reset

**Table 11-27. EDMA3CC Configuration Register (CCCFG) Field Descriptions**

| Bit   | Field       | Value | Description                                    |
|-------|-------------|-------|--|
| 31-26 | Reserved    | 0     | Reserved.                                      |
| 25    | MP_EXIST    | 0     | Memory protection existence.                   |
|       |             | 1     | Reserved.<br>Memory protection logic included. |
| 24    | CHMAP_EXIST | 0     | Channel mapping existence.                     |
|       |             | 1     | Reserved.<br>Channel mapping logic included.   |
| 23-22 | Reserved    | 0     | Reserved.                                      |
| 21-20 | NUM_REGN    | 0-3h  | Number of MP and shadow regions.               |
|       |             | 0-1h  | Reserved.                                      |
|       |             | 2h    | 4 regions.                                     |
|       |             | 3h    | Reserved.                                      |
| 19    | Reserved    | 0     | Reserved.                                      |
| 18-16 | NUM_EVQUE   | 0-7h  | Number of queues/number of TCs.                |
|       |             | 0-1h  | Reserved.                                      |
|       |             | 2h    | 3 EDMA3TCs/Event Queues                        |
|       |             | 3h-7h | Reserved.                                      |
| 15    | Reserved    | 0     | Reserved.                                      |
| 14-12 | NUM_PAENTRY | 0-7h  | Number of PaRAM sets.                          |
|       |             | 0-3h  | Reserved.                                      |
|       |             | 4h    | 256 PaRAM sets.                                |
|       |             | 5h-7h | Reserved.                                      |
| 11    | Reserved    | 0     | Reserved.                                      |

**Table 11-27. EDMA3CC Configuration Register (CCCFG) Field Descriptions (continued)**

| Bit  | Field      | Value                       | Description   |
|------|------------|-----------------------------|---|
| 10-8 | NUM_INTCH  | 0-7h<br>0-3h<br>4h<br>5h-7h | Number of interrupt channels.<br>Reserved.<br>64 interrupt channels.<br>Reserved. |
| 7    | Reserved   | 0                           | Reserved.   |
| 6-4  | NUM_QDMACH | 0-7h<br>0-3h<br>4h<br>5h-7h | Number of QDMA channels.<br>Reserved.<br>8 QDMA channels.<br>Reserved.            |
| 3    | Reserved   | 0                           | Reserved.   |
| 2-0  | NUM_DMACH  | 0-7h<br>0-4h<br>5h<br>6h-7h | Number of DMA channels.<br>Reserved.<br>64 DMA channels.<br>Reserved.             |

#### 11.4.1.1.3 EDMA3CC System Configuration Register (SYSCONFIG)

The EDMA3CC system configuration register is used for clock management configuration. The register is shown in [Figure 11-44](#) and described in [Table 11-28](#).

**Figure 11-44. EDMA3CC System Configuration Register (SYSCONFIG)**

|    |          |     |   |             |          |         |           |       |
|----|----------|-----|---|-------------|----------|---------|-----------|-------|
| 31 |          |     |   |             |          |         |           | 16    |
|    | Reserved |     |   |             |          |         |           |       |
|    | R-0      |     |   |             |          |         |           |       |
| 15 | Reserved | 6   | 5 | 4           | 3        | 2       | 1         | 0     |
|    |          |     |   | STANDBYMODE | IDLEMODE | FREEEMU | SOFTRESET |       |
|    |          | R-0 |   | R/W-2       | R/W-2    | R/W-0   |           | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

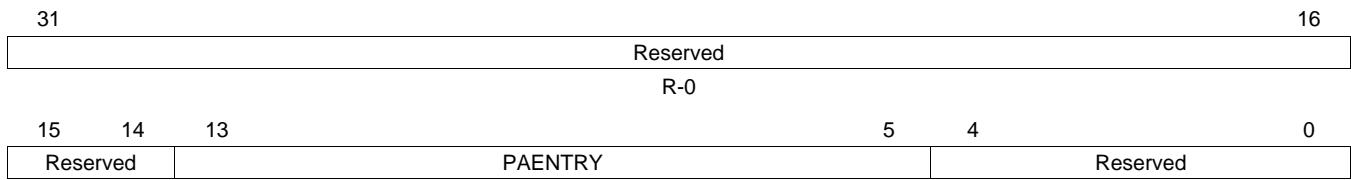
**Table 11-28. EDMA3CC System Configuration Register (SYSCONFIG) Field Descriptions**

| Bit  | Field       | Value | Description   |
|------|-------------|-------|---|
| 31-6 | Reserved    | 0     | Read returns 0.   |
| 5-4  | STANDBYMODE |       | Configuration of the local initiator state management mode. By definition, initiator may generate read/write transaction as long as it is out of STANDBY state.<br>0x0 = Force-standby mode: local initiator is unconditionally placed in standby state. Backup mode, for debug only.<br>0x1 = No-standby mode: local initiator is unconditionally placed out of standby state. Backup mode, for debug only.<br>0x2 = Smart-standby mode: local initiator standby status depends on local conditions, i.e., the module's functional requirement from the initiator. IP module should not generate (initiator-related) wakeup events.<br>0x3 = Smart-Standby wakeup-capable mode: local initiator standby status depends on local conditions, i.e., the module's functional requirement from the initiator. IP module may generate (master-related) wakeup events when in standby state. Mode is only relevant if the appropriate IP module "mwakeup" output is implemented. |
| 3-2  | IDLEMODE    |       | Configuration of the local target state management mode. By definition, target can handle read/write transaction as long as it is out of IDLE state.<br>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only.<br>0x1 = No-idle mode: local target never enters idle state. Backup mode, for debug only.<br>0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events.<br>0x3 = Reserved.   |
| 1    | FREEEMU     |       | Sensitivity to emulation (debug) suspend input signal.<br>0 = IP module is sensitive to emulation suspend.<br>1 = IP module is not sensitive to emulation suspend.  |
| 0    | SOFTRESET   | 0     | Software reset (optional)   |

#### 11.4.1.1.4 DMA Channel Map *n* Registers (DCHMAP*n*)

The DMA channel map *n* register (DCHMAP*n*) is shown in [Figure 11-45](#) and described in [Table 11-29](#).

**Figure 11-45. DMA Channel Map *n* Registers (DCHMAP*n*)**



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 11-29. DMA Channel Map *n* Registers (DCHMAP*n*) Field Descriptions**

| Bit   | Field    | Value  | Description   |
|-------|----------|--------|---|
| 31-14 | Reserved | 0      | Reserved  |
| 13-5  | PAENTRY  | 0-1FFh | Points to the PaRAM set number for DMA channel <i>n</i> . |
| 4-0   | Reserved | 0      | Reserved  |

#### **11.4.1.1.5 QDMA Channel Map n Registers (QCHMAPn)**

Each QDMA channel in EDMA3CC can be associated with any PaRAM set available on the device. Furthermore, the specific trigger word (0-7) of the PaRAM set can be programmed. The PaRAM set association and trigger word for every QDMA channel register is configurable using the QDMA channel map  $n$  register (QCHMAP $n$ ).

The QCHMAP $n$  is shown in Figure 11-46 and described in Table 11-30.

**NOTE:** At reset the QDMA channel map registers for all QDMA channels point to PaRAM set 0. If an application makes use of both a DMA channel that points to PaRAM set 0 and any QDMA channels, ensure that  $\text{QCHMAP}_n$  is programmed appropriately to point to a different PaRAM entry.

**Figure 11-46. QDMA Channel Map  $n$  Registers (QCHMAP $n$ )**

|          |          |    |  |        |   |   |          |   |     |
|----------|----------|----|--|--------|---|---|----------|---|-----|
| 31       | Reserved |    |  |        |   |   |          |   | 16  |
|          | R-0      |    |  |        |   |   |          |   |     |
| 15       | 14       | 13 |  |        | 5 | 4 |          | 2 | 1 0 |
| Reserved | PAENTRY  |    |  | TRWORD |   |   | Reserved |   |     |
| R-0      | R/W-0    |    |  | R/W-0  |   |   | R-0      |   |     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-30. QDMA Channel Map  $n$  Registers (QCHMAP $n$ ) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b>                 | <b>Description</b>  |
|------------|--------------|------------------------------|---|
| 31-14      | Reserved     | 0                            | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 13-5       | PAENTRY      | 0-1FFh<br>0-FFh<br>100h-1FFh | PAENTRY points to the PaRAM set number for QDMA channel <i>n</i> .<br>Parameter entry 0 through 255.<br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 4-2        | TRWORD       | 0-7h                         | Points to the specific trigger word of the PaRAM set defined by PAENTRY. A write to the trigger word results in a QDMA event being recognized.  |
| 1-0        | Reserved     | 0                            | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |

#### 11.4.1.1.6 DMA Channel Queue *n* Number Registers (DMAQNUM*n*)

The DMA channel queue number register (DMAQNUM*n*) allows programmability of each of the 64 DMA channels in the EDMA3CC to submit its associated synchronization event to any event queue in the EDMA3CC. At reset, all channels point to event queue 0.

The DMAQNUM*n* is shown in [Figure 11-47](#) and described in [Table 11-31](#). [Table 11-32](#) shows the channels and their corresponding bits in DMAQNUM*n*.

---

**NOTE:** Because the event queues in EDMA3CC have a fixed association to the transfer controllers, that is, Q0 TRs are submitted to TC0, Q1 TRs are submitted to TC1, etc., by programming DMAQNUM*n* for a particular DMA channel *n* also dictates which transfer controller is utilized for the data movement (or which EDMA3TC receives the TR request).

---

**Figure 11-47. DMA Channel Queue *n* Number Registers (DMAQNUM*n*)**

| 31   | 30 | 28    | 27   | 26 | 24    | 23   | 22 | 20    | 19   | 18 | 16    |
|------|----|-------|------|----|-------|------|----|-------|------|----|-------|
| Rsvd |    | En    |
| R-0  |    | R/W-0 |
| 15   | 14 | 12    | 11   | 10 | 8     | 7    | 6  | 4     | 3    | 2  | 0     |
| Rsvd |    | En    |
| R-0  |    | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 11-31. DMA Channel Queue *n* Number Registers (DMAQNUM*n*) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0-7h  | DMA queue number. Contains the event queue number to be used for the corresponding DMA channel. Programming DMAQNUM <i>n</i> for an event queue number to a value more than the number of queues available in the EDMA3CC results in undefined behavior. |
|      |       | 0     | Event <i>n</i> is queued on Q0.  |
|      |       | 1h    | Event <i>n</i> is queued on Q1.  |
|      |       | 2h    | Event <i>n</i> is queued on Q2.  |
|      |       | 3h    | Event <i>n</i> is queued on Q3.  |
|      |       | 4h-7h | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |

**Table 11-32. Bits in DMAQNUM*n***

| Channel Number (DMAQNUM <i>n</i> ) |    |     |     |     |     |     |     |
|------------------------------------|----|-----|-----|-----|-----|-----|-----|
| En bit                             | 0  | 1   | 2   | 3   | 4   | 5   | 6   |
| 0-2                                | E0 | E8  | E16 | E24 | E32 | E40 | E48 |
| 4-6                                | E1 | E9  | E17 | E25 | E33 | E41 | E49 |
| 8-10                               | E2 | E10 | E18 | E26 | E34 | E42 | E50 |
| 12-14                              | E3 | E11 | E19 | E27 | E35 | E43 | E51 |
| 16-18                              | E4 | E12 | E20 | E28 | E36 | E44 | E52 |
| 20-22                              | E5 | E13 | E21 | E29 | E37 | E45 | E53 |
| 24-26                              | E6 | E14 | E22 | E30 | E38 | E46 | E54 |
| 28-30                              | E7 | E15 | E23 | E31 | E39 | E47 | E55 |
|                                    |    |     |     |     |     |     | E63 |

#### 11.4.1.1.7 QDMA Channel Queue Number Register (QDMAQNUM)

The QDMA channel queue number register (QDMAQNUM) is used to program all the QDMA channels in the EDMA3CC to submit the associated QDMA event to any of the event queues in the EDMA3CC.

The QDMAQNUM is shown in [Figure 11-48](#) and described in [Table 11-33](#).

**Figure 11-48. QDMA Channel Queue Number Register (QDMAQNUM)**

| 31   | 30    | 28   | 27 | 26    | 24   | 23 | 22    | 20   | 19 | 18    | 16 |
|------|-------|------|----|-------|------|----|-------|------|----|-------|----|
| Rsvd | E7    | Rsvd |    | E6    | Rsvd |    | E5    | Rsvd |    | E4    |    |
| R-0  | R/W-0 | R-0  |    | R/W-0 | R-0  |    | R/W-0 | R-0  |    | R/W-0 |    |
| 15   | 14    | 12   | 11 | 10    | 8    | 7  | 6     | 4    | 3  | 2     | 0  |
| Rsvd | E3    | Rsvd |    | E2    | Rsvd |    | E1    | Rsvd |    | E0    |    |
| R-0  | R/W-0 | R-0  |    | R/W-0 | R-0  |    | R/W-0 | R-0  |    | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-33. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-15 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 14-0  | $E_n$    | 0-7h  | QDMA queue number. Contains the event queue number to be used for the corresponding QDMA channel.<br><br>0 Event $n$ is queued on Q0.<br>1h Event $n$ is queued on Q1.<br>2h Event $n$ is queued on Q2.<br>3h Event $n$ is queued on Q3.<br><br>4h-7h Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |

#### 11.4.1.1.8 Queue Priority Register (QUEPRI)

The queue priority register (QUEPRI) allows you to change the priority of the individual queues and the priority of the transfer request (TR) associated with the events queued in the queue. Because the queue to EDMA3TC mapping is fixed, programming QUEPRI essentially governs the priority of the associated transfer controller(s) read/write commands with respect to the other bus masters in the device. You can modify the EDMA3TC priority to obtain the desired system performance.

The QUEPRI is shown in [Figure 11-49](#) and described in [Table 11-34](#).

**Figure 11-49. Queue Priority Register (QUEPRI)**

| 31   | Reserved |      |       |      |       |      |       |   |   |   | 16 |       |
|------|----------|------|-------|------|-------|------|-------|---|---|---|----|-------|
|      |          |      |       |      |       |      |       |   |   |   |    | R-0   |
| 15   | 14       | 12   | 11    | 10   | 8     | 7    | 6     | 4 | 3 | 2 | 0  |       |
| Rsvd | PRIQ3    | Rsvd | PRIQ2 | Rsvd | PRIQ1 | Rsvd | PRIQ0 |   |   |   |    |       |
| R-0  | R/W-0    | R-0  | R/W-0 |      | R/W-0 |      | R-0   |   |   |   |    | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-34. Queue Priority Register (QUEPRI) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-15 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 14-12 | PRIQ3    | 0-7h  | Priority level for queue 3. Dictates the priority level used by TC3 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |
| 11    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 10-8  | PRIQ2    | 0-7h  | Priority level for queue 2. Dictates the priority level used by TC2 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |
| 7     | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 6-4   | PRIQ1    | 0-7h  | Priority level for queue 1. Dictates the priority level used by TC1 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |
| 3     | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 2-0   | PRIQ0    | 0-7h  | Priority level for queue 0. Dictates the priority level used by TC0 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |

### 11.4.1.2 Error Registers

The EDMA3CC contains a set of registers that provide information on missed DMA and/or QDMA events, and instances when event queue thresholds are exceeded. If any of the bits in these registers is set, it results in the EDMA3CC generating an error interrupt.

#### 11.4.1.2.1 Event Missed Registers (EMR/EMRH)

For a particular DMA channel, if a second event is received prior to the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the event missed registers (EMR/EMRH). All trigger types are treated individually, that is, manual triggered (ESR/ESRH), chain triggered (CER/CERH), and event triggered (ER/ERH) are all treated separately. The EMR/EMRH bits for a channel are also set if an event on that channel encounters a NULL entry (or a NULL TR is serviced). If any EMR/EMRH bit is set (and all errors, including bits in other error registers (QEMR, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. For details on EDMA3CC error interrupt generation, see *Error Interrupts*.

The EMR is shown in [Figure 11-50](#) and described in [Table 11-35](#). The EMRH is shown in [Figure 11-51](#) and described in [Table 11-36](#).

**Figure 11-50. Event Missed Register (EMR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-35. Event Missed Register (EMR) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Channel 0-31 event missed. En is cleared by writing a 1 to the corresponding bit in the event missed clear register (EMCR). |
|      |       | 1     | No missed event.  |
|      |       |       | Missed event occurred.  |

**Figure 11-51. Event Missed Register High (EMRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-36. Event Missed Register High (EMRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Channel 32-63 event missed. En is cleared by writing a 1 to the corresponding bit in the event missed clear register high (EMCRH). |
|      |       | 1     | No missed event.   |
|      |       |       | Missed event occurred.   |

#### 11.4.1.2.2 Event Missed Clear Registers (EMCR/EMCRH)

Once a missed event is posted in the event missed registers (EMR/EMRH), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the event missed clear registers (EMCR/EMCRH). Writing a 1 to any of the bits clears the corresponding missed event (bit) in EMR/EMRH; writing a 0 has no effect.

The EMCR is shown in [Figure 11-52](#) and described in [Table 11-37](#). The EMCRH is shown in [Figure 11-53](#) and described in [Table 11-38](#).

**Figure 11-52. Event Missed Clear Register (EMCR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-37. Event Missed Clear Register (EMCR) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event missed 0-31 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. |
|      |       | 1     | No effect.  |
|      |       |       | Corresponding missed event bit in the event missed register (EMR) is cleared (En = 0).                                      |

**Figure 11-53. Event Missed Clear Register High (EMCRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-38. Event Missed Clear Register High (EMCRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event missed 32–63 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. |
|      |       | 1     | No effect.   |
|      |       |       | Corresponding missed event bit in the event missed register high (EMRH) is cleared (En = 0).                                 |

#### 11.4.1.2.3 QDMA Event Missed Register (QEMR)

For a particular QDMA channel, if two QDMA events are detected without the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the QDMA event missed register (QEMR). The QEMR bits for a channel are also set if a QDMA event on the channel encounters a NULL entry (or a NULL TR is serviced). If any QEMR bit is set (and all errors, including bits in other error registers (EMR/EMRH, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. For details on EDMA3CC error interrupt generation, see *Error Interrupts*.

The QEMR is shown in [Figure 11-54](#) and described in [Table 11-39](#).

**Figure 11-54. QDMA Event Missed Register (QEMR)**

|    |          |  |     |     |     |     |     |     |     |     |
|----|----------|--|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 | Reserved |  |     |     |     |     |     |     | 16  |     |
|    | R-0      |  |     |     |     |     |     |     |     |     |
| 15 | Reserved |  |     |     |     |     |     |     | 0   |     |
|    | R-0      |  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|    |          |  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
|    |          |  | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-39. QDMA Event Missed Register (QEMR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-0  | En       | 0     | Channel 0-7 QDMA event missed. En is cleared by writing a 1 to the corresponding bit in the QDMA event missed clear register (QEMCR).   |
|      |          | 1     | No missed event.<br>Missed event occurred.  |

#### 11.4.1.2.4 QDMA Event Missed Clear Register (QEMCR)

Once a missed event is posted in the QDMA event missed registers (QEMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the QDMA event missed clear registers (QEMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in QEMR; writing a 0 has no effect.

The QEMCR is shown in [Figure 11-55](#) and described in [Table 11-40](#).

**Figure 11-55. QDMA Event Missed Clear Register (QEMCR)**

|                                     |          |  |  |  |  |  |  |  |  |  |    |  |  |  |  |  |  |  |  |  |  |
|-------------------------------------|----------|--|--|--|--|--|--|--|--|--|----|--|--|--|--|--|--|--|--|--|--|
| 31                                  | Reserved |  |  |  |  |  |  |  |  |  | 16 |  |  |  |  |  |  |  |  |  |  |
| R-0                                 |          |  |  |  |  |  |  |  |  |  |    |  |  |  |  |  |  |  |  |  |  |
| 15                                  | Reserved |  |  |  |  |  |  |  |  |  | 0  |  |  |  |  |  |  |  |  |  |  |
| R-0                                 |          |  |  |  |  |  |  |  |  |  |    |  |  |  |  |  |  |  |  |  |  |
| w-0 w-0 w-0 w-0 W-0 W-0 W-0 W-0 W-0 |          |  |  |  |  |  |  |  |  |  |    |  |  |  |  |  |  |  |  |  |  |

LEGEND: W = Write only; -n = value after reset

**Table 11-40. QDMA Event Missed Clear Register (QEMCR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-0  | En       | 0     | QDMA event missed clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC.             |
|      |          | 1     | No effect.  |
|      |          | 1     | Corresponding missed event bit in the QDMA event missed register (QEMR) is cleared (En = 0).  |

#### 11.4.1.2.5 EDMA3CC Error Register (CCERR)

The EDMA3CC error register (CCERR) indicates whether or not at any instant of time the number of events queued up in any of the event queues exceeds or equals the threshold/watermark value that is set in the queue watermark threshold register (QWMTHRA). Additionally, CCERR also indicates if when the number of outstanding TRs that have been programmed to return transfer completion code (TRs which have the TCINTEN or TCCHEN bit in OPT set) to the EDMA3CC has exceeded the maximum allowed value of 63. If any bit in CCERR is set (and all errors, including bits in other error registers (EMR/EMRH, QEMR) were previously cleared), the EDMA3CC generates an error interrupt. For details on EDMA3CC error interrupt generation, see *Error Interrupts*. Once the error bits are set in CCERR, they can only be cleared by writing to the corresponding bits in the EDMA3CC error clear register (CCERRCLR).

The CCERR is shown in [Figure 11-56](#) and described in [Table 11-41](#).

**Figure 11-56. EDMA3CC Error Register (CCERR)**

|          |          |          |          |          |   |    |    |
|----------|----------|----------|----------|----------|---|----|----|
| 31       | Reserved |          |          |          |   | 17 | 16 |
| R-0      |          |          |          |          |   |    |    |
| 15       | 4        | 3        | 2        | 1        | 0 |    |    |
| Reserved | QTHRXC03 | QTHRXC02 | QTHRXC01 | QTHRXC00 |   |    |    |
| R-0      | R-0      | R-0      | R-0      | R-0      |   |    |    |

LEGEND: R = Read only; -n = value after reset

**Table 11-41. EDMA3CC Error Register (CCERR) Field Descriptions**

| Bit   | Field     | Value | Description  |
|-------|-----------|-------|--|
| 31-17 | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.          |
| 16    | TCCERR    | 0     | Transfer completion code error. TCCERR is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).        |
|       |           | 1     | Total number of allowed TCCs outstanding has not been reached.   |
|       |           | 1     | Total number of allowed TCCs has been reached.   |
| 15-4  | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.          |
| 3     | QTHRXC3D3 | 0     | Queue threshold error for queue 3. QTHRXC3D3 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).  |
|       |           | 0     | Watermark/threshold has not been exceeded.   |
|       |           | 1     | Watermark/threshold has been exceeded.   |
| 2     | QTHRXC2D2 | 0     | Queue threshold error for queue 2. QTHRXC2D2 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).  |
|       |           | 0     | Watermark/threshold has not been exceeded.   |
|       |           | 1     | Watermark/threshold has been exceeded.   |
| 1     | QTHRXC1D1 | 0     | Queue threshold error for queue 1 . QTHRXC1D1 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR). |
|       |           | 0     | Watermark/threshold has not been exceeded.   |
|       |           | 1     | Watermark/threshold has been exceeded.   |
| 0     | QTHRXC0D0 | 0     | Queue threshold error for queue 0. QTHRXC0D0 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR).  |
|       |           | 0     | Watermark/threshold has not been exceeded.   |
|       |           | 1     | Watermark/threshold has been exceeded.   |

#### 11.4.1.2.6 EDMA3CC Error Clear Register (CCERRCLR)

The EDMA3CC error clear register (CCERRCLR) is used to clear any error bits that are set in the EDMA3CC error register (CCERR). In addition, CCERRCLR also clears the values of some bit fields in the queue status registers (QSTAT $n$ ) associated with a particular event queue. Writing a 1 to any of the bits clears the corresponding bit in CCERR; writing a 0 has no effect.

The CCERRCLR is shown in [Figure 11-57](#) and described in [Table 11-42](#).

**Figure 11-57. EDMA3CC Error Clear Register (CCERRCLR)**

|          |          |     |     |     |     |     |
|----------|----------|-----|-----|-----|-----|-----|
| 31       | Reserved |     |     |     | 17  | 16  |
|          |          |     |     |     | W-0 | W-0 |
| 15       | 4        | 3   | 2   | 1   | 0   |     |
| Reserved | W-0      | W-0 | W-0 | W-0 | W-0 | W-0 |
|          |          |     |     |     |     |     |

LEGEND: W = Write only; -n = value after reset

**Table 11-42. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-17 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 16    | TCCERR   | 0     | Transfer completion code error clear.<br>No effect.   |
|       |          | 1     | Clears the TCCERR bit in the EDMA3CC error register (CCERR).  |

**Table 11-42. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions (continued)**

| Bit  | Field     | Value | Description   |
|------|-----------|-------|---|
| 15-4 | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 3    | QTHRXC3D3 | 0     | Queue threshold error clear for queue 3.<br>No effect.  |
|      |           | 1     | Clears the QTHRXC3D3 bit in the EDMA3CC error register (CCERR) and the WM and THRXC3D bits in the queue status register 3 (QSTAT3).     |
| 2    | QTHRXC2D2 | 0     | Queue threshold error clear for queue 2.<br>No effect.  |
|      |           | 1     | Clears the QTHRXC2D2 bit in the EDMA3CC error register (CCERR) and the WM and THRXC2D bits in the queue status register 2 (QSTAT2).     |
| 1    | QTHRXC1D1 | 0     | Queue threshold error clear for queue 1.<br>No effect.  |
|      |           | 1     | Clears the QTHRXC1D1 bit in the EDMA3CC error register (CCERR) and the WM and THRXC1D bits in the queue status register 1 (QSTAT1).     |
| 0    | QTHRXC0D0 | 0     | Queue threshold error clear for queue 0.<br>No effect.  |
|      |           | 1     | Clears the QTHRXC0D0 bit in the EDMA3CC error register (CCERR) and the WM and THRXC0D bits in the queue status register 0 (QSTAT0).     |

#### 11.4.1.2.7 Error Evaluation Register (EEVAL)

The EDMA3CC error interrupt is asserted whenever an error bit is set in any of the error registers (EMR/EMRH, QEMR, and CCERR). For subsequent error bits that get set, the EDMA3CC error interrupt is reass器ted only when transitioning from an “all the error bits cleared” to “at least one error bit is set”. Alternatively, a CPU write of 1 to the EVAL bit in the error evaluation register (EEVAL) results in reasserting the EDMA3CC error interrupt, if there are any outstanding error bits set due to subsequent error conditions. Writes of 0 have no effect.

The EEVAL is shown in [Figure 11-58](#) and described in [Table 11-43](#).

**Figure 11-58. Error Evaluation Register (EEVAL)**

|     |          |  |  |                              |
|-----|----------|--|--|------------------------------|
| 31  | Reserved |  |  | 16                           |
| R-0 |          |  |  |                              |
| 15  | Reserved |  |  | 2      1      0              |
|     |          |  |  | Rsvd    EVAL<br>R/W-0    W-0 |

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -n = value after reset

**Table 11-43. Error Evaluation Register (EEVAL) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-2 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0    | EVAL     | 0     | Error interrupt evaluate.<br>No effect.   |
|      |          | 1     | EDMA3CC error interrupt will be pulsed if any errors have not been cleared in any of the error registers (EMR/EMRH, QEMR, or CCERR).    |

### 11.4.1.3 Region Access Enable Registers

The region access enable register group consists of the DMA access enable registers (DRAEm and DRAEHm) and the QDMA access enable registers (QRAEm). Where  $m$  is the number of shadow regions in the EDMA3CC memory map for a device. You can configure these registers to assign ownership of DMA/QDMA channels to a particular shadow region.

#### 11.4.1.3.1 DMA Region Access Enable for Region $m$ (DRAEm)

The DMA region access enable register for shadow region  $m$  (DRAEm/DRAEHm) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all DMA registers in the shadow region  $m$  view of the DMA channel registers. See the EDMA3CC register memory map ([Table 11-25](#)) for a list of all the DMA channel and interrupt registers mapped in the shadow region view. Additionally, the DRAEm/DRAEHm configuration determines completion of which DMA channels will result in assertion of the shadow region  $m$  DMA completion interrupt (see *EDMA3 Interrupts*).

The DRAEm is shown in [Figure 11-59](#) and described in [Table 11-44](#). The DRAEHm is shown in [Figure 11-60](#) and described in [Table 11-44](#).

**Figure 11-59. DMA Region Access Enable Register for Region  $m$  (DRAEm)**

| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| E31  | E30  | E29  | E28  | E27  | E26  | E25  | E24  | E23  | E22  | E21  | E20  | E19  | E18  | E17  | E16  |
| RW-0 |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| E15  | E14  | E13  | E12  | E11  | E10  | E9   | E8   | E7   | E6   | E5   | E4   | E3   | E2   | E1   | E0   |
| RW-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Figure 11-60. DMA Region Access Enable High Register for Region  $m$  (DRAEHm)**

| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| E63  | E62  | E61  | E60  | E59  | E58  | E57  | E56  | E55  | E54  | E53  | E52  | E51  | E50  | E49  | E48  |
| RW-0 |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| E47  | E46  | E45  | E44  | E43  | E42  | E41  | E40  | E39  | E38  | E37  | E36  | E35  | E34  | E33  | E32  |
| RW-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 11-44. DMA Region Access Enable Registers for Region M (DRAEm/DRAEHm)  
Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | DMA region access enable for bit $n$ /channel $n$ in region $m$ .<br>Accesses via region $m$ address space to bit $n$ in any DMA channel register are not allowed. Reads return 0 on bit $n$ and writes do not modify the state of bit $n$ . Enabled interrupt bits for bit $n$ do not contribute to the generation of a transfer completion interrupt for shadow region $m$ . |
|      |       | 1     | Accesses via region $m$ address space to bit $n$ in any DMA channel register are allowed. Reads return the value from bit $n$ and writes modify the state of bit $n$ . Enabled interrupt bits for bit $n$ contribute to the generation of a transfer completion interrupt for shadow region $m$ .  |

#### 11.4.1.3.2 QDMA Region Access Enable Registers (QRAEm)

The QDMA region access enable register for shadow region  $m$  (QRAEm) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all QDMA registers in the shadow region  $m$  view of the QDMA registers. This includes all 4-bit QDMA registers.

The QRAEm is shown in [Figure 11-61](#) and described in [Table 11-45](#).

**Figure 11-61. QDMA Region Access Enable for Region  $m$  (QRAEm)32-bit, 2 Rows**

|    |          |   |      |      |      |      |      |      |      |      |    |
|----|----------|---|------|------|------|------|------|------|------|------|----|
| 31 |          |   |      |      |      |      |      |      |      |      | 16 |
|    | Reserved |   |      |      |      |      |      |      |      |      |    |
|    | R-0      |   |      |      |      |      |      |      |      |      |    |
| 15 |          | 8 | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |    |
|    | Reserved |   | E7   | E6   | E5   | E4   | E3   | E2   | E1   | E0   |    |
|    | R-0      |   | RW-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-45. QDMA Region Access Enable for Region M (QRAEm) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 7-0  | En       | 0     | QDMA region access enable for bit $n$ /QDMA channel $n$ in region $m$ .<br>Accesses via region $m$ address space to bit $n$ in any QDMA channel register are not allowed. Reads return 0 on bit $n$ and writes do not modify the state of bit $n$ . |
|      |          | 1     | Accesses via region $m$ address space to bit $n$ in any QDMA channel register are allowed. Reads return the value from bit $n$ and writes modify the state of bit $n$ .   |

#### 11.4.1.4 Status/Debug Visibility Registers

The following set of registers provide visibility into the event queues and a TR life cycle. These are useful for system debug as they provide in-depth visibility for the events queued up in the event queue and also provide information on what parts of the EDMA3CC logic are active once the event has been received by the EDMA3CC.

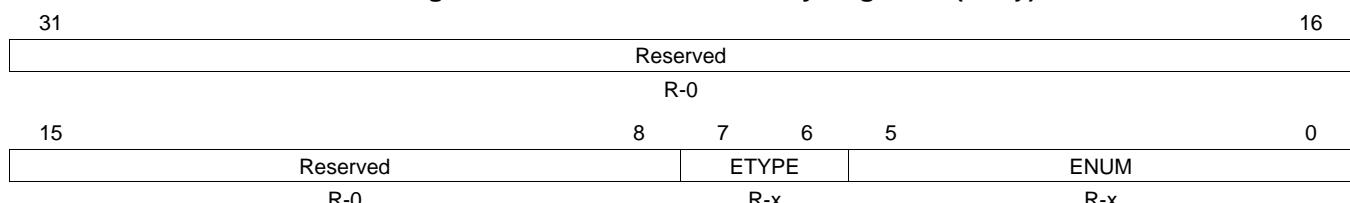
##### 11.4.1.4.1 Event Queue Entry Registers (QxEy)

The event queue entry registers (QxEy) exist for all 16 queue entries (the maximum allowed queue entries) for all event queues in the EDMA3CC.

There are Q0E0 to Q0E15, Q1E0 to Q1E15, Q2E0 to Q2E15, and Q3E0 to Q3E15. Each register details the event number (ENUM) and the event type (ETYPE). For example, if the value in Q1E4 is read as 000 004Fh, this means the 4th entry in queue 1 is a manually-triggered event on DMA channel 15.

The QxEy is shown in [Figure 11-62](#) and described in [Table 11-46](#).

**Figure 11-62. Event Queue Entry Registers (QxEy)**



LEGEND: R = Read only; -n = value after reset; -x = value is indeterminate after reset

**Table 11-46. Event Queue Entry Registers (QxEy) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-6  | ETYPE    | 0-3h  | Event entry y in queue x. Specifies the specific event type for the given entry in the event queue.                                     |
|      |          | 0     | Event triggered via ER.   |
|      |          | 1h    | Manual triggered via ESR.   |
|      |          | 2h    | Chain triggered via CER.  |
|      |          | 3h    | Auto-triggered via QER.   |
| 5-0  | ENUM     | 0-3Fh | Event entry y in queue x. Event number:   |
|      |          | 0-3h  | QDMA channel number (0 to 3).   |
|      |          | 0-3Fh | DMA channel/event number (0 to 63).   |

#### 11.4.1.4.2 Queue Status Registers (QSTAT $n$ )

The queue status registers (QSTAT $n$ ) is shown in [Figure 11-63](#) and described in [Table 11-47](#).

**Figure 11-63. Queue Status Register  $n$  (QSTAT $n$ )**

| 31       | 25 | 24     | 23       | 21       | 20  | 16      |
|----------|----|--------|----------|----------|-----|---------|
| Reserved |    | THRXCD | Reserved |          | WM  |         |
| R-0      |    | R-0    | R-0      |          | R-0 |         |
| 15       | 13 | 12     | 8        | 7        | 4   | 3       |
| Reserved |    | NUMVAL |          | Reserved |     | STRTPTR |
| R-0      |    | R-0    |          | R-0      |     | R-0     |

LEGEND: R = Read only; -n = value after reset

**Table 11-47. Queue Status Register  $n$  (QSTAT $n$ ) Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31-25 | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 24    | THRXCD   | 0       | Threshold exceeded. THRXC $D$ is cleared by writing a 1 to the corresponding QTHRXC $Dn$ bit in the EDMA3CC error clear register (CCERRCLR).   |
|       |          | 1       | Threshold specified by the Q $n$ bit in the queue watermark threshold A register (QWMTHRA) has not been exceeded.  |
|       |          | 1       | Threshold specified by the Q $n$ bit in the queue watermark threshold A register (QWMTHRA) has been exceeded.  |
| 23-21 | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 20-16 | WM       | 0-10h   | Watermark for maximum queue usage. Watermark tracks the most entries that have been in queue $n$ since reset or since the last time that the watermark (WM) bit was cleared. WM is cleared by writing a 1 to the corresponding QTHRXC $Dn$ bit in the EDMA3CC error clear register (CCERRCLR). |
|       |          | 0-10h   | Legal values are 0 (empty) to 10h (full).  |
|       |          | 11h-1Fh | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 15-13 | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 12-8  | NUMVAL   | 0-10h   | Number of valid entries in queue $n$ . The total number of entries residing in the queue manager FIFO at a given instant. Always enabled.  |
|       |          | 0-10h   | Legal values are 0 (empty) to 10h (full).  |
|       |          | 11h-1Fh | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 7-4   | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 3-0   | STRTPTR  | 0-Fh    | Start pointer. The offset to the head entry of queue $n$ , in units of entries. Always enabled. Legal values are 0 (0th entry) to Fh (15th entry).   |

#### 11.4.1.4.3 Queue Watermark Threshold A Register (QWMTHRA)

The queue watermark threshold A register (QWMTHRA) is shown in [Figure 11-64](#) and described in [Table 11-48](#).

**Figure 11-64. Queue Watermark Threshold A Register (QWMTHRA)**

| 31       | 29 | 28      | 24 | 23       | 21 | 20      | 16 |
|----------|----|---------|----|----------|----|---------|----|
| Reserved |    | Q3      |    | Reserved |    | Q2      |    |
| R-0      |    | R/W-10  |    | R-0      |    | R/W-10  |    |
| 15       | 13 | 12      | 8  | 7        | 5  | 4       | 0  |
| Reserved |    | Q1      |    | Reserved |    | Q0      |    |
| R-0      |    | R/W-10h |    | R-0      |    | R/W-10h |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-48. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions**

| Bit   | Field    | Value                            | Description   |
|-------|----------|----------------------------------|---|
| 31-29 | Reserved | 0                                | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 28-24 | Q3       | 0-1Fh<br>0-10h<br>11h<br>12h-1Fh | Queue threshold for queue 3 value. The QTHRXC3D bit in the EDMA3CC error register (CCERR) and the THRXC3D bit in the queue status register 3 (QSTAT3) are set when the number of events in queue 3 at an instant in time (visible via the NUMVAL bit in QSTAT3) equals or exceeds the value specified by Q3.<br><br>The default is 16 (maximum allowed).<br><br>Disables the threshold errors.<br><br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 23-21 | Reserved | 0                                | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 20-16 | Q2       | 0-1Fh<br>0-10h<br>11h<br>12h-1Fh | Queue threshold for queue 2 value. The QTHRXC2D bit in the EDMA3CC error register (CCERR) and the THRXC2D bit in the queue status register 2 (QSTAT2) are set when the number of events in queue 2 at an instant in time (visible via the NUMVAL bit in QSTAT2) equals or exceeds the value specified by Q2.<br><br>The default is 16 (maximum allowed).<br><br>Disables the threshold errors.<br><br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 15-13 | Reserved | 0                                | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 12-8  | Q1       | 0-10h<br>11h<br>12h-1Fh          | Queue threshold for queue 1 value. The QTHRXC1D bit in the EDMA3CC error register (CCERR) and the THRXC1D bit in the queue status register 1 (QSTAT1) are set when the number of events in queue 1 at an instant in time (visible via the NUMVAL bit in QSTAT1) equals or exceeds the value specified by Q1.<br><br>The default is 16 (maximum allowed).<br><br>Disables the threshold errors.<br><br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-5   | Reserved | 0                                | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 4-0   | Q0       | 0-10h<br>11h<br>12h-1Fh          | Queue threshold for queue 0 value. The QTHRXC0D bit in the EDMA3CC error register (CCERR) and the THRXC0D bit in the queue status register 0 (QSTAT0) are set when the number of events in queue 0 at an instant in time (visible via the NUMVAL bit in QSTAT0) equals or exceeds the value specified by Q0.<br><br>The default is 16 (maximum allowed).<br><br>Disables the threshold errors.<br><br>Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |

#### 11.4.1.4.4 EDMA3CC Status Register (CCSTAT)

The EDMA3CC status register (CCSTAT) has a number of status bits that reflect which parts of the EDMA3CC logic is active at any given instant of time. The CCSTAT is shown in [Figure 11-65](#) and described in [Table 11-49](#).

**Figure 11-65. EDMA3CC Status Register (CCSTAT)**

|          |          |          |      |          |          |          |          |  |    |
|----------|----------|----------|------|----------|----------|----------|----------|--|----|
| 31       | Reserved |          |      |          |          |          |          |  | 24 |
| R-0      |          |          |      |          |          |          |          |  |    |
| 23       | 22       | 21       | 20   | 19       | 18       | 17       | 16       |  |    |
| Reserved |          | Reserved |      | QUEACTV3 | QUEACTV2 | QUEACTV1 | QUEACTV0 |  |    |
| R-0      |          | R-0      |      | R-0      | R-0      | R-0      | R-0      |  |    |
| 15       | 14       | 13       | 12   | 11       | 10       | 9        | 8        |  |    |
| Reserved |          | COMPACTV |      |          |          |          |          |  |    |
| R-0      |          |          |      |          |          |          |          |  |    |
| 7        | 6        | 5        | 4    | 3        | 2        | 1        | 0        |  |    |
| Reserved |          |          | ACTV | Reserved | TRACTV   | QEVTACTV | EVTACTV  |  |    |
| R-0      |          |          | R-0  | R-0      | R-0      | R-0      | R-0      |  |    |

LEGEND: R = Read only; -n = value after reset

**Table 11-49. EDMA3CC Status Register (CCSTAT) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 31-20 | Reserved | 0      | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 19    | QUEACTV3 | 0      | Queue 3 active.<br>No events are queued in queue 3.  |
|       |          | 1      | At least one TR is queued in queue 3.  |
| 18    | QUEACTV2 | 0      | Queue 2 active.<br>No events are queued in queue 2.  |
|       |          | 1      | At least one TR is queued in queue 2.  |
| 17    | QUEACTV1 | 0      | Queue 1 active.<br>No events are queued in queue 1.  |
|       |          | 1      | At least one TR is queued in queue 1.  |
| 16    | QUEACTV0 | 0      | Queue 0 active.<br>No events are queued in queue 0.  |
|       |          | 1      | At least one TR is queued in queue 0.  |
| 15-14 | Reserved | 0      | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 13-8  | COMPACTV | 0-3Fh  | Completion request active. The COMPACTV field reflects the count for the number of completion requests submitted to the transfer controllers. This count increments every time a TR is submitted and is programmed to report completion (the TCINTEN or TCCCHEN bits in OPT in the parameter entry associated with the TR are set). The counter decrements for every valid TCC received back from the transfer controllers. If at any time the count reaches a value of 63, the EDMA3CC will not service any new TRs until the count is less than 63 (or return a transfer completion code from a transfer controller, which would decrement the count). |
|       |          | 0      | No completion requests outstanding.  |
|       |          | 1h-3Fh | Total of 1 completion request to 63 completion requests are outstanding.   |
| 7-5   | Reserved | 0      | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |

**Table 11-49. EDMA3CC Status Register (CCSTAT) Field Descriptions (continued)**

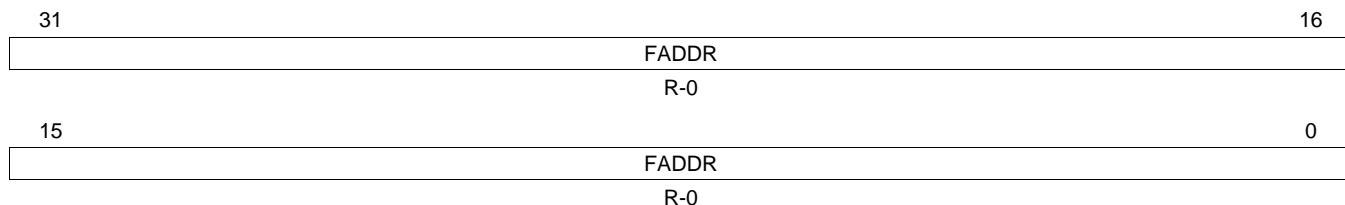
| Bit | Field    | Value | Description   |
|-----|----------|-------|---|
| 4   | ACTV     | 0     | Channel controller active. Channel controller active is a logical-OR of each of the *ACTV bits. The ACTV bit remains high through the life of a TR. |
|     |          |       | Channel is idle..   |
|     |          | 1     | Channel is busy.  |
| 3   | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.             |
|     |          |       | Transfer request active.  |
|     |          | 1     | Transfer request processing/submission logic is inactive.   |
| 2   | TRACTV   | 0     | Transfer request processing/submission logic is active.   |
|     |          |       | QEVTACTV  |
|     |          | 1     | QDMA event active.<br>No enabled QDMA events are active within the EDMA3CC.<br>At least one enabled QDMA event (QER) is active within the EDMA3CC.  |
| 1   | EVTACTV  | 0     | DMA event active.<br>No enabled DMA events are active within the EDMA3CC.   |
|     |          |       | At least one enabled DMA event (ER and EER, ESR, CER) is active within the EDMA3CC.   |
|     |          | 1     |   |

### 11.4.1.5 Memory Protection Address Space

#### 11.4.1.5.1 *Memory Protection Fault Address Register (MPFAR)*

The memory protection fault address register (MPFAR) is shown in [Figure 11-66](#) and described in [Table 11-50](#). A CPU write of 1 to the MPFCLR bit in the memory protection fault command register (MPFCR) causes any error conditions stored in MPFAR to be cleared.

**Figure 11-66. Memory Protection Fault Address Register (MPFAR)**



LEGEND: R = Read only; -n = value after reset

**Table 11-50. Memory Protection Fault Address Register (MPFAR) Field Descriptions**

| Bit  | Field | Value        | Description   |
|------|-------|--------------|---|
| 31-0 | FADDR | 0xFFFF FFFFh | Fault address. This 32-bit read-only status register contains the fault address when a memory protection violation is detected. This register can only be cleared via the memory protection fault command register (MPFCR). |

#### 11.4.1.5.2 Memory Protection Fault Status Register (MPFSR)

The memory protection fault status register (MPFSR) is shown in [Figure 11-67](#) and described in [Table 11-51](#). A CPU write of 1 to the MPFCLR bit in the memory protection fault command register (MPFCR) causes any error conditions stored in MPFSR to be cleared.

**Figure 11-67. Memory Protection Fault Status Register (MPFSR)**

| 31       |    |     |   |          |     |     |     |     |     |     |   |  |  |  | 16  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----------|----|-----|---|----------|-----|-----|-----|-----|-----|-----|---|--|--|--|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Reserved |    |     |   |          |     |     |     |     |     |     |   |  |  |  | R-0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R-0      |    |     |   |          |     |     |     |     |     |     |   |  |  |  | 0   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15       | 13 | 12  | 9 | 8        | 6   | 5   | 4   | 3   | 2   | 1   | 0 |  |  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Reserved |    | FID |   | Reserved | SRE | SWE | SXE | URE | UWE | UXE |   |  |  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R-0      |    | R-0 |   | R-0      | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |   |  |  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

LEGEND: R = Read only; -n = value after reset

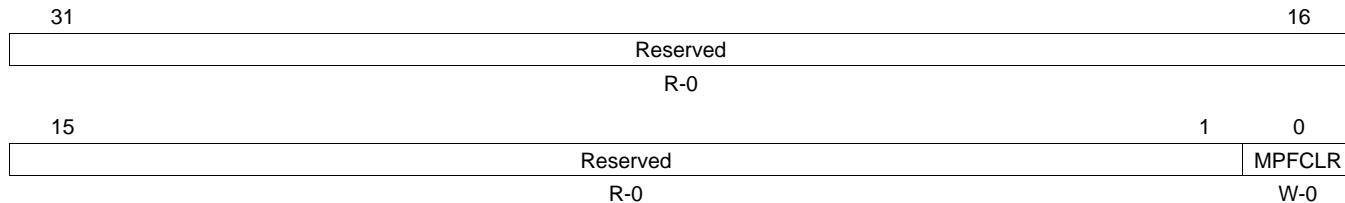
**Table 11-51. Memory Protection Fault Status Register (MPFSR) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-13 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 12-9  | FID      | 0-Fh  | Faulted identification. FID contains valid information if any of the MP error bits (UXE, UWE, URE, SXE, SWE, SRE) are nonzero (that is, if an error has been detected.) The FID field contains the privilege ID for the specific request/requestor that resulted in an MP error. |
| 8-6   | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 5     | SRE      | 0     | Supervisor read error.<br>No error detected.   |
|       |          | 1     | Supervisor level task attempted to read from a MP page without SR permissions.   |
| 4     | SWE      | 0     | Supervisor write error.<br>No error detected.  |
|       |          | 1     | Supervisor level task attempted to write to a MP page without SW permissions.  |
| 3     | SXE      | 0     | Supervisor execute error.<br>No error detected.  |
|       |          | 1     | Supervisor level task attempted to execute from a MP page without SX permissions.  |
| 2     | URE      | 0     | User read error.<br>No error detected.   |
|       |          | 1     | User level task attempted to read from a MP page without UR permissions.   |
| 1     | UWE      | 0     | User write error.<br>No error detected.  |
|       |          | 1     | User level task attempted to write to a MP page without UW permissions.  |
| 0     | UXE      | 0     | User execute error.<br>No error detected.  |
|       |          | 1     | User level task attempted to execute from a MP page without UX permissions.  |

#### **11.4.1.5.3 Memory Protection Fault Command Register (MPFCR)**

The memory protection fault command register (MPFCR) is shown in [Figure 11-68](#) and described in [Table 11-52](#).

**Figure 11-68. Memory Protection Fault Command Register (MPFCR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-52. Memory Protection Fault Command Register (MPFCR) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-1 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 0    | MPFCLR   | 0     | Fault clear register.<br>CPU write of 0 has no effect.   |
|      |          | 1     | CPU write of 1 to the MPFCLR bit causes any error conditions stored in the memory protection fault address register (MPFAR) and the memory protection fault status register (MPFSR) to be cleared. |

#### 11.4.1.5.4 Memory Protection Page Attribute Register (MPPAn)

The memory protection page attribute register (MPPAn) is shown in [Figure 11-69](#) and described in [Table 11-53](#).

**Figure 11-69. Memory Protection Page Attribute Register (MPPAn)**

| 31   | Reserved |      |      |      |      |     |      |          |    |    |    |    |    |    |   | 16 |
|------|----------|------|------|------|------|-----|------|----------|----|----|----|----|----|----|---|----|
| R-0  |          |      |      |      |      |     |      |          |    |    |    |    |    |    |   |    |
| 15   | 14       | 13   | 12   | 11   | 10   | 9   | 8    | 7        | 6  | 5  | 4  | 3  | 2  | 1  | 0 |    |
| AID5 | AID4     | AID3 | AID2 | AID1 | AID0 | EXT | Rsvd | Reserved | SR | SW | SX | UR | UW | UX |   |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-53. Memory Protection Page Attribute Register (MPPAn) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.      |
| 15-10 | AID $m$  | 0     | Allowed ID 'N'<br>Requests with Privilege ID == N are not allowed to region M, regardless of permission settings (UW, UR, SW, SR).           |
|       |          | 1     | Requests with Privilege ID == N are permitted, if access type is allowed as defined by permission settings (UW, UR, SW, SR).                 |
| 9     | EXT      | 0     | External Allowed ID.<br>Requests with Privilege ID $\geq$ 6 are not allowed to region M, regardless of permission settings (UW, UR, SW, SR). |
|       |          | 1     | Requests with Privilege ID $\geq$ 6 are permitted, if access type is allowed as defined by permission settings (UW, UR, SW, SR).             |
| 8     | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.      |
| 7-6   | Reserved | 1     | Reserved. Always write 1 to this bit.  |
| 5     | SR       | 0     | Supervisor read permission.<br>Supervisor read accesses are not allowed from region M.   |
|       |          | 1     | Supervisor write accesses are allowed from region M addresses.   |
| 4     | SW       | 0     | Supervisor write permission.<br>Supervisor write accesses are not allowed to region M.   |
|       |          | 1     | Supervisor write accesses are allowed to region N addresses.   |
| 3     | SX       | 0     | Supervisor execute permission.<br>Supervisor execute accesses are not allowed from region M.   |
|       |          | 1     | Supervisor execute accesses are allowed from region M addresses.   |
| 2     | UR       | 0     | User read permission.<br>User read accesses are not allowed from region M.   |
|       |          | 1     | User read accesses are allowed from region N addresses.  |
| 1     | UW       | 0     | User write permission.<br>User write accesses are not allowed to region M.   |
|       |          | 1     | User write accesses are allowed to region M addresses.   |
| 0     | UX       | 0     | User execute permission.<br>User execute accesses are not allowed from region M.   |
|       |          | 1     | User execute accesses are allowed from region M addresses.   |

### 11.4.1.6 DMA Channel Registers

The following sets of registers pertain to the 64 DMA channels. The 64 DMA channels consist of a set of registers (with exception of DMAQNUM $n$ ) that each have 64 bits and the bit position of each register matches the DMA channel number. Each register is named with the format *reg\_name* that corresponds to DMA channels 0 through 31 and *reg\_name\_High* that corresponds to DMA channels 32 through 64.

For example, the event register (ER) corresponds to DMA channel 0 through 31 and the event register high register (ERH) corresponds to DMA channel 32 through 63. The register is typically called the event register.

The DMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region are controlled by the DMA region access registers (DRAEm/DRAEHm). The registers are described in [Section 11.4.1.3.1](#) and the details for shadow region/global region usage is explained in *EDMA3 Channel Controller Regions*.

#### 11.4.1.6.1 Event Registers (ER, ERH)

All external events are captured in the event register (ER/ERH). The events are latched even when the events are not enabled. If the event bit corresponding to the latched event is enabled (EER.En/EERH.En = 1), then the event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. The event register bits are automatically cleared (ER.En/ERH.En = 0) once the corresponding events are prioritized and serviced. If ER.En/ERH.En are already set and another event is received on the same channel/event, then the corresponding event is latched in the event miss register (EMR.En/EMRH.En), provided that the event was enabled (EER.En/EERH.En = 1).

Event  $n$  can be cleared by the CPU writing a 1 to corresponding event bit in the event clear register (ECR/ECRH). The setting of an event is a higher priority relative to clear operations (via hardware or software). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The *Debug List* table provides the type of synchronization events and the EDMA3CC channels associated to each of these external events.

The ER is shown in [Figure 11-70](#) and described in [Table 11-54](#). The ERH is shown in [Figure 11-71](#) and described in [Table 11-55](#).

**Figure 11-70. Event Register (ER)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | 19  | E18 | E17 | E16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-54. Event Register (ER) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event 0-31. Events 0-31 are captured by the EDMA3CC and are latched into ER. The events are set (En = 1) even when events are disabled (En = 0 in the event enable register, EER).<br>EDMA3CC event is not asserted. |
|      |       | 1     | EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.  |

**Figure 11-71. Event Register High (ERH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-55. Event Register High (ERH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event 32-63. Events 32-63 are captured by the EDMA3CC and are latched into ERH. The events are set (En = 1) even when events are disabled (En = 0 in the event enable register high, EERH).<br>EDMA3CC event is not asserted. |
|      |       | 1     | EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.   |

#### 11.4.1.6.2 Event Clear Registers (ECR, ECRH)

Once an event has been posted in the event registers (ER/ERH), the event is cleared in two ways. If the event is enabled in the event enable register (EER/EERH) and the EDMA3CC submits a transfer request for the event to the EDMA3TC, it clears the corresponding event bit in the event register. If the event is disabled in the event enable register (EER/EERH), the CPU can clear the event by way of the event clear registers (ECR/ECRH).

Writing a 1 to any of the bits clears the corresponding event; writing a 0 has no effect. Once an event bit is set in the event register, it remains set until EDMA3CC submits a transfer request for that event or the CPU clears the event by setting the corresponding bit in ECR/ECRH.

The ECR is shown in [Figure 11-72](#) and described in [Table 11-56](#). The ECRH is shown in [Figure 11-73](#) and described in [Table 11-57](#).

**Figure 11-72. Event Clear Register (ECR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-56. Event Clear Register (ECR) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event clear for event 0-31. Any of the event bits in ECR is set to clear the event (En) in the event register (ER). A write of 0 has no effect. |
|      |       | 1     | No effect.  |
|      |       |       | EDMA3CC event is cleared in the event register (ER).  |

**Figure 11-73. Event Clear Register High (ECRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-57. Event Clear Register High (ECRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event clear for event 32-63. Any of the event bits in ECRH are set to clear the event (En) in the event register high (ERH). A write of 0 has no effect. |
|      |       | 1     | No effect.   |
|      |       |       | EDMA3CC event is cleared in the event register high (ERH).   |

### 11.4.1.6.3 Event Set Registers (ESR, ESRH)

The event set registers (ESR/ESRH) allow the CPU (EDMA3 programmers) to manually set events to initiate DMA transfer requests. CPU writes of 1 to any event set register (En) bits set the corresponding bits in the registers. The set event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. Writing a 0 has no effect.

The event set registers operate independent of the event registers (ER/ERH), and a write of 1 is always considered a valid event regardless of whether the event is enabled (the corresponding event bits are set or cleared in EER.En/EERH.En).

Once the event is set in the event set registers, it cannot be cleared by CPU writes, in other words, the event clear registers (ECR/ECRH) have no effect on the state of ESR/ESRH. The bits will only be cleared once the transfer request corresponding to the event has been submitted to the transfer controller. The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Manually-triggered transfers via writes to ESR/ESRH allow the CPU to submit DMA requests in the system, these are relevant for memory-to-memory transfer scenarios. If the ESR.En/ESRH.En bit is already set and another CPU write of 1 is attempted to the same bit, then the corresponding event is latched in the event missed registers (EMR.En/EMRH.En = 1).

The ESR is shown in [Figure 11-74](#) and described in [Table 11-58](#). The ESRH is shown in [Figure 11-75](#) and described in [Table 11-59](#).

**Figure 11-74. Event Set Register (ESR)**

| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| E31  | E30  | E29  | E28  | E27  | E26  | E25  | E24  | E23  | E22  | E21  | E20  | E19  | E18  | E17  | E16  |
| RW-0 |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| E15  | E14  | E13  | E12  | E11  | E10  | E9   | E8   | E7   | E6   | E5   | E4   | E3   | E2   | E1   | E0   |
| RW-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 11-58. Event Set Register (ESR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event set for event 0-31.<br>No effect.  |
|      |       | 1     | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

**Figure 11-75. Event Set Register High (ESRH)**

| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24   | 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16   |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| E63  | E62  | E61  | E60  | E59  | E58  | E57  | E56  | E55  | E54  | E53  | E52  | E51  | E50  | E49  | E48  |
| RW-0 |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8    | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| E47  | E46  | E45  | E44  | E43  | E42  | E41  | E40  | E39  | E38  | E37  | E36  | E35  | E34  | E33  | E32  |
| RW-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 11-59. Event Set Register High (ESRH) Field Descriptions**

| Bit  | Field | Value  | Description  |
|------|-------|--------|--|
| 31-0 | En    | 0<br>1 | Event set for event 32-63.<br>No effect.<br>Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

#### 11.4.1.6.4 Chained Event Registers (CER, CERH)

When the OPTIONS parameter for a PaRAM entry is programmed to returned a chained completion code (ITCCHEN = 1 and/or TCCHEN = 1), then the value dictated by the TCC[5:0] (also programmed in OPT) forces the corresponding event bit to be set in the chained event registers (CER/CERH). The set chained event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. This results in a chained-triggered transfer.

The chained event registers do not have any enables. The generation of a chained event is essentially enabled by the PaRAM entry that has been configured for intermediate and/or final chaining on transfer completion. The En bit is set (regardless of the state of EER.En/EERH.En) when a chained completion code is returned from one of the transfer controllers or is generated by the EDMA3CC via the early completion path. The bits in the chained event register are cleared when the corresponding events are prioritized and serviced.

If the En bit is already set and another chaining completion code is return for the same event, then the corresponding event is latched in the event missed registers (EMR.En/EMRH.En = 1). The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The CER is shown in [Figure 11-76](#) and described in [Table 11-60](#). The CERH is shown in [Figure 11-77](#) and described in [Table 11-61](#).

**Figure 11-76. Chained Event Register (CER)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-60. Chained Event Register (CER) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Chained event for event 0-31.<br>No effect.  |
|      |       | 1     | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

**Figure 11-77. Chained Event Register High (CERH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-61. Chained Event Register High (CERH) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31-0       | En           | 0<br>1       | Chained event set for event 32-63.<br>No effect.<br>Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

#### 11.4.1.6.5 Event Enable Registers (EER, EERH)

The EDMA3CC provides the option of selectively enabling/disabling each event in the event registers (ER/ERH) by using the event enable registers (EER/EERH). If an event bit in EER/EERH is set (using the event enable set registers, EESR/EESRH), it will enable that corresponding event. Alternatively, if an event bit in EER/EERH is cleared (using the event enable clear registers, EECR/EECRH), it will disable the corresponding event.

The event registers latch all events that are captured by EDMA3CC, even if the events are disabled (although EDMA3CC does not process it). Enabling an event with a pending event already set in the event registers enables the EDMA3CC to process the already set event like any other new event. The EER/EERH settings do not have any effect on chained events (CER.En/CERH.En = 1) and manually set events (ESR.En/ESRH.En = 1).

The EER is shown in [Figure 11-78](#) and described in [Table 11-62](#). The EERH is shown in [Figure 11-79](#) and described in [Table 11-63](#).

**Figure 11-78. Event Enable Register (EER)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-62. Event Enable Register (EER) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event enable for events 0-31.<br>Event is not enabled. An external event latched in the event register (ER) is not evaluated by the EDMA3CC. |
|      |       | 1     | Event is enabled. An external event latched in the event register (ER) is evaluated by the EDMA3CC.  |

**Figure 11-79. Event Enable Register High (EERH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-63. Event Enable Register High (EERH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event enable for events 32-63.<br>Event is not enabled. An external event latched in the event register high (ERH) is not evaluated by the EDMA3CC. |
|      |       | 1     | Event is enabled. An external event latched in the event register high (ERH) is evaluated by the EDMA3CC.   |

#### 11.4.1.6.6 Event Enable Clear Register (EECR, EECRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable clear registers (EECR/EECRH) are used to disable events. Writes of 1 to the bits in EECR/EECRH clear the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EECR is shown in [Figure 11-80](#) and described in [Table 11-64](#). The EECRH is shown in [Figure 11-81](#) and described in [Table 11-65](#).

**Figure 11-80. Event Enable Clear Register (EECR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-64. Event Enable Clear Register (EECR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event enable clear for events 0-31.<br>No effect.  |
|      |       | 1     | Event is disabled. Corresponding bit in the event enable register (EER) is cleared (En = 0). |

**Figure 11-81. Event Enable Clear Register High (EECRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-65. Event Enable Clear Register High (EECRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Event enable clear for events 32-63.<br>No effect.   |
|      |       | 1     | Event is disabled. Corresponding bit in the event enable register high (EERH) is cleared (En = 0). |

#### 11.4.1.6.7 Event Enable Set Registers (EESR, EESRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable set registers (EESR/EESRH) are used to enable events. Writes of 1 to the bits in EESR/EESRH set the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EESR is shown in [Figure 11-82](#) and described in [Table 11-66](#). The EESRH is shown in [Figure 11-83](#) and described in [Table 11-67](#).

**Figure 11-82. Event Enable Set Register (EESR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-66. Event Enable Set Register (EESR) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event enable set for events 0-31.<br>No effect.   |
|      |       | 1     | Event is enabled. Corresponding bit in the event enable register (EER) is set (En = 1). |

**Figure 11-83. Event Enable Set Register High (EESRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-67. Event Enable Set Register High (EESRH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Event enable set for events 32-63.<br>No effect.  |
|      |       | 1     | Event is enabled. Corresponding bit in the event enable register high (EERH) is set (En = 1). |

#### 11.4.1.6.8 Secondary Event Registers (SER, SERH)

The secondary event registers (SER/SERH) provide information on the state of a DMA channel or event (0 through 63). If the EDMA3CC receives a TR synchronization due to a manual-trigger, event-trigger, or chained-trigger source (ESR.En/ESRH.En = 1, ER.En/ERH.En = 1, or CER.En/CERH.En = 1), which results in the setting of a corresponding event bit in SER/SERH (SER.En/SERH.En = 1), it implies that the corresponding DMA event is in the queue.

Once a bit corresponding to an event is set in SER/SERH, the EDMA3CC does not prioritize additional events on the same DMA channel. Depending on the condition that lead to the setting of the SER bits, either the EDMA3CC hardware or the software (using SECR/SECRH) needs to clear the SER/SERH bits for the EDMA3CC to evaluate subsequent events (subsequent transfers) on the same channel. For additional conditions that can cause the secondary event registers to be set, see *EDMA Overview*.

The SER is shown in [Figure 11-84](#) and described in [Table 11-68](#). The SERH is shown in [Figure 11-85](#) and described in [Table 11-69](#).

**Figure 11-84. Secondary Event Register (SER)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-68. Secondary Event Register (SER) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Secondary event register. The secondary event register is used along with the event register (ER) to provide information on the state of an event. |
|      |       | 1     | Event is not currently stored in the event queue.  |
|      |       |       | Event is currently stored in the event queue. Event arbiter will not prioritize additional events.   |

**Figure 11-85. Secondary Event Register High (SERH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-69. Secondary Event Register High (SERH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Secondary event register. The secondary event register is used along with the event register high (ERH) to provide information on the state of an event. |
|      |       | 1     | Event is not currently stored in the event queue.  |
|      |       |       | Event is currently stored in the event queue. Event submission/prioritization logic will not prioritize additional events.                               |

#### 11.4.1.6.9 Secondary Event Clear Registers (SECR, SECRH)

The secondary event clear registers (SECR/SECRH) clear the status of the secondary event registers (SER/SERH). CPU writes of 1 clear the corresponding set bits in SER/SERH. Writes of 0 have no effect.

The SECR is shown in [Figure 11-86](#) and described in [Table 11-70](#). The SECRH is shown in [Figure 11-87](#) and described in [Table 11-71](#).

**Figure 11-86. Secondary Event Clear Register (SECR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E15 | E14 | E13 | E12 | E11 | E10 | E9  | E8  | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-70. Secondary Event Clear Register (SECR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Secondary event clear register.<br>No effect.                                |
|      |       | 1     | Corresponding bit in the secondary event register (SER) is cleared (En = 0). |

**Figure 11-87. Secondary Event Clear Register High (SECRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-71. Secondary Event Clear Register High (SECRH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Secondary event clear register.<br>No effect.                                       |
|      |       | 1     | Corresponding bit in the secondary event registers high (SERH) is cleared (En = 0). |

### 11.4.1.7 Interrupt Registers

All DMA/QDMA channels can be set to assert an EDMA3CC completion interrupt to the CPU on transfer completion, by appropriately configuring the PaRAM entry associated with the channels. The following set of registers is used for the transfer completion interrupt reporting/generating by the EDMA3CC. For more details on EDMA3CC completion interrupt generation, see *EDMA3 Interrupts*.

#### 11.4.1.7.1 Interrupt Enable Registers (IER, IERH)

Interrupt enable registers (IER/IERH) are used to enable/disable the transfer completion interrupt generation by the EDMA3CC for all DMA/QDMA channels. The IER/IERH cannot be written to directly. To set any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable set registers (IESR/IESRH). Similarly, to clear any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable clear registers (IECR/IECRH).

The IER is shown in [Figure 11-88](#) and described in [Table 11-72](#). The IERH is shown in [Figure 11-89](#) and described in [Table 11-73](#).

**Figure 11-88. Interrupt Enable Register (IER)**

| 31  | 30  | 29  | 28  | 27   | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I31 | I30 | I29 | I28 | I27I | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| R-0 | R-0 | R-0 | R-0 | R-0  | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15  | 14  | 13  | 12  | 11   | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I15 | I14 | I13 | I12 | I11  | I10 | I9  | I8  | I7  | I6  | I5  | I4  | I3  | I2  | I1  | I0  |
| R-0 | R-0 | R-0 | R-0 | R-0  | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-72. Interrupt Enable Register (IER) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Interrupt enable for channels 0-31.<br>Interrupt is not enabled. |
|      |       | 1     | Interrupt is enabled.  |
|      |       |       |  |

**Figure 11-89. Interrupt Enable Register High (IERH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-73. Interrupt Enable Register High (IERH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Interrupt enable for channels 32-63.<br>Interrupt is not enabled. |
|      |       | 1     | Interrupt is enabled.   |
|      |       |       |   |

#### 11.4.1.7.2 Interrupt Enable Clear Register (IECR, IECRH)

The interrupt enable clear registers (IECR/IECRH) are used to clear interrupts. Writes of 1 to the bits in IECR/IECRH clear the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IEGR is shown in [Figure 11-90](#) and described in [Table 11-74](#). The IECRH is shown in [Figure 11-91](#) and described in [Table 11-75](#).

**Figure 11-90. Interrupt Enable Clear Register (IECR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I15 | I14 | I13 | I12 | I11 | I10 | I9  | I8  | I7  | I6  | I5  | I4  | I3  | I2  | I1  | I0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-74. Interrupt Enable Clear Register (IECR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Interrupt enable clear for channels 0-31.<br>No effect.                            |
|      |       | 1     | Corresponding bit in the interrupt enable register (IER) is cleared ( $I_n = 0$ ). |

**Figure 11-91. Interrupt Enable Clear Register High (IECRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-75. Interrupt Enable Clear Register High (IECRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | En    | 0     | Interrupt enable clear for channels 32-63.<br>No effect.                                 |
|      |       | 1     | Corresponding bit in the interrupt enable register high (IERH) is cleared ( $I_n = 0$ ). |

#### 11.4.1.7.3 Interrupt Enable Set Registers (IESR, IESRH)

The interrupt enable set registers (IESR/IESRH) are used to enable interrupts. Writes of 1 to the bits in IESR/IESRH set the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IESR is shown in [Figure 11-92](#) and described in [Table 11-76](#). The IESRH is shown in [Figure 11-93](#) and described in [Table 11-77](#).

**Figure 11-92. Interrupt Enable Set Register (IESR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I15 | I14 | I13 | I12 | I11 | I10 | I9  | I8  | I7  | I6  | I5  | I4  | I3  | I2  | I1  | I0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-76. Interrupt Enable Set Register (IESR) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Interrupt enable set for channels 0-31.<br>No effect.                     |
|      |       | 1     | Corresponding bit in the interrupt enable register (IER) is set (In = 1). |

**Figure 11-93. Interrupt Enable Set Register High (IESRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-77. Interrupt Enable Set Register High (IESRH) Field Descriptions**

| Bit  | Field | Value | Description   |
|------|-------|-------|---|
| 31-0 | En    | 0     | Interrupt enable clear for channels 32-63.<br>No effect.                        |
|      |       | 1     | Corresponding bit in the interrupt enable register high (IERH) is set (In = 1). |

#### 11.4.1.7.4 Interrupt Pending Register (IPR, IPRH)

If the TCINTEN and/or ITCINTEN bit in the channel option parameter (OPT) is set in the PaRAM entry associated with the channel (DMA or QDMA), then the EDMA3TC (for normal completion) or the EDMA3CC (for early completion) returns a completion code on transfer or intermediate transfer completion. The value of the returned completion code is equal to the TCC bit in OPT for the PaRAM entry associated with the channel.

When an interrupt transfer completion code with  $TCC = n$  is detected by the EDMA3CC, then the corresponding bit is set in the interrupt pending register (IPR.In, if  $n = 0$  to 31; IPRH.In, if  $n = 32$  to 63). Note that once a bit is set in the interrupt pending registers, it remains set; it is your responsibility to clear these bits. The bits set in IPR/IPRH are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR/ICRH).

The IPR is shown in [Figure 11-94](#) and described in [Table 11-78](#). The IPRH is shown in [Figure 11-95](#) and described in [Table 11-79](#).

**Figure 11-94. Interrupt Pending Register (IPR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I15 | I14 | I13 | I12 | I11 | I10 | I9  | I8  | I7  | I6  | I5  | I4  | I3  | I2  | I1  | I0  |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-78. Interrupt Pending Register (IPR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | In    | 0     | Interrupt pending for $TCC = 0$ -31.   |
|      |       | 1     | Interrupt transfer completion code is not detected or was cleared.                       |
|      |       |       | Interrupt transfer completion code is detected ( $In = 1$ , $n = \text{EDMA3TC}[5:0]$ ). |

**Figure 11-95. Interrupt Pending Register High (IPRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| R-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-79. Interrupt Pending Register High (IPRH) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | In    | 0     | Interrupt pending for $TCC = 32$ -63.  |
|      |       | 1     | Interrupt transfer completion code is not detected or was cleared.                       |
|      |       |       | Interrupt transfer completion code is detected ( $In = 1$ , $n = \text{EDMA3TC}[5:0]$ ). |

#### 11.4.1.7.5 Interrupt Clear Registers (ICR, ICRH)

The bits in the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR/ICRH). Writes of 0 have no effect. All set bits in IPR/IPRH must be cleared to allow EDMA3CC to assert additional transfer completion interrupts.

The ICR is shown in [Figure 11-96](#) and described in [Table 11-80](#). The ICRH is shown in [Figure 11-97](#) and described in [Table 11-81](#).

**Figure 11-96. Interrupt Clear Register (ICR)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I15 | I14 | I13 | I12 | I11 | I10 | I9  | I8  | I7  | I6  | I5  | I4  | I3  | I2  | I1  | I0  |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-80. Interrupt Clear Register (ICR) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | In    | 0     | Interrupt clear register for TCC = 0-31.<br>No effect.                         |
|      |       | 1     | Corresponding bit in the interrupt pending register (IPR) is cleared (In = 0). |

**Figure 11-97. Interrupt Clear Register High (ICRH)**

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 |

LEGEND: W = Write only; -n = value after reset

**Table 11-81. Interrupt Clear Register High (ICRH) Field Descriptions**

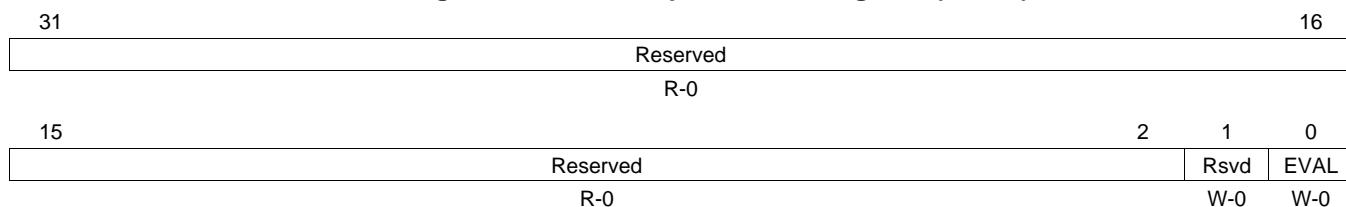
| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | In    | 0     | Interrupt clear register for TCC = 32-63.<br>No effect.                              |
|      |       | 1     | Corresponding bit in the interrupt pending register high (IPRH) is cleared (In = 0). |

#### 11.4.1.7.6 Interrupt Evaluate Register (IEVAL)

The interrupt evaluate register (IEVAL) is the only register that physically exists in both the global region and the shadow regions. In other words, the read/write accessibility for the shadow region IEVAL is not affected by the DMA/QDMA region access registers (DRAEm/DRAEHm, QRAEn/QRAEHn). IEVAL is needed for robust ISR operations to ensure that interrupts are not missed by the CPU.

The IEVAL is shown in [Figure 11-98](#) and described in [Table 11-82](#).

**Figure 11-98. Interrupt Evaluate Register (IEVAL)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-82. Interrupt Evaluate Register (IEVAL) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-2 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 0    | EVAL     | 0     | Interrupt evaluate.<br>No effect.  |
|      |          | 1     | Causes EDMA3CC completion interrupt to be pulsed, if any enabled (IERn/IERHn = 1) interrupts are still pending (IPRn/IPRHn = 1).<br><br>The EDMA3CC completion interrupt that is pulsed depends on which IEVAL is being exercised. For example, writing to the EVAL bit in IEVAL pulses the global completion interrupt, but writing to the EVAL bit in IEVAL0 pulses the region 0 completion interrupt. |

### 11.4.1.8 QDMA Registers

The following sets of registers control the QDMA channels in the EDMA3CC. The QDMA channels (with the exception of the QDMA queue number register) consist of a set of registers, each of which have a bit location. Each bit position corresponds to a QDMA channel number. The QDMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write accessibility in the shadow region address region is controlled by the QDMA region access registers (QRAEn/QRAEHn). *EDMA3 Channel Controller Regions* details shadow region/global region usage.

#### 11.4.1.8.1 QDMA Event Register (QER)

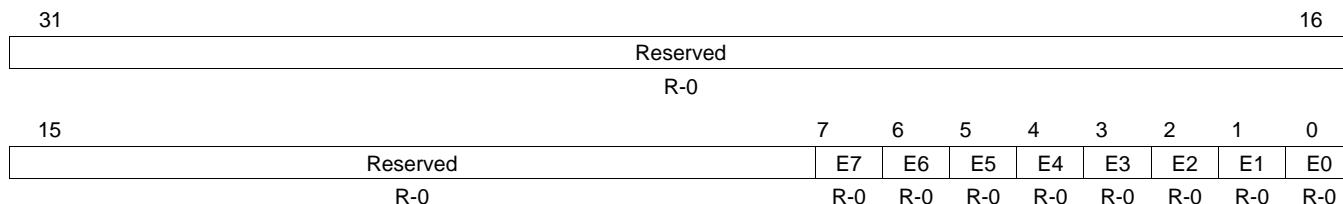
The QDMA event register (QER) channel  $n$  bit is set ( $En = 1$ ) when the CPU or any EDMA3 programmer (including EDMA3) performs a write to the trigger word (using the QDMA channel mapping register (QCHMAP $n$ )) in the PaRAM entry associated with QDMA channel  $n$  (which is also programmed using QCHMAP $n$ ). The  $En$  bit is also set when the EDMA3CC performs a link update on a PaRAM address that matches the QCHMAP $n$  settings. The QDMA event is latched only if the QDMA event enable register (QEER) channel  $n$  bit is also enabled (QEER. $En = 1$ ). Once a bit is set in QER, then the corresponding QDMA event (auto-trigger) is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. For additional conditions that can lead to the setting of QER bits, see *EDMA Overview*.

The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then the QDMA event missed register (QEMR) would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The set bits in QER are only cleared when the transfer request associated with the corresponding channels has been processed by the EDMA3CC and submitted to the transfer controller. If the  $En$  bit is already set and a QDMA event for the same QDMA channel occurs prior to the original being cleared, then the second missed event is latched in QEMR ( $En = 1$ ).

The QER is shown in [Figure 11-99](#) and described in [Table 11-83](#).

**Figure 11-99. QDMA Event Register (QER)**



LEGEND: R = Read only; -n = value after reset

**Table 11-83. QDMA Event Register (QER) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-0  | En       | 0     | QDMA event for channels 0-7.<br>No effect.  |
|      |          | 1     | Corresponding QDMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC.                             |

#### 11.4.1.8.2 QDMA Event Enable Register (QEER)

The EDMA3CC provides the option of selectively enabling/disabling each channel in the QDMA event register (QER) by using the QDMA event enable register (QEER). If any of the event bits in QEER is set (using the QDMA event enable set register, QEESR), it will enable that corresponding event. Alternatively, if any event bit in QEER is cleared (using the QDMA event enable clear register, QEECR), it will disable the corresponding QDMA channel. The QDMA event register will not latch any event for a QDMA channel, if it is not enabled via QEER.

The QEER is shown in [Figure 11-100](#) and described in [Table 11-84](#).

**Figure 11-100. QDMA Event Enable Register (QEER)**

|     |          |  |     |     |     |     |     |     |     |     |
|-----|----------|--|-----|-----|-----|-----|-----|-----|-----|-----|
| 31  | Reserved |  |     |     |     |     |     |     | 16  |     |
| R-0 |          |  |     |     |     |     |     |     |     |     |
| 15  | Reserved |  |     |     |     |     |     |     | 0   |     |
|     | R-0      |  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|     |          |  | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-84. QDMA Event Enable Register (QEER) Field Descriptions**

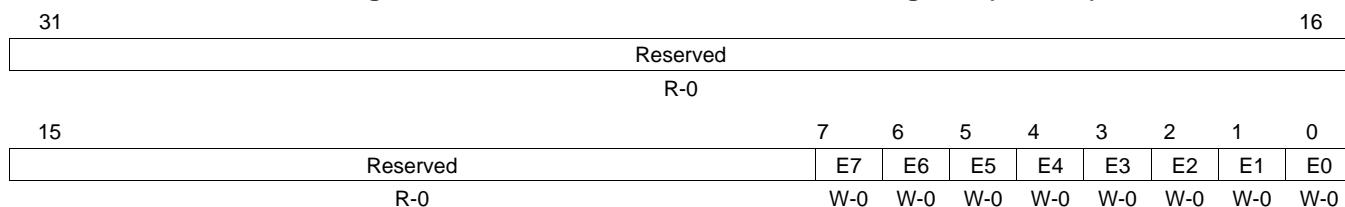
| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.                             |
| 7-0  | En       | 0     | QDMA event enable for channels 0-7.<br>QDMA channel <i>n</i> is not enabled. QDMA event will not be recognized and will not latch in the QDMA event register (QER). |
|      |          | 1     | QDMA channel <i>n</i> is enabled. QDMA events will be recognized and will get latched in the QDMA event register (QER).   |

#### 11.4.1.8.3 QDMA Event Enable Clear Register (QEECR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable clear register (QEECR) is used to disable events. Writes of 1 to the bits in QEECR clear the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEECR is shown in [Figure 11-101](#) and described in [Table 11-85](#).

**Figure 11-101. QDMA Event Enable Clear Register (QEECR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-85. QDMA Event Enable Clear Register (QEECR) Field Descriptions**

| Bit  | Field    | Value  | Description  |
|------|----------|--------|--|
| 31-8 | Reserved | 0      | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.                            |
| 7-0  | En       | 0<br>1 | QDMA event enable clear for channels 0-7.<br>No effect.<br>QDMA event is disabled. Corresponding bit in the QDMA event enable register (QEER) is cleared (En = 0). |

#### 11.4.1.8.4 QDMA Event Enable Set Register (QEESR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable set register (QEESR) is used to enable events. Writes of 1 to the bits in QEESR set the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEESR is shown in [Figure 11-102](#) and described in [Table 11-86](#).

**Figure 11-102. QDMA Event Enable Set Register (QEESR)**

|    |          |     |     |     |     |     |     |     |     |
|----|----------|-----|-----|-----|-----|-----|-----|-----|-----|
| 31 |          |     |     |     |     |     |     |     | 16  |
|    | Reserved |     |     |     |     |     |     |     |     |
|    | R-0      |     |     |     |     |     |     |     |     |
| 15 |          | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|    | Reserved | E7  | E6  | E5  | E4  | E3  | E2  | E1  | E0  |
|    | R-0      | W-0 |

LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-86. QDMA Event Enable Set Register (QEESR) Field Descriptions**

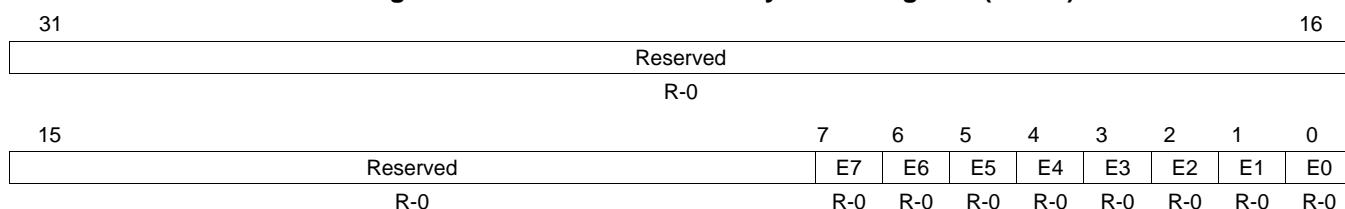
| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-0  | En       | 0     | QDMA event enable set for channels 0-7.   |
|      |          | 1     | No effect.<br>QDMA event is enabled. Corresponding bit in the QDMA event enable register (QEER) is set (En = 1).                        |

#### 11.4.1.8.5 QDMA Secondary Event Register (QSER)

The QDMA secondary event register (QSER) provides information on the state of a QDMA event. If at any time a bit corresponding to a QDMA channel is set in QSER, that implies that the corresponding QDMA event is in the queue. Once a bit corresponding to a QDMA channel is set in QSER, the EDMA3CC does not prioritize additional events on the same QDMA channel. Depending on the condition that lead to the setting of the QSER bits, either the EDMA3CC hardware or the software (using QSECR) needs to clear the QSER bits for the EDMA3CC to evaluate subsequent QDMA events on the channel. Based on whether the associated TR request is valid, or it is a null or dummy TR, the implications on the state of QSER and the required user actions to submit another QDMA transfer might be different. For additional conditions that can cause the secondary event registers (QSER\SER) to be set, see *EDMA Overview*.

The QSER is shown in [Figure 11-103](#) and described in [Table 11-87](#).

**Figure 11-103. QDMA Secondary Event Register (QSER)**



LEGEND: R = Read only; -n = value after reset

**Table 11-87. QDMA Secondary Event Register (QSER) Field Descriptions**

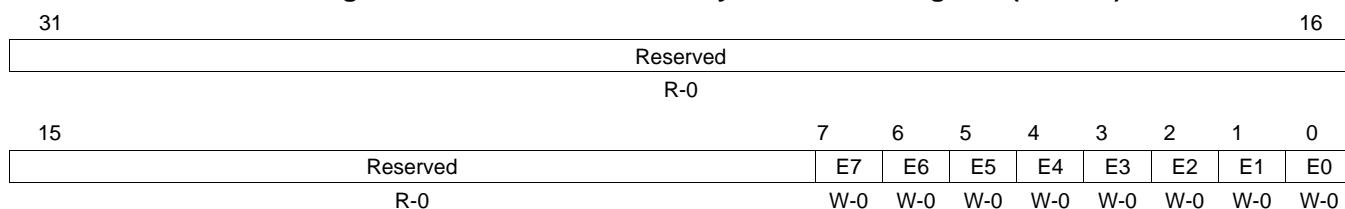
| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7-0  | En       | 0     | QDMA secondary event register for channels 0-7.<br>QDMA event is not currently stored in the event queue.                               |
|      |          | 1     | QDMA event is currently stored in the event queue. EDMA3CC will not prioritize additional events.                                       |

#### 11.4.1.8.6 QDMA Secondary Event Clear Register (QSECR)

The QDMA secondary event clear register (QSECR) clears the status of the QDMA secondary event register (QSER) and the QDMA event register (QER). CPU writes of 1 clear the corresponding set bits in QSER and QER. Writes of 0 have no effect. Note that this differs from the secondary event clear register (SECR) operation, which only clears the secondary event register (SER) bits and does not affect the event registers.

The QSECR is shown in [Figure 11-104](#) and described in [Table 11-88](#).

**Figure 11-104. QDMA Secondary Event Clear Register (QSECR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-88. QDMA Secondary Event Clear Register (QSECR) Field Descriptions**

| Bit  | Field    | Value  | Description   |
|------|----------|--------|---|
| 31-8 | Reserved | 0      | Reserved  |
| 7-0  | En       | 0<br>1 | QDMA secondary event clear register for channels 0-7.<br>No effect.<br>Corresponding bit in the QDMA secondary event register (QSER) and the QDMA event register (QER) is cleared (En = 0). |

### 11.4.2 EDMA3 Transfer Controller Registers

Table 11-89 lists the memory-mapped registers for the EDMA3 transfer controller (EDMA3TC).

**Table 11-89. EDMA3TC Registers**

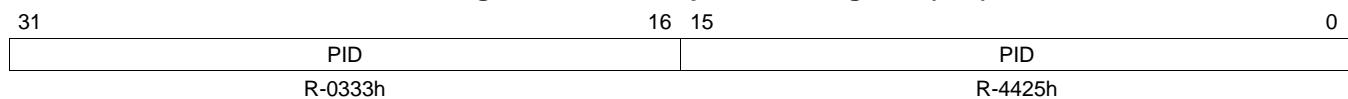
| Offset | Acronym   | Register Description  | Section                             |
|--------|-----------|---|-------------------------------------|
| 00h    | PID       | Peripheral Identification Register                            | <a href="#">Section 11.4.2.1</a>    |
| 04h    | TCCFG     | EDMA3TC Configuration Register                                | <a href="#">Section 11.4.2.2</a>    |
| 0100h  | TCSTAT    | EDMA3TC Channel Status Register                               | <a href="#">Section 11.4.2.3</a>    |
| 0120h  | ERRSTAT   | Error Register  | <a href="#">Section 11.4.2.4.1</a>  |
| 0124h  | ERREN     | Error Enable Register   | <a href="#">Section 11.4.2.4.2</a>  |
| 0128h  | ERRCLR    | Error Clear Register  | <a href="#">Section 11.4.2.4.3</a>  |
| 012Ch  | ERRDET    | Error Details Register  | <a href="#">Section 11.4.2.4.4</a>  |
| 0130h  | ERRCMD    | Error Interrupt Command Register                              | <a href="#">Section 11.4.2.4.5</a>  |
| 0140h  | RDRATE    | Read Rate Register  | <a href="#">Section 11.4.2.5</a>    |
| 0240h  | SAOPT     | Source Active Options Register                                | <a href="#">Section 11.4.2.6.1</a>  |
| 0244h  | SASRC     | Source Active Source Address Register                         | <a href="#">Section 11.4.2.6.2</a>  |
| 0248h  | SACNT     | Source Active Count Register                                  | <a href="#">Section 11.4.2.6.3</a>  |
| 024Ch  | SADST     | Source Active Destination Address Register                    | <a href="#">Section 11.4.2.6.4</a>  |
| 0250h  | SABIDX    | Source Active Source B-Index Register                         | <a href="#">Section 11.4.2.6.5</a>  |
| 0254h  | SAMPPRXY  | Source Active Memory Protection Proxy Register                | <a href="#">Section 11.4.2.6.6</a>  |
| 0258h  | SACNTRLD  | Source Active Count Reload Register                           | <a href="#">Section 11.4.2.6.7</a>  |
| 025Ch  | SASRCBREF | Source Active Source Address B-Reference Register             | <a href="#">Section 11.4.2.6.8</a>  |
| 0260h  | SADSTBREF | Source Active Destination Address B-Reference Register        | <a href="#">Section 11.4.2.6.9</a>  |
| 0280h  | DFCNTRLD  | Destination FIFO Set Count Reload                             | <a href="#">Section 11.4.2.6.16</a> |
| 0284h  | DFSRCBREF | Destination FIFO Set Destination Address B Reference Register | <a href="#">Section 11.4.2.6.17</a> |
| 0288h  | DFDSTBREF | Destination FIFO Set Destination Address B Reference Register | <a href="#">Section 11.4.2.6.18</a> |
| 0300h  | DFOPT0    | Destination FIFO Options Register 0                           | <a href="#">Section 11.4.2.6.10</a> |
| 0304h  | DFSRC0    | Destination FIFO Source Address Register 0                    | <a href="#">Section 11.4.2.6.11</a> |
| 0308h  | DFCNT0    | Destination FIFO Count Register 0                             | <a href="#">Section 11.4.2.6.12</a> |
| 030Ch  | DFDST0    | Destination FIFO Destination Address Register 0               | <a href="#">Section 11.4.2.6.13</a> |
| 0310h  | DFBIDX0   | Destination FIFO BIDX Register 0                              | <a href="#">Section 11.4.2.6.14</a> |
| 0314h  | DFMPPRXY0 | Destination FIFO Memory Protection Proxy Register 0           | <a href="#">Section 11.4.2.6.15</a> |
| 0340h  | DFOPT1    | Destination FIFO Options Register 1                           | <a href="#">Section 11.4.2.6.10</a> |
| 0344h  | DFSRC1    | Destination FIFO Source Address Register 1                    | <a href="#">Section 11.4.2.6.11</a> |
| 0348h  | DFCNT1    | Destination FIFO Count Register 1                             | <a href="#">Section 11.4.2.6.12</a> |
| 034Ch  | DFDST1    | Destination FIFO Destination Address Register 1               | <a href="#">Section 11.4.2.6.13</a> |
| 0350h  | DFBIDX1   | Destination FIFO BIDX Register 1                              | <a href="#">Section 11.4.2.6.14</a> |
| 0354h  | DFMPPRXY1 | Destination FIFO Memory Protection Proxy Register 1           | <a href="#">Section 11.4.2.6.15</a> |
| 0380h  | DFOPT2    | Destination FIFO Options Register 2                           | <a href="#">Section 11.4.2.6.10</a> |
| 0384h  | DFSRC2    | Destination FIFO Source Address Register 2                    | <a href="#">Section 11.4.2.6.11</a> |
| 0388h  | DFCNT2    | Destination FIFO Count Register 2                             | <a href="#">Section 11.4.2.6.12</a> |
| 038Ch  | DFDST2    | Destination FIFO Destination Address Register 2               | <a href="#">Section 11.4.2.6.13</a> |
| 0390h  | DFBIDX2   | Destination FIFO BIDX Register 2                              | <a href="#">Section 11.4.2.6.14</a> |
| 0394h  | DFMPPRXY2 | Destination FIFO Memory Protection Proxy Register 2           | <a href="#">Section 11.4.2.6.15</a> |
| 03C0h  | DFOPT3    | Destination FIFO Options Register 3                           | <a href="#">Section 11.4.2.6.10</a> |
| 03C4h  | DFSRC3    | Destination FIFO Source Address Register 3                    | <a href="#">Section 11.4.2.6.11</a> |
| 03C8h  | DFCNT3    | Destination FIFO Count Register 3                             | <a href="#">Section 11.4.2.6.12</a> |
| 03CCh  | DFDST3    | Destination FIFO Destination Address Register 3               | <a href="#">Section 11.4.2.6.13</a> |
| 03D0h  | DFBIDX3   | Destination FIFO BIDX Register 3                              | <a href="#">Section 11.4.2.6.14</a> |

**Table 11-89. EDMA3TC Registers (continued)**

| Offset | Acronym   | Register Description                                | Section                             |
|--------|-----------|---|-------------------------------------|
| 03D4h  | DFMPPRXY3 | Destination FIFO Memory Protection Proxy Register 3 | <a href="#">Section 11.4.2.6.15</a> |

#### 11.4.2.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) is a constant register that uniquely identifies the EDMA3TC and specific revision of the EDMA3TC. The PID is shown in [Figure 11-105](#) and described in [Table 11-90](#).

**Figure 11-105. Peripheral ID Register (PID)**

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

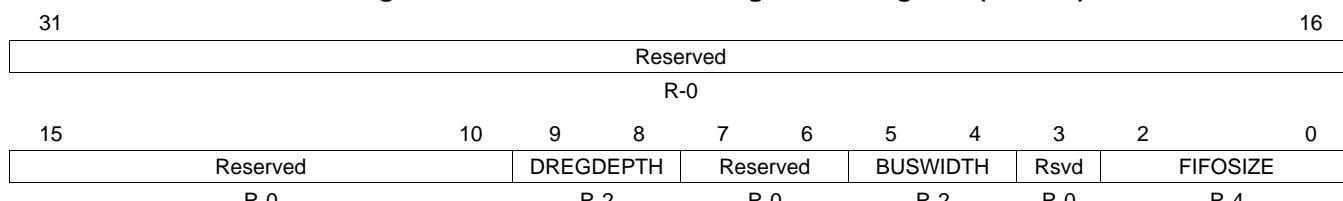
**Table 11-90. Peripheral ID Register (PID) Field Descriptions**

| Bit  | Field | Value        | Description            |
|------|-------|--------------|------------------------|
| 31-0 | PID   | 0xFFFF FFFFh | Peripheral identifier. |

#### 11.4.2.2 EDMA3TC Configuration Register (TCCFG)

The EDMA3TC configuration register (TCCFG) is shown in [Figure 11-106](#) and described in [Table 11-91](#).

**Figure 11-106. EDMA3TC Configuration Register (TCCFG)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset; -x = value is indeterminate after reset

**Table 11-91. EDMA3TC Configuration Register (TCCFG) Field Descriptions**

| Bit   | Field     | Value | Description                                       |
|-------|-----------|-------|---|
| 31-10 | Reserved  | 0     | Reserved.   |
| 9-8   | DREGDEPTH | 0-3h  | Destination register FIFO depth parameterization. |
|       |           | 0     | Reserved.   |
|       |           | 1h    | Reserved.   |
|       |           | 2h    | 4 entry (for TC0, TC1, TC2, and TC3)              |
|       |           | 3h    | Reserved.   |
| 7-6   | Reserved  | 0     | Reserved.   |
| 5-4   | BUSWIDTH  | 0-3h  | Bus width parameterization.                       |
|       |           | 0-1h  | Reserved.   |
|       |           | 2h    | 128-bit   |
|       |           | 3h    | Reserved.   |
| 3     | Reserved  | 0     | Reserved.   |
| 2-0   | FIFOSIZE  | 0-7h  | FIFO size   |
|       |           | 0-1h  | Reserved.   |
|       |           | 2h    | Reserved.   |
|       |           | 3h    | Reserved.   |
|       |           | 4h    | 512 byte FIFO                                     |
|       |           | 5h    | Reserved.   |
|       |           | 6h-7h | Reserved.   |

### 11.4.2.3 EDMA3TC Channel Status Register (TCSTAT)

The EDMA3TC channel status register (TCSTAT) is shown in [Figure 11-107](#) and described in [Table 11-92](#).

**Figure 11-107. EDMA3TC Channel Status Register (TCSTAT)**

|          |          |           |    |          |        |          |          |  |    |
|----------|----------|-----------|----|----------|--------|----------|----------|--|----|
| 31       | Reserved |           |    |          |        |          |          |  | 16 |
| R-0      |          |           |    |          |        |          |          |  |    |
| 15       | 14       | 13        | 12 | 11       | 9      | 8        |          |  |    |
| Reserved |          | DFSTRTPTR |    | Reserved |        | Reserved |          |  |    |
| R-0      |          | R-0       |    | R-0      |        | R-1      |          |  |    |
| 7        | 6        | 4         | 3  | 2        | 1      | 0        |          |  |    |
| Reserved |          | DSTACTV   |    | Reserved | WSACTV | SRCACTV  | PROGBUSY |  |    |
| R-0      |          | R-0       |    | R-0      | R-0    | R-0      | R-0      |  |    |

LEGEND: R = Read only; -n = value after reset

**Table 11-92. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions**

| Bit   | Field     | Value | Description   |
|-------|-----------|-------|---|
| 31-14 | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 13-12 | DFSTRTPTR | 0-3h  | Destination FIFO start pointer. Represents the offset to the head entry of the destination register FIFO, in units of entries.  |
| 11-9  | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 8     | Reserved  | 1     | Reserved. Always read as 1.   |
| 7     | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 6-4   | DSTACTV   | 0-7h  | Destination active state. Specifies the number of transfer requests (TRs) that are resident in the destination register FIFO at a given instant. This bit field can be primarily used for advanced debugging. Legal values are constrained by the destination register FIFO depth parameterization (DSTREGDEPTH) parameter.<br><br>0 Destination FIFO is empty.<br>1h Destination FIFO contains 1 TR.<br>2h Destination FIFO contains 2 TRs.<br>3h Destination FIFO contains 3 TRs.<br>4h Destination FIFO contains 4 TRs. (Full if DSTREGDEPTH==4).<br><br>If the destination register FIFO is empty, then any TR written to Prog Set immediately transitions to the destination register FIFO. If the destination register FIFO is not empty and not full, then any TR written to Prog Set immediately transitions to the destination register FIFO set if the source active state (SRCACTV) bit is set to idle.<br><br>If the destination register FIFO is full, then TRs cannot transition to the destination register FIFO. The destination register FIFO becomes not full when the TR at the head of the destination register FIFO is completed.<br><br>5h-7h Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 3     | Reserved  | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 2     | WSACTV    | 0     | Write status active   |
|       |           | 1     | Write status is not pending. Write status has been received for all previously issued write commands.   |
| 1     | SRCACTV   | 0     | Source active state   |
|       |           | 1     | Source controller is idle. Source active register set contains a previously processed transfer request.   |
|       |           |       | Source controller is busy servicing a transfer request.   |

**Table 11-92. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions (continued)**

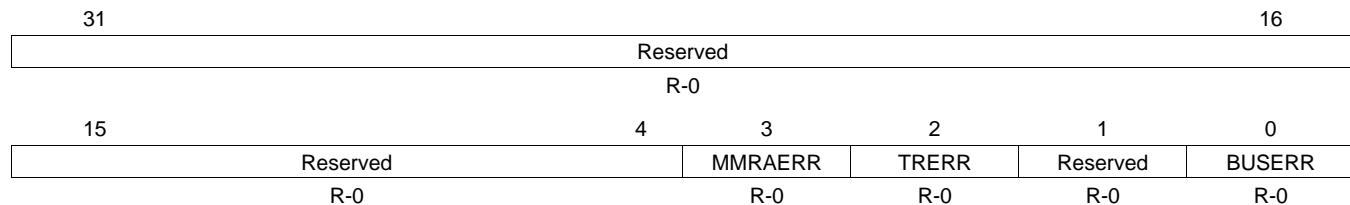
| Bit | Field    | Value | Description  |
|-----|----------|-------|--|
| 0   | PROGBUSY | 0     | Program register set busy<br>Program set idle and is available for programming by the EDMA3CC. |

### 11.4.2.4 Error Registers

#### 11.4.2.4.1 Error Register (ERRSTAT)

The error status register (ERRSTAT) is shown in [Figure 11-108](#) and described in [Table 11-93](#).

**Figure 11-108. Error Register (ERRSTAT)**



LEGEND: R = Read only; -n = value after reset

**Table 11-93. Error Register (ERRSTAT) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-4 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 3    | MMRAERR  | 0     | MMR address error.<br>Condition is not detected.  |
|      |          | 1     | User attempted to read or write to an invalid address in configuration memory map.  |
| 2    | TRERR    | 0     | Transfer request (TR) error event.<br>Condition is not detected.  |
|      |          | 1     | TR detected that violates constant addressing mode transfer (SAM or DAM is set) alignment rules or has ACNT or BCNT == 0.               |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0    | BUSERR   | 0     | Bus error event.<br>Condition is not detected.  |
|      |          | 1     | EDMA3TC has detected an error at source or destination address. Error information can be read from the error details register (ERRDET). |

#### 11.4.2.4.2 Error Enable Register (ERREN)

The error enable register (ERREN) is shown in [Figure 11-109](#) and described in [Table 11-94](#).

When any of the enable bits are set, a bit set in the corresponding ERRSTAT causes an assertion of the EDMA3TC interrupt.

**Figure 11-109. Error Enable Register (ERREN)**

|    |          |   |         |       |          |        |
|----|----------|---|---------|-------|----------|--------|
| 31 |          |   |         |       |          | 16     |
|    | Reserved |   |         |       |          |        |
|    | R-0      |   |         |       |          |        |
| 15 |          | 4 | 3       | 2     | 1        | 0      |
|    | Reserved |   | MMRAERR | TRERR | Reserved | BUSERR |
|    | R-0      |   | R/W-0   | R/W-0 | R/W-0    | R/W-0  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

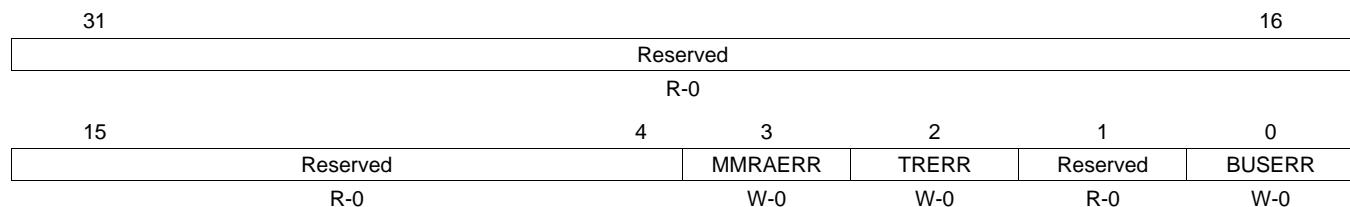
**Table 11-94. Error Enable Register (ERREN) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-4 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 3    | MMRAERR  | 0     | Interrupt enable for MMR address error (MMRAERR).   |
|      |          | 1     | MMRAERR is disabled.  |
| 2    | TRERR    | 0     | Interrupt enable for transfer request error (TRERR).  |
|      |          | 1     | TRERR is disabled.  |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
|      |          | 1     | TRERR is enabled and contributes to the state of EDMA3TC error interrupt generation.  |
| 0    | BUSERR   | 0     | Interrupt enable for bus error (BUSERR).  |
|      |          | 1     | BUSERR is disabled.   |
|      |          |       | BUSERR is enabled and contributes to the state of EDMA3TC error interrupt generation.   |

#### 11.4.2.4.3 Error Clear Register (ERRCLR)

The error clear register (ERRCLR) is shown in [Figure 11-110](#) and described in [Table 11-95](#).

**Figure 11-110. Error Clear Register (ERRCLR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-95. Error Clear Register (ERRCLR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-4 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 3    | MMRAERR  | 0     | Interrupt enable clear for the MMRAERR bit in the error status register (ERRSTAT).  |
|      |          | 1     | No effect.<br>Clears the MMRAERR bit in ERRSTAT but does not clear the error details register (ERRDET).                                 |
| 2    | TRERR    | 0     | Interrupt enable clear for the TRERR bit in the error status register (ERRSTAT).  |
|      |          | 1     | No effect.<br>Clears the TRERR bit in ERRSTAT but does not clear the error details register (ERRDET).                                   |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0    | BUSERR   | 0     | Interrupt clear for the BUSERR bit in the error status register (ERRSTAT).  |
|      |          | 1     | No effect.<br>Clears the BUSERR bit in ERRSTAT and clears the error details register (ERRDET).  |

#### 11.4.2.4.4 Error Details Register (ERRDET)

The error details register (ERRDET) is shown in [Figure 11-111](#) and described in [Table 11-96](#).

**Figure 11-111. Error Details Register (ERRDET)**

| 31       | Reserved |    |     |   |   |     |   |   | 18       | 17     | 16      |
|----------|----------|----|-----|---|---|-----|---|---|----------|--------|---------|
|          |          |    |     |   |   |     |   |   |          | TCCHEN | TCINTEN |
|          | R-0      |    |     |   |   |     |   |   | R-0      | R-0    |         |
| 15       | 14       | 13 |     | 8 | 7 |     | 4 | 3 |          | 0      |         |
| Reserved | TCC      |    |     |   |   |     |   |   | Reserved | STAT   |         |
|          | R-0      |    | R-0 |   |   | R-0 |   |   | R-0      |        |         |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

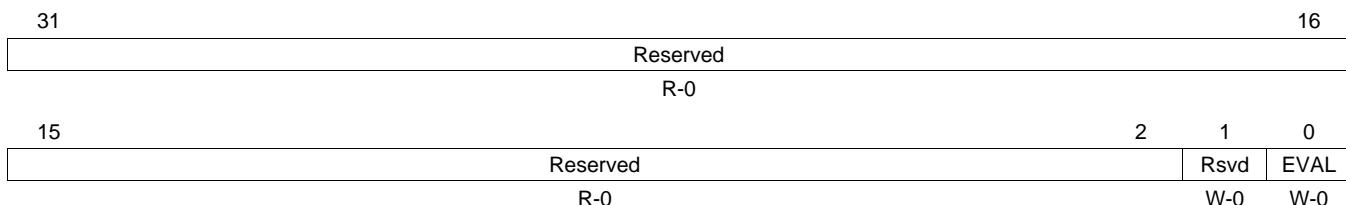
**Table 11-96. Error Details Register (ERRDET) Field Descriptions**

| Bit    | Field    | Value | Description  |
|--------|----------|-------|--|
| 31-8   | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 17     | TCCHEN   | 0-1   | Transfer completion chaining enable. Contains the TCCHEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.  |
| 16     | TCINTEN  | 0-1   | Transfer completion interrupt enable. Contains the TCINTEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.  |
| 15-14  | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 13 - 8 | TCC      | 0-3Fh | Transfer complete code. Contains the TCC value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error.  |
| 7-4    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 3-0    | STAT     | 0-Fh  | Transaction status. Stores the nonzero status/error code that was detected on the read status or write status bus. If read status and write status are returned on the same cycle, then the EDMA3TC chooses nonzero version. If both are nonzero, then the write status is treated as higher priority. |
|        |          | 0     | No error.  |
|        |          | 1h-7h | Read error.  |
|        |          | 8h-Fh | Write error.   |

#### 11.4.2.4.5 Error Interrupt Command Register (ERRCMD)

The error command register (ERRCMD) is shown in [Figure 11-112](#) and described in [Table 11-97](#).

**Figure 11-112. Error Interrupt Command Register (ERRCMD)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 11-97. Error Interrupt Command Register (ERRCMD) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-2 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 1    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0    | EVAL     | 0     | Error evaluate<br>No effect   |
|      |          | 1     | EDMA3TC error line is pulsed if any of the error status register (ERRSTAT) bits are set.  |

#### 11.4.2.5 Read Rate Register (RDRATE)

The EDMA3 transfer controller issues read commands at a rate controlled by the read rate register (RDRATE). The RDRATE defines the number of idle cycles that the read controller must wait before issuing subsequent commands. This applies both to commands within a transfer request packet (TRP) and for commands that are issued for different transfer requests (TRs). For instance, if RDRATE is set to 4 cycles between reads, there are 3 inactive cycles between reads.

RDRATE allows flexibility in transfer controller access requests to an endpoint. For an application, RDRATE can be manipulated to slow down the access rate, so that the endpoint may service requests from other masters during the inactive EDMA3TC cycles.

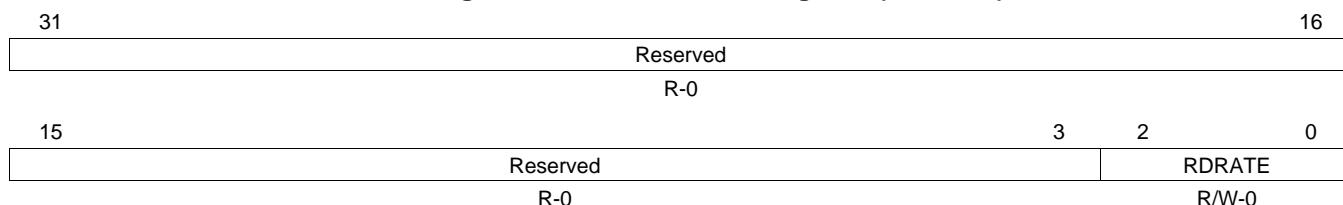
The RDRATE is shown in [Figure 11-113](#) and described in [Table 11-98](#).

---

**NOTE:** It is expected that the RDRATE value for a transfer controller is static, as it is decided based on the application requirement. It is not recommended to change this setting on the fly.

---

**Figure 11-113. Read Rate Register (RDRATE)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-98. Read Rate Register (RDRATE) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-3 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 2-0  | RDRATE   | 0-7h  | Read rate. Controls the number of cycles between read commands. This is a global setting that applies to all TRs for this EDMA3TC.<br>0      Reads issued as fast as possible.<br>1h     4 cycles between reads.<br>2h     8 cycles between reads.<br>3h     16 cycles between reads.<br>4h     32 cycles between reads.<br>5h-7h   Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |

### 11.4.2.6 EDMA3TC Channel Registers

The EDMA3TC channel registers are split into three parts: the programming registers, the source active registers, and the destination FIFO register. This section describes the registers and their functions. The program register set is programmed by the channel controller, and is for internal use. The other two sets are read-only and provided to facilitate advanced debug capabilities. The number of destination FIFO register sets depends on the destination FIFO depth.

TC0 has a FIFO depth of 2, so there are two sets of destination FIFO registers associated with TC0. TC1, TC2, and TC3 have a destination FIFO depth of 4, so there are four sets of destination FIFO registers for each of these transfer controllers.

#### 11.4.2.6.1 Source Active Options Register (SAOPT)

The source active options register (SAOPT) is shown in [Figure 11-114](#) and described in [Table 11-99](#).

**Figure 11-114. Source Active Options Register (SAOPT)**

| 31 |          |      |       |      |   |       |  |  | 23       | 22     | 21    | 20      | 19       | 18 | 17    | 16 |
|----|----------|------|-------|------|---|-------|--|--|----------|--------|-------|---------|----------|----|-------|----|
|    | Reserved |      |       |      |   |       |  |  |          | TCCHEN | Rsvd  | TCINTEN | Reserved |    | TCC   |    |
|    | R-0      |      |       |      |   |       |  |  |          | R/W-0  | R-0   | R/W-0   | R-0      |    | R/W-0 |    |
| 15 | 12       | 11   | 10    | 8    | 7 | 6     |  |  | 4        | 3      | 2     | 1       | 0        |    |       |    |
|    | TCC      | Rsvd | FWID  | Rsvd |   | PRI   |  |  | Reserved | DAM    | SAM   |         |          |    |       |    |
|    | R/W-0    | R-0  | R/W-0 | R-0  |   | R/W-0 |  |  | R-0      | R/W-0  | R/W-0 |         |          |    |       |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-99. Source Active Options Register (SAOPT) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-23 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 22    | TCCHEN   | 0     | Transfer complete chaining enable   |
|       |          | 1     | Transfer complete chaining is disabled.   |
| 21    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 20    | TCINTEN  | 0     | Transfer complete interrupt enable.   |
|       |          | 1     | Transfer complete interrupt is disabled.  |
| 19-18 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 17-12 | TCC      | 0-3Fh | Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3PCC module.                           |
| 11    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 10-8  | FWID     | 0-7h  | FIFO width. Applies if either SAM or DAM is set to constant addressing mode.  |
|       |          | 0     | FIFO width is 8-bit.  |
|       |          | 1h    | FIFO width is 16-bit.   |
|       |          | 2h    | FIFO width is 32-bit.   |
|       |          | 3h    | FIFO width is 64-bit.   |
|       |          | 4h    | FIFO width is 128-bit.  |
|       |          | 5h    | FIFO width is 256-bit.  |
|       |          | 6h-7h | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7     | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |

**Table 11-99. Source Active Options Register (SAOPT) Field Descriptions (continued)**

| <b>Bit</b> | <b>Field</b> | <b>Value</b>             | <b>Description</b>   |
|------------|--------------|--------------------------|--|
| 6-4        | PRI          | 0-7h<br>0<br>1h-6h<br>7h | Transfer priority. Reflects the values programmed in the QUEPRI register in the EDMACC.<br>Priority 0 - Highest priority<br>Priority 1 to priority 6<br>Priority 7 - Lowest priority   |
| 3-2        | Reserved     | 0                        | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.  |
| 1          | DAM          | 0<br>1                   | Destination address mode within an array<br>Increment (INCR) mode. Destination addressing within an array increments.<br>Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width. |
| 0          | SAM          | 0<br>1                   | Source address mode within an array<br>Increment (INCR) mode. Source addressing within an array increments.<br>Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.                |

#### 11.4.2.6.2 Source Active Source Address Register (SASRC)

The source active source address register (SASRC) is shown in [Figure 11-115](#) and described in [Table 11-100](#).

**Figure 11-115. Source Active Source Address Register (SASRC)**

|    |       |     |
|----|-------|-----|
| 31 | SADDR | 0   |
|    |       | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-100. Source Active Source Address Register (SASRC) Field Descriptions**

| Bit  | Field | Value        | Description   |
|------|-------|--------------|---|
| 31-0 | SADDR | 0xFFFF FFFFh | Source address for program register set. EDMA3TC updates value according to source addressing mode (SAM bit in the source active options register, SAOPT) . |

#### 11.4.2.6.3 Source Active Count Register (SACNT)

The source active count register (SACNT) is shown in [Figure 11-116](#) and described in [Table 11-101](#).

**Figure 11-116. Source Active Count Register (SACNT)**

|    |      |     |
|----|------|-----|
| 31 | BCNT | 16  |
|    |      | R-0 |
| 15 | ACNT | 0   |
|    |      | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 11-101. Source Active Count Register (SACNT) Field Descriptions**

| Bit   | Field | Value   | Description  |
|-------|-------|---------|--|
| 31-16 | BCNT  | 0xFFFFh | B dimension count. Number of arrays to be transferred, where each array is ACNT in length. It is decremented after each read command appropriately. Represents the amount of data remaining to be read. It should be 0 when transfer request (TR) is complete. |
| 15-0  | ACNT  | 0xFFFFh | A dimension count. Number of bytes to be transferred in first dimension. It is decremented after each read command appropriately. Represents the amount of data remaining to be read. It should be 0 when transfer request (TR) is complete.                   |

#### 11.4.2.6.4 Source Active Destination Address Register (SADST)

The source active destination address register (SADST) is shown in [Figure 11-117](#) and described in [Table 11-102](#).

**Figure 11-117. Source Active Destination Address Register (SADST)**

|    |          |    |
|----|----------|----|
| 31 | Reserved | 16 |
|    | R-0      |    |
| 15 | Reserved | 0  |
|    | R-0      |    |

LEGEND: R = Read only; -n = value after reset

**Table 11-102. Source Active Destination Address Register (SADST) Field Descriptions**

| Bit  | Field     | Value | Description                  |
|------|-----------|-------|------------------------------|
| 31-0 | Reserve d | 0     | Reserved. Always reads as 0. |

#### 11.4.2.6.5 Source Active Source B-Dimension Index Register (SABIDX)

The source active set B-dimension index register (SABIDX) is shown in [Figure 11-118](#) and described in [Table 11-103](#).

**Figure 11-118. Source Active Source B-Dimension Index Register (SABIDX)**

|    |       |    |
|----|-------|----|
| 31 | DBIDX | 16 |
|    | R-0   |    |
| 15 | SBIDX | 0  |
|    | R-0   |    |

LEGEND: R = Read only; -n = value after reset

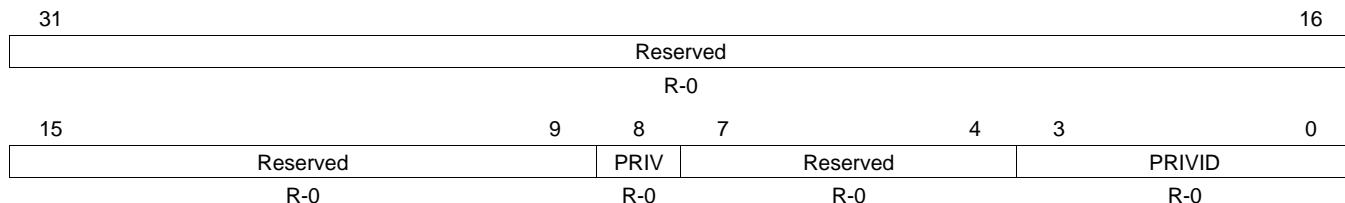
**Table 11-103. Source Active Source B-Dimension Index Register (SABIDX) Field Descriptions**

| Bit   | Field | Value   | Description  |
|-------|-------|---------|--|
| 31-16 | DBIDX | 0       | B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. Always reads as 0. |
| 15-0  | SBIDX | 0xFFFFh | B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array.                        |

#### 11.4.2.6.6 Source Active Memory Protection Proxy Register (SAMPPRXY)

The source active memory protection proxy register (SAMPPRXY) is shown in [Figure 11-119](#) and described in [Table 11-104](#).

**Figure 11-119. Source Active Memory Protection Proxy Register (SAMPPRXY)**



LEGEND: R = Read only; -n = value after reset

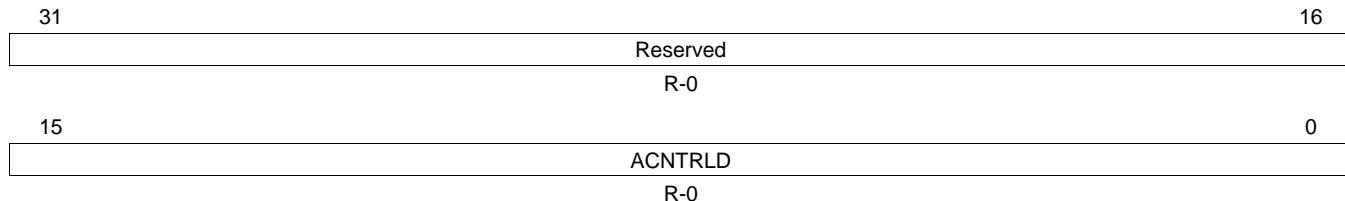
**Table 11-104. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-9 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 8    | PRIV     |       | Privilege level. The privilege level used by the host to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>The privilege ID is used while issuing read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.<br>0 User-level privilege.<br>1 Supervisor-level privilege. |
| 7-4  | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 3-0  | PRIVID   | 0-Fh  | Privilege ID. This contains the privilege ID of the host that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>This PRIVID value is used while issuing read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.  |

#### 11.4.2.6.7 Source Active Count Reload Register (SACNTRLD)

The source active count reload register (SACNTRLD) is shown in [Figure 11-120](#) and described in [Table 11-105](#).

**Figure 11-120. Source Active Count Reload Register (SACNTRLD)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

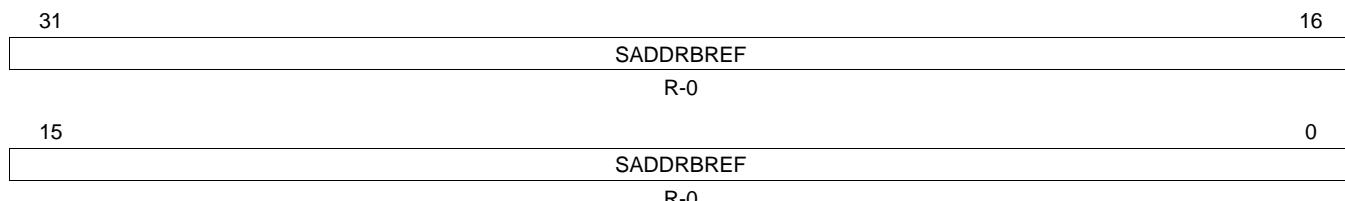
**Table 11-105. Source Active Count Reload Register (SACNTRLD) Field Descriptions**

| Bit   | Field    | Value   | Description   |
|-------|----------|---------|---|
| 31-16 | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.               |
| 15-0  | ACNTRLD  | 0xFFFFh | A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced. |

#### 11.4.2.6.8 Source Active Source Address B-Reference Register (SASRCBREF)

The source active source address B-reference register (SASRCBREF) is shown in [Figure 11-121](#) and described in [Table 11-106](#).

**Figure 11-121. Source Active Source Address B-Reference Register (SASRCBREF)**



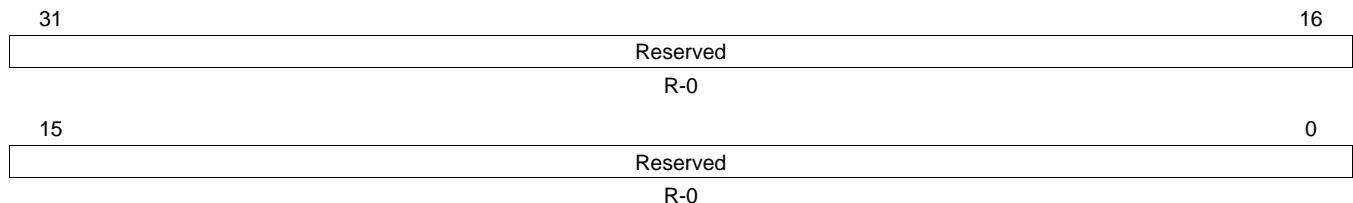
LEGEND: R = Read only; -n = value after reset

**Table 11-106. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions**

| Bit  | Field     | Value        | Description   |
|------|-----------|--------------|---|
| 31-0 | SADDRBREF | 0xFFFF FFFFh | Source address B-reference. Represents the starting address for the array currently being read. |

**11.4.2.6.9 Source Active Destination Address B-Reference Register (SADSTBREF)**

The source active destination address B-reference register (SADSTBREF) is shown in [Figure 11-122](#) and described in [Table 11-107](#).

**Figure 11-122. Source Active Destination Address B-Reference Register (SADSTBREF)**


LEGEND: R = Read only; -n = value after reset

**Table 11-107. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions**

| Bit  | Field    | Value | Description                  |
|------|----------|-------|------------------------------|
| 31-0 | Reserved | 0     | Reserved. Always reads as 0. |

#### 11.4.2.6.10 Destination FIFO Options Register (DFOPT $n$ )

The destination FIFO options register (DFOPT $n$ ) is shown in [Figure 11-123](#) and described in [Table 11-108](#).

**NOTE:** The value for  $n$  varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 11-123. Destination FIFO Options Register (DFOPT $n$ )**

| 31 | Reserved |      |       |      | 23     | 22   | 21      | 20       | 19    | 18    | 17 | 16 |
|----|----------|------|-------|------|--------|------|---------|----------|-------|-------|----|----|
|    |          |      |       |      | TCCHEN | Rsvd | TCINTEN | Reserved |       | TCC   |    |    |
|    | R-0      |      |       |      | R/W-0  | R-0  | R/W-0   | R-0      | R/W-0 |       |    |    |
| 15 | 12       | 11   | 10    | 8    | 7      | 6    |         | 4        | 3     | 2     | 1  | 0  |
|    | TCC      | Rsvd | FWID  | Rsvd | PRI    |      |         | Reserved | DAM   | SAM   |    |    |
|    | R/W-0    | R-0  | R/W-0 | R-0  | R/W-0  |      |         | R-0      | R/W-0 | R/W-0 |    |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 11-108. Destination FIFO Options Register (DFOPT $n$ ) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-23 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 22    | TCCHEN   | 0     | Transfer complete chaining enable   |
|       |          | 1     | Transfer complete chaining is disabled  |
|       |          | 1     | Transfer complete chaining is enabled   |
| 21    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 20    | TCINTEN  | 0     | Transfer complete interrupt enable.   |
|       |          | 1     | Transfer complete interrupt is disabled.  |
|       |          | 1     | Transfer complete interrupt is enabled.   |
| 19-18 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 17-12 | TCC      | 0-3Fh | Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3PCC module.                           |
| 11    | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 10-8  | FWID     | 0-7h  | FIFO width. Applies if either SAM or DAM is set to constant addressing mode.  |
|       |          | 0     | FIFO width is 8-bit.  |
|       |          | 1h    | FIFO width is 16-bit.   |
|       |          | 2h    | FIFO width is 32-bit.   |
|       |          | 3h    | FIFO width is 64-bit.   |
|       |          | 4h    | FIFO width is 128-bit.  |
|       |          | 5h    | FIFO width is 256-bit.  |
|       |          | 6h-7h | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 7     | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 6-4   | PRI      | 0-7h  | Transfer priority   |
|       |          | 0     | Priority 0 - Highest priority   |
|       |          | 1h-6h | Priority 1 to priority 6  |
|       |          | 7h    | Priority 7 - Lowest priority  |
| 3-2   | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |

**Table 11-108. Destination FIFO Options Register (DFOPT $n$ ) Field Descriptions (continued)**

| Bit | Field | Value | Description   |
|-----|-------|-------|---|
| 1   | DAM   | 0     | Destination address mode within an array<br>Increment (INCR) mode. Destination addressing within an array increments. |
|     |       | 1     | Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width.       |
| 0   | SAM   | 0     | Source address mode within an array<br>Increment (INCR) mode. Source addressing within an array increments.           |
|     |       | 1     | Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width.            |

#### 11.4.2.6.11 Destination FIFO Source Address Register (DFSRC $n$ )

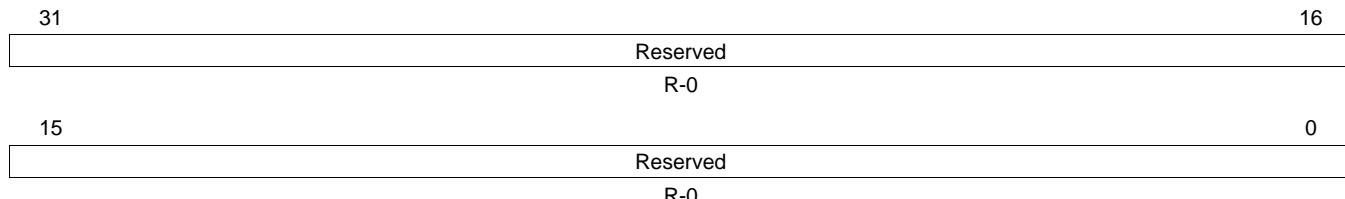
The destination FIFO source address register (DFSRC $n$ ) is shown in [Figure 11-124](#) and described in [Table 11-109](#).

---

**NOTE:** The value for  $n$  varies from 0 to DSTREGDEPTH for the given EDMA3TC.

---

**Figure 11-124. Destination FIFO Source Address Register (DFSRC $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

**Table 11-109. Destination FIFO Source Address Register (DFSRC $n$ ) Field Descriptions**

| Bit  | Field    | Value | Description                 |
|------|----------|-------|-----------------------------|
| 31-0 | Reserved | 0     | Reserved. Always read as 0. |

#### 11.4.2.6.12 Destination FIFO Count Register (DFCNT $n$ )

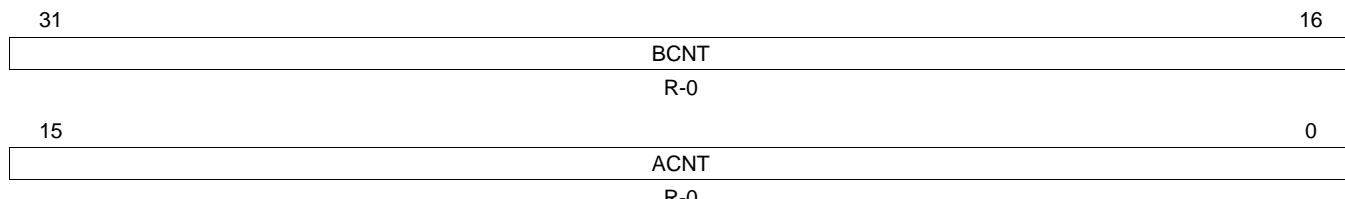
The destination FIFO count register (DFCNT $n$ ) is shown in [Figure 11-125](#) and described in [Table 11-110](#).

---

**NOTE:** The value for  $n$  varies from 0 to DSTREGDEPTH for the given EDMA3TC.

---

**Figure 11-125. Destination FIFO Count Register (DFCNT $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

**Table 11-110. Destination FIFO Count Register (DFCNT $n$ ) Field Descriptions**

| Bit   | Field | Value   | Description   |
|-------|-------|---------|---|
| 31-16 | BCNT  | 0-FFFFh | B-dimension count. Number of arrays to be transferred, where each array is ACNT in length. Count/count remaining for destination register set. Represents the amount of data remaining to be written. |
| 15-0  | ACNT  | 0-FFFFh | A-dimension count. Number of bytes to be transferred in first dimension count/count remaining for destination register set. Represents the amount of data remaining to be written.                    |

#### 11.4.2.6.13 Destination FIFO Destination Address Register (DFDST $n$ )

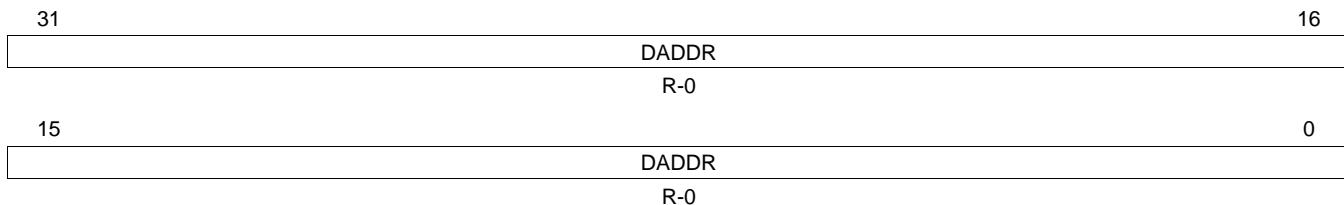
The destination FIFO destination address register (DFDST $n$ ) is shown in [Figure 11-126](#) and described in [Table 11-111](#).

---

**NOTE:** The value for  $n$  varies from 0 to DSTREGDEPTH for the given EDMA3TC.

---

**Figure 11-126. Destination FIFO Destination Address Register (DFDST $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

**Table 11-111. Destination FIFO Destination Address Register (DFDST $n$ ) Field Descriptions**

| Bit  | Field | Value | Description  |
|------|-------|-------|--|
| 31-0 | DADDR | 0     | Destination address for the destination FIFO register set. When a transfer request (TR) is complete, the final value should be the address of the last write command issued. |

#### 11.4.2.6.14 Destination FIFO B-Index Register (DFBIDX $n$ )

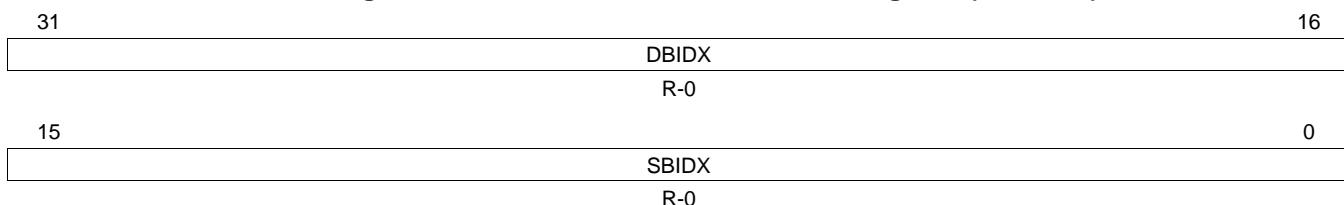
The destination FIFO B-index register (DFBIDX $n$ ) is shown in [Figure 11-127](#) and described in [Table 11-112](#).

---

**NOTE:** The value for  $n$  varies from 0 to DSTREGDEPTH for the given EDMA3TC.

---

**Figure 11-127. Destination FIFO B-Index Register (DFBIDX $n$ )**



LEGEND: R = Read only; - $n$  = value after reset

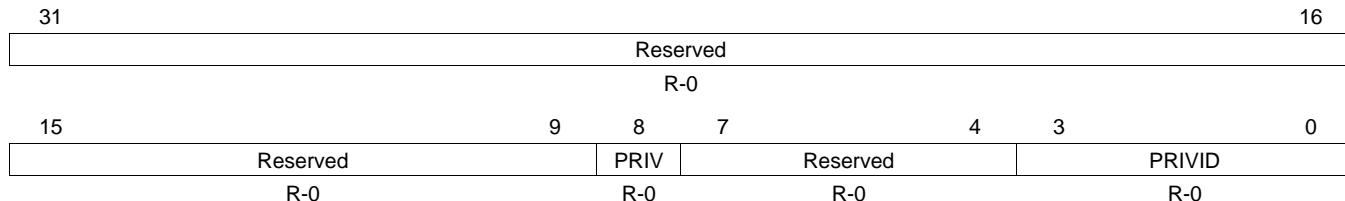
**Table 11-112. Destination FIFO B-Index Register (DFBIDX $n$ ) Field Descriptions**

| Bit   | Field | Value   | Description   |
|-------|-------|---------|---|
| 31-16 | DBIDX | 0-FFFFh | B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. |
| 15-0  | SBIDX | 0       | Always read as 0.   |

#### **11.4.2.6.15 Destination FIFO Memory Protection Proxy Register (DFMPPRXY $n$ )**

The destination FIFO memory protection proxy register (DFMPPRXY $n$ ) is shown in [Figure 11-128](#) and described in [Table 11-104](#).

**Figure 11-128. Destination FIFO Memory Protection Proxy Register (DFMPPRXY $n$ )**



LEGEND: R = Read only; -n = value after reset

**Table 11-113. Destination FIFO Memory Protection Proxy Register (DFMPPRXY $n$ ) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-9 | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 8    | PRIV     |       | Privilege level. This contains the Privilege level used by the EDMA3 programmer to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>The privilege ID is used while issuing read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction.<br>0 User-level privilege<br>1 Supervisor-level privilege |
| 7-4  | Reserved | 0     | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.   |
| 3-0  | PRIVID   | 0-Fh  | Privilege ID. This contains the Privilege ID of the EDMA3 programmer that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>This PRIVID value is used while issuing read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction.  |

#### 11.4.2.6.16 Destination FIFO Count Reload Register (DFCNTRLD $n$ )

The destination FIFO count reload register (DFCNTRLD $n$ ) is shown in [Figure 11-129](#) and described in [Table 11-114](#).

**Figure 11-129. Destination FIFO Count Reload Register (DFCNTRLD $n$ )**

|    |          |    |
|----|----------|----|
| 31 | Reserved | 16 |
|    | R-0      |    |
| 15 | ACNTRLD  | 0  |
|    | R-0      |    |

LEGEND: R = Read only; -n = value after reset

**Table 11-114. Destination FIFO Count Reload Register (DFCNTRLD $n$ ) Field Descriptions**

| Bit   | Field    | Value   | Description   |
|-------|----------|---------|---|
| 31-16 | Reserved | 0       | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior.               |
| 15-0  | ACNTRLD  | 0xFFFFh | A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced. |

#### 11.4.2.6.17 Destination FIFO Source Address B-Reference Register (DFSRCBREF $n$ )

The destination FIFO source address B-reference register (DFSRCBREF $n$ ) is shown in [Figure 11-130](#) and described in [Table 11-115](#).

**Figure 11-130. Destination FIFO Source Address B-Reference Register (DFSRCBREF $n$ )**

|    |          |    |
|----|----------|----|
| 31 | Reserved | 16 |
|    | R-0      |    |
| 15 | Reserved | 0  |
|    | R-0      |    |

LEGEND: R = Read only; -n = value after reset

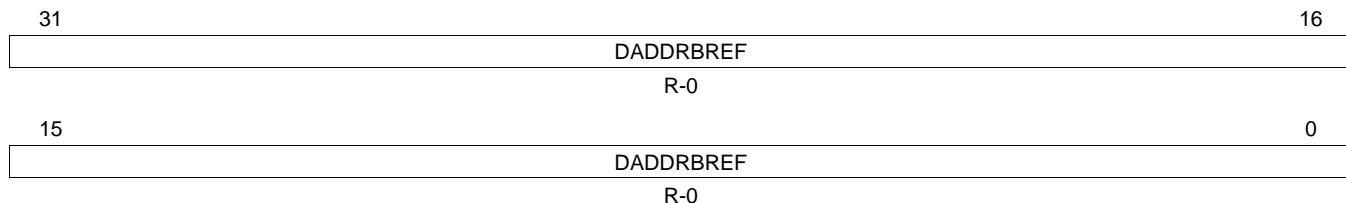
**Table 11-115. Destination FIFO Source Address B-Reference Register (DFSRCBREF $n$ ) Field Descriptions**

| Bit  | Field    | Value | Description                 |
|------|----------|-------|-----------------------------|
| 31-0 | Reserved | 0     | Reserved. Always read as 0. |

#### 11.4.2.6.18 Destination FIFO Destination Address B-Reference (DFDSTBREF $n$ )

The destination FIFO destination address B-reference register (DFDSTBREF $n$ ) is shown in [Figure 11-131](#) and described in [Table 11-116](#).

**Figure 11-131. Destination FIFO Destination Address B-Reference Register (DFDSTBREF $n$ )**



LEGEND: R = Read only; -n = value after reset

**Table 11-116. Destination FIFO Destination Address B-Reference Register (DFDSTBREF $n$ ) Field Descriptions**

| Bit  | Field     | Value        | Description   |
|------|-----------|--------------|---|
| 31-0 | DADDRBREF | 0xFFFF FFFFh | Destination address reference for the destination FIFO register set. Represents the starting address for the array currently being written. |

## 11.5 Appendix A

### 11.5.1 Debug Checklist

This section lists some tips to keep in mind while debugging applications using the EDMA3.

The following table provides some common issues and their probable causes and resolutions.

**Table 11-117. Debug List**

| Issue  | Description/Solution   |
|--|--|
| The transfer associated with the channel does not happen. The channel does not get serviced.         | <p>The EDMA3CC may not service a transfer request, even though the associated ParAM set is programmed appropriately. Check for the following:</p> <ol style="list-style-type: none"> <li>1) Verify that events are enabled, i.e., if an external/peripheral event is latched in Event Registers (ER/ERH), make sure that the event is enabled in the Event Enable Registers (EER/EERH). Similarly, for QDMA channels, make sure that QDMA events are appropriately enabled in the QDMA Event Enable Register (QEER).</li> <li>2) Verify that the DMA or QDMA Secondary Event Register (SER/SERH/QSERH) bits corresponding to the particular event or channel are not set.</li> </ol>   |
| The Secondary Event Registers bits are set, not allowing additional transfers to occur on a channel. | <p>It is possible that a trigger event was received when the parameter set associated with the channel/event was a NULL set for a previous transfer on the channel. This is typical in two cases:</p> <ol style="list-style-type: none"> <li>1) QDMA channels: Typically if the parameter set is non-static and expected to be terminated by a NULL set (i.e., OPT STATIC = 0, LINK = 0xFFFF), the parameter set is updated with a NULL set after submission of the last TR. Because QDMA channels are auto-triggered, this update caused the generation of an event. An event generated for a NULL set causes an error condition and results in setting the bits corresponding to the QDMA channel in the QEMR and QSER. This will disable further prioritization of the channel.</li> <li>2) DMA channels used in a continuous mode: The peripheral may be set up to continuously generate infinite events (for instance, in case of McASP, every time the data shifts out from the DXR register, it generates an XEV). The parameter set may be programmed to expect only a finite number of events and to be terminated by a NULL link. After the expected number of events, the parameter set is reloaded with a NULL parameter set. Because the peripheral will generate additional events, an error condition is set in the SER.Ex and EMR.Ex set, preventing further event prioritization. You must ensure that the number of events received is limited to the expected number of events for which the parameter set is programmed, or you must ensure that bits corresponding to particular channel or event are not set in the Secondary event registers (SER/SERH/QSER) and Event Missed Registers (EMR/EMRH/QEMR) before trying to perform subsequent transfers for the event/channel.</li> </ol> |

**Table 11-117. Debug List (continued)**

| Issue   | Description/Solution  |
|---|---|
| Completion interrupts are not asserted, or no further interrupts are received after the first completion interrupt. | <p>You must ensure the following:</p> <ol style="list-style-type: none"> <li>1) The interrupt generation is enabled in the OPT of the associated PaRAM set (TCINTEN = 1 and/or ITCINTEN = 1).</li> <li>2) The interrupts are enabled in the EDMA3 Channel Controller, via the Interrupt Enable Registers (IER/IERH).</li> <li>3) The corresponding interrupts are enabled in the device interrupt controller.</li> <li>4) The set interrupts are cleared in the interrupt pending registers (IPR/IPRH) before exiting the transfer completion interrupt service routine (ISR). See <a href="#">Section 11.3.9.1.2</a> for details on writing EDMA3 ISRs.</li> <li>5) If working with shadow region interrupts, make sure that the DMA Region Access registers (DRAE/DRAEH) are set up properly, because the DRAE/DRAEH registers act as secondary enables for shadow region completion interrupts, along with the IER/IERH registers.</li> </ol> <p>If working with shadow region interrupts, make sure that the bits corresponding to the transfer completion code (TCC) value are also enabled in the DRAE/DRAEH registers. For instance, if the PaRAM set associated with Channel 0 returns a completion code of 63 (OPT.TCC=63), ensure that DRAEH.E63 is also set for a shadow region completion interrupt because the interrupt pending register bit set will be IPRH.I63 (not IPR.I0).</p> |

### 11.5.2 Miscellaneous Programming/Debug Tips

1. For several registers, the setting and clearing of bits needs to be done via separate dedicated registers. For example, the Event Register (ER/ERH) can only be cleared by writing a 1 to the corresponding bits in the Event Clear Registers (ECR/ECRH). Similarly, the Event Enable Register (EER/EERH) bits can only be set with writes of 1 to the Event Enable Set Registers (EESR/EESRH) and cleared with writes of 1 to the corresponding bits in the Event Enable Clear Register (EECR/EECRH).
2. Writes to the shadow region memory maps are governed by region access registers (DRAE/DRAEH/QRAE). If the appropriate channels are not enabled in these registers, read/write access to the shadow region memory map is not enabled.
3. When working with shadow region completion interrupts, ensure that the DMA Region Access Registers (DRAE/DRAEH) for every region are set in a mutually exclusive way (unless it is a requirement for an application). If there is an overlap in the allocated channels and transfer completion codes (setting of Interrupt Pending Register bits) in the region resource allocation, it results in multiple shadow region completion interrupts. For example, if DRAE0.E0 and DRAE1.E0 are both set, then on completion of a transfer that returns a TCC=0, they will generate both shadow region 0 and 1 completion interrupts.
4. While programming a non-dummy parameter set, ensure the CCNT is not left to zero.
5. Enable the EDMA3CC error interrupt in the device controller and attach an interrupt service routine (ISR) to ensure that error conditions are not missed in an application and are appropriately addressed with the ISR.
6. Depending on the application, you may want to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue.
7. In applications where a large transfer is broken into sets of small transfers using chaining or other methods, you might choose to use the early chaining option to reduce the time between the sets of transfers and increase the throughput. However, keep in mind that with early completion, all data might have not been received at the end point when completion is reported because the EDMA3CC internally signals completion when the TR is submitted to the EDMA3TC, potentially before any data has been transferred.
8. The event queue entries can be observed to determine the last few events if there is a system failure (provided the entries were not bypassed).

### 11.5.3 Setting Up a Transfer

The following list provides a quick guide for the typical steps involved in setting up a transfer.

Step 1. Initiating a DMA/QDMA channel

- (a) Determine the type of channel (QDMA or DMA) to be used.
- (b) Channel mapping
  - (i) If using a QDMA channel, program the QCHMAP with the parameter set number to which the channel maps and the trigger word.
  - (ii) If using a DMA channel, program the DCHMAP with the parameter set number to which the channel maps.
- (c) If the channel is being used in the context of a shadow region, ensure the DRAE/DRAEH for the region is properly set up to allow read write accesses to bits in the event registers and interrupt registers in the Shadow region memory map. The subsequent steps in this process should be done using the respective shadow region registers. (Shadow region descriptions and usage are provided in [Section 11.3.7.1](#).)
- (d) Determine the type of triggering used.
  - (i) If external events are used for triggering (DMA channels), enable the respective event in EER/EERH by writing into EESR/EESRH.
  - (ii) If QDMA Channel is used, enable the channel in QEER by writing into QEESR.
- (e) Queue setup
  - (i) If a QDMA channel is used, set up the QDMAQNUM to map the channel to the respective event queue.
  - (ii) If a DMA channel is used, set up the DMAQNUM to map the event to the respective event queue.

Step 2. Parameter set setup

- (a) Program the PaRAM set number associated with the channel. Note that if it is a QDMA channel, the PaRAM entry that is configured as trigger word is written to last. Alternatively, enable the QDMA channel (step 1-b-ii above) just before the write to the trigger word.  
See [Section 11.3.19](#) for parameter set field setups for different types of transfers. See the sections on chaining ([Section 11.3.8](#)) and interrupt completion ([Section 11.3.9](#)) on how to set up final/intermediate completion chaining and/or interrupts.

Step 3. Interrupt setup

- (a) Enable the interrupt in the IER/IERH by writing into IESR/IESRH.
- (b) Ensure that the EDMA3CC completion interrupt (either the global or the shadow region interrupt) is enabled properly in the device interrupt controller.
- (c) Ensure the EDMA3CC completion interrupt (this refers to either the Global interrupt or the shadow region interrupt) is enabled properly in the Device Interrupt controller.
- (d) Set up the interrupt controller properly to receive the expected EDMA3 interrupt.

**Step 4. Initiate transfer**

- (a) This step is highly dependent on the event trigger source:
- (i) If the source is an external event coming from a peripheral, the peripheral will be enabled to start generating relevant EDMA3 events that can be latched to the ER transfer.
  - (ii) For QDMA events, writes to the trigger word (step 2-a above) will initiate the transfer.
  - (iii) Manually triggered transfers will be initiated by writes to the Event Set Registers (ESR/ESRH).
  - (iv) Chained-trigger events initiate when a previous transfer returns a transfer completion code equal to the chained channel number.

**Step 5. Wait for completion**

- (a) If the interrupts are enabled as mentioned in step 3 above, then the EDMA3CC will generate a completion interrupt to the CPU whenever transfer completion results in setting the corresponding bits in the interrupt pending register (IPR\IPRH). The set bits must be cleared in the IPR\IPRH by writing to corresponding bit in ICR\ICRH.
- (b) If polling for completion (interrupts not enabled in the device controller), then the application code can wait on the expected bits to be set in the IPR\IPRH. Again, the set bits in the IPR\IPRH must be manually cleared via ICR\ICRH before the next set of transfers is performed for the same transfer completion code values.

## **Touchscreen Controller**

This chapter describes the touchscreen controller of the device.

| Topic                                       | Page |
|---|------|
| 12.1 Introduction .....                     | 1016 |
| 12.2 Integration .....                      | 1017 |
| 12.3 Functional Description .....           | 1019 |
| 12.4 Operational Modes .....                | 1021 |
| 12.5 Touchscreen Controller Registers ..... | 1025 |

## 12.1 Introduction

The touchscreen controller and analog-to-digital converter subsystem (TSC\_ADC\_SS) is an 8-channel general-purpose analog-to-digital converter (ADC) with optional support for interleaving touchscreen (TS) conversions for a 4-wire, 5-wire, or 8-wire resistive panel. The TSC\_ADC\_SS can be configured for use in one of the following applications:

- 8 general-purpose ADC channels
- 4-wire TSC with 4 general-purpose ADC channels
- 5-wire TSC with 3 general-purpose ADC channels
- 8-wire TSC

### 12.1.1 TSC\_ADC Features

The main features of the TSC\_ADC\_SS include:

- Support for 4-wire, 5-wire, and 8-wire resistive TS panels
- Support for interleaving TS capture and general-purpose ADC modes
- Programmable FSM sequencer that supports 16 steps:
  - Software register bit for start of conversion
  - Optional start of conversion HW synchronized to Pen touch or external HW event (but not both)
  - Single conversion (one-shot)
  - Continuous conversions
  - Sequence through all input channels based on a mask
  - Programmable OpenDelay before sampling each channel
  - Programmable sampling delay for each channel
  - Programmable averaging of input samples - 16/8/4/2/1
  - Differential or singled ended mode setting for each channel
  - Store data in either of two FIFO groups
  - Option to encode channel number with data
  - Support for servicing FIFOs via DMA or CPU
  - Programmable DMA Request event (for each FIFO)
  - Dynamically enable or disable channel inputs during operation
  - Stop bit to end conversion
- Support for the following interrupts and status, with masking:
  - Interrupt for HW pen (touch) event
  - Interrupt for HW Pen Up event
  - Interrupt after a sequence of conversions (all non-masked channels)
  - Interrupt for FIFO threshold levels
  - Interrupt if sampled data is out of a programmable range
  - Interrupt for FIFO overflow and underflow conditions
  - Status bit to indicate if ADC is busy converting

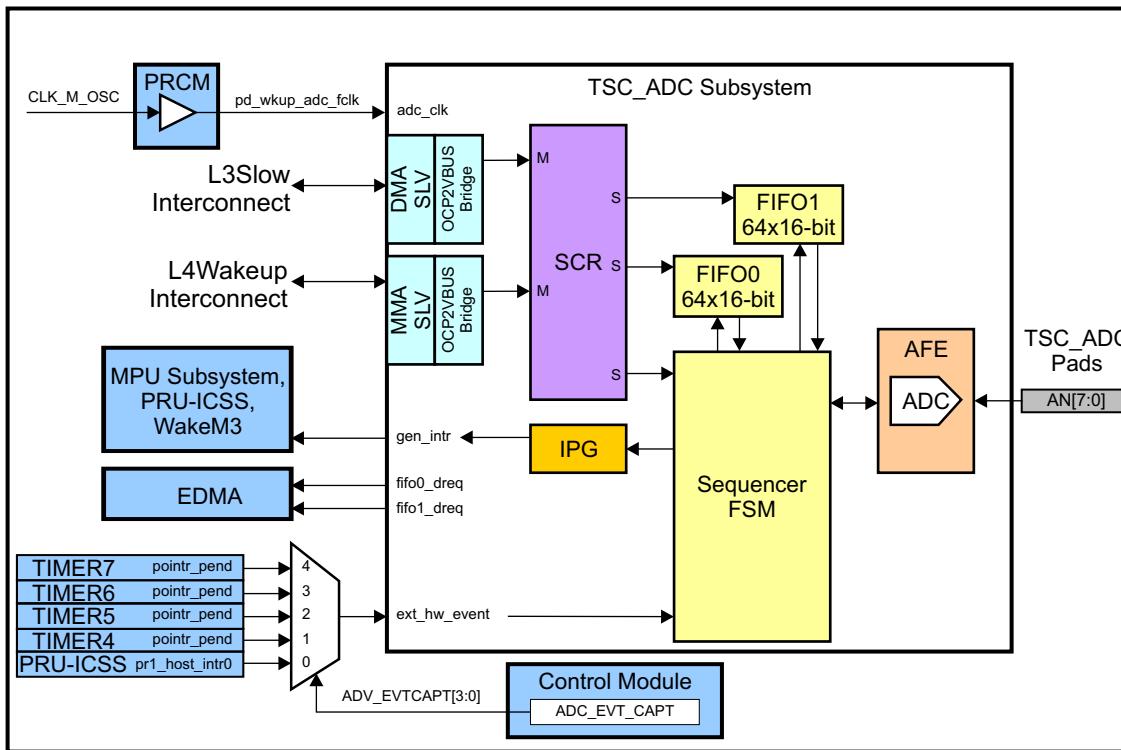
### 12.1.2 Unsupported TSC\_ADC\_SS Features

This device supports all TSC\_ADC\_SS features.

## 12.2 Integration

Figure 12-1 shows the integration of the TSC\_ADC module in the device.

**Figure 12-1. TSC\_ADC Integration**



pr1\_host\_intr[0:7] corresponds to Host-2 to Host-9 of the PRU-ICSS interrupt controller.

### 12.2.1 TSC\_ADC Connectivity Attributes

The general connectivity attributes for the TSC\_ADC module are summarized in [Table 12-1](#).

**Table 12-1. TSC\_ADC Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power domain        | Wakeup Domain   |
| Clock domain        | PD_WKUP_L4_WKUP_GCLK (OCP)<br>PD_WKUP_ADC_FCLK (Func)                               |
| Reset signals       | WKUP_DOM_RST_N  |
| Idle/Wakeup signals | Smart idle<br>Wakeup  |
| Interrupt request   | 1 interrupt to MPU Subsystem (ADC_TSC_GENINT), PRU-ICSS (gen_intr_pend), and WakeM3 |
| DMA request         | 2 Events (tsc_adc_FIFO0, tsc_adc_FIFO1)   |
| Physical address    | L3 Slow slave port (DMA)<br>L4 Wkup slave port (MMR)                                |

### 12.2.2 TSC\_ADC Clock and Reset Management

The TSC\_ADC has two clock domains. The ADC uses the adc\_clk. The bus interfaces, FIFOs, sequencer, and all other logic use the ocp\_clk.

**Table 12-2. TSC\_ADC Clock Signals**

| Clock Signal                      | Max Freq        | Reference / Source | Comments                          |
|-----------------------------------|-----------------|--------------------|-----------------------------------|
| ocp_clk<br>OCP / Functional clock | 100 MHz         | CORE_CLKOUTM4 / 2  | pd_pkup_l4_pkup_gclk<br>From PRCM |
| adc_clk<br>ADC clock              | 24 MHz<br>(typ) | CLK_M_OSC          | pd_pkup_adc_fclk<br>From PRCM     |

### 12.2.3 TSC\_ADC Pin List

The TSC\_ADC external interface signals are shown in [Table 12-3](#).

**Table 12-3. TSC\_ADC Pin List**

| Pin     | Type  | Description                              |
|---------|-------|--|
| AN[7:0] | I     | Analog Input                             |
| VREFN   | Power | Analog Reference Input Negative Terminal |
| VREFP   | Power | Analog Reference Input Positive Terminal |

## 12.3 Functional Description

Before enabling the TSC\_ADC\_SS module, the user must first program the Step Configuration registers in order to configure a channel input to be sampled. There are 16 programmable Step Configuration registers which are used by the sequencer to control which switches to turn on or off (inputs to the AFE), which channel to sample, , and which mode to use (HW triggered or SW enabled, one-shot or continuous, averaging, where to save the FIFO data, etc).

### 12.3.1 HW Synchronized or SW Channels

The user can control the start behavior of each channel input by deciding if a channel should be sampled immediately (SW enabled) after it is enabled, or if the channel should wait for a HW event to occur first ( a HW event must either be mapped to the touch screen PEN event or mapped to theHW event input signal, but not both) . Each channel input can be configured independently via the Step Configuration register.

### 12.3.2 Open Delay and Sample Delay

The user can program the delay between driving the inputs to the AFE and the time to send the start of conversion signal. This delay can be used to allow the voltages to stabilize on the touch screen panel before sampling. This delay is called “open delay” and can also be programmed to zero. The user also has control of the sampling time (width of the start of conversion signal) to the AFE which is called the “sample delay”. Each channel input is configured independently via the Step Delay register.

### 12.3.3 Averaging of Samples (1, 2, 4, 8, and 16)

Each channel input has the capability to average the sampled data. The valid averaging options are 1 (no average), 2, 4, 8, and 16. If averaging is turned on, then the channel is immediately sampled again (up to 16 times) and final averaged sample data is stored in the FIFO. Each channel input can be independently configured via the Step Configuration registers.

### 12.3.4 One-Shot (Single) or Continuous Mode

When the sequencer finishes cycling through all the enabled steps , the user can decide if the sequencer should stop (one-shot), or loop back and schedule a the step again (continuous).

If one-shot mode is selected, the sequencer will take care of disabling the step enable bit after the conversion. If continuous mode is selected, it is the software's responsibility to turn off the step enable bit.

### 12.3.5 Interrupts

The following interrupts are supported via enable bits and maskable.

The HW Pen (Touch) interrupt is generated when the user presses the touch screen. This can only occur if the AFE is configured properly (i.e., one of the Pen Ctrl bits must be enabled, and also the correct setting for a path to ground in the StepConfig Registers). Although the HW Pen interrupt can be disabled by the SW, the event will still trigger the sequencer to start if the step is configured as a HW synchronized event. The HW Pen interrupt is an synchronous event and can be used even if the TSC\_ADC\_SS clocks are disabled. The HW Pen interrupt can be used to as a wakeup source.

An END\_OF\_SEQUENCE interrupt is generated after the sequencer finishes servicing the last enabled step.

A PEN\_UP interrupt is generated only if the HW steps are used and the Charge step must be enabled. If a pen\_down event caused the HW steps to be scheduled, and after the sequencer finishes servicing the charge step, if the HW Pen (touch) is not active then PEN UP is generated.

Each FIFO has support for generating interrupts when the FIFO word count has reached a programmable threshold level. The user can program these registers to the desired word count at which the CPU should be interrupted. Whenever the threshold counter value is reached, it sets the FIFO THRESHOLD interrupt flag, and the CPU is interrupted if the FIFO THRESHOLD interrupt enable bit is set. The CPU clears the interrupt flag, after emptying the FIFO, by writing a '1' to the FIFO TRESHOLD interrupt status bit. To determine how many samples are currently in the FIFO at a given moment, the FIFO\_WORD\_COUNT register can be read by the CPU.

The FIFO can also generate FIFO\_OVERRUN and FIFO\_UNDERFLOW interrupts. The user can mask these events by programming the enable bits. To clear a FIFO underflow or FIFO overrun interrupt, the user should write a '1' so the status bit. The TSC\_ADC\_SS does not recover from these conditions automatically. Therefore, the software will need to disable and then again enable the TSC\_ADC\_SS. Before user can turn the module back on, he must first check if the ADC FSM is in IDLE state by reading from the ADC\_STATUS\_REG register.

### 12.3.6 DMA Requests

Each FIFO group can be serviced by either a DMA or by the CPU. To generate DMA requests, the user must first enable the TSC\_ADC\_SS\_DMAENABLE\_SET Register. Also, the user can program the desired number of words to generate a DMA request using the TSC\_ADC\_SS\_DMAREQ register. When the FIFO level reaches or exceeds that value, a DMA request is generated.

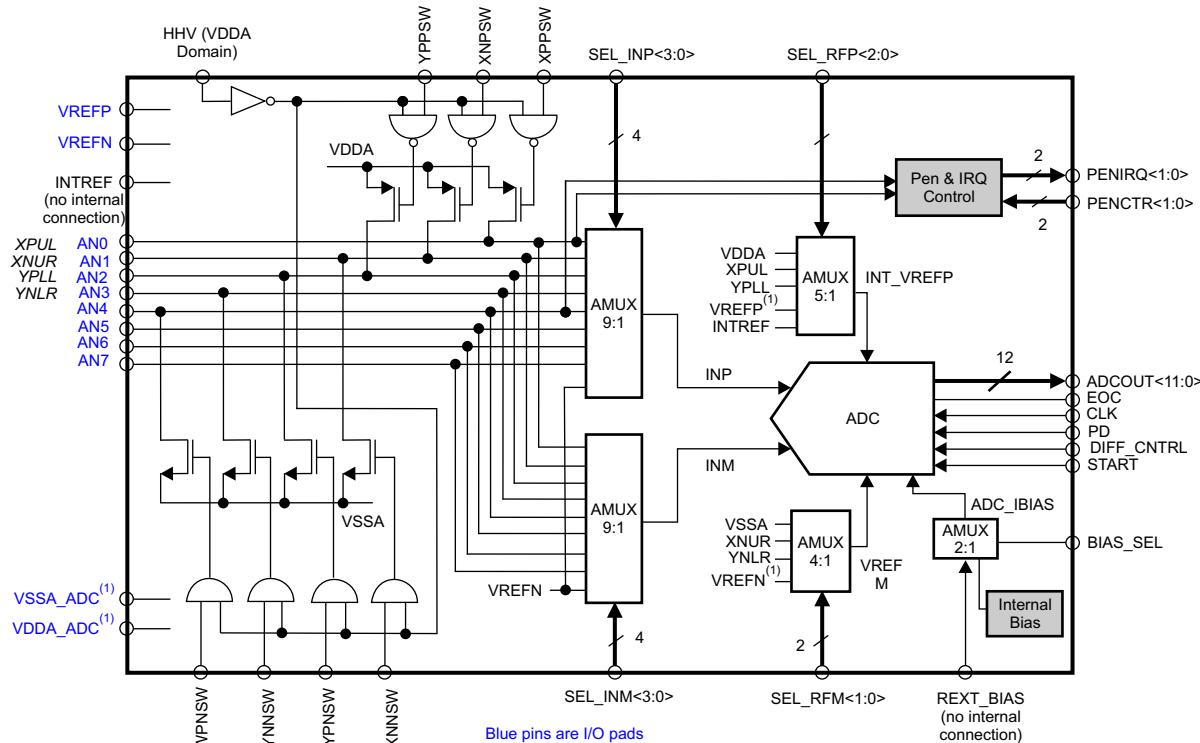
The DMA slave port allows for burst reads in order to effectively move the FIFO data. Internally, the OCP DMA address (MSB) is decoded for either FIFO 0 or FIFO 1. The lower bits of the DMA addresses are ignored since the FIFO pointers are incremented internally. Refer to **section 5 (DMA)** for more information.

### 12.3.7 Analog Front End (AFE) Functional Block Diagram

The AFE features are listed below, and some are controlled by the TSC\_ADC\_SS:

- 12-bit ADC
- Sampling rate can be as fast as every 15 ADC clock cycles
- Support for internal ADC clock divider logic
- Support for configuring the delay between samples also the sampling time

**Figure 12-2. Functional Block Diagram**



(1) In AM335x ARM Cortex-A8 Microprocessors (MPUs) (literature number [SPRS717](#)):

- VDDA\_ADC and VSSA\_ADC are referred to as "Internal References"
- VREFP and VREFN are referred to as "External References"

## 12.4 Operational Modes

The sequencer is completely controlled by software and behaves accordingly to how the **Step Registers** are programmed. A **step** is the general term for sampling a channel input. It is defined by the programmer who decides which input values to send to the AFE (via the StepConfig register), and also how (via StepConfig register) and when (via StepDelay register) to sample a channel input.

A step consists of these three registers:

- StepEnable: Enables or disables the step
- StepConfig: Controls the input values to the ADC (the reference voltages, the pull up/down transistor biasing, which input channel to sample, differential control, HW synchronized or SW enabled, averaging, and which FIFO group to save the data)
- StepDelay: Controls the OpenDelay (the time between driving the AFE inputs until sending the SOC signal to the AFE), and also controls the SampleDelay (the time for the ADC to sample the input signal)

The sequencer supports a total of 16 programmable steps, a touchscreen Charge step, and an Idle step. Each step consists of the three registers above (StepEnable, StepConfig, and StepDelay), except the Idle step does not have a StepEnable bit (it must always be on), and it also does not have a StepDelay register. Also, during the Idle and touchscreen Charge steps, the ADC does not actually sample a channel.

Assuming all the steps are configured as general purpose mode (no touchscreen), then the steps would be configured as SW enabled. When the TSC\_ADC\_SS is first enabled, the sequencer will always start in the Idle step and then wait for a StepEnable[n] bit to turn on. After a step is enabled, the sequencer will start with the lowest step (1) and continue until step (16). If a step is not enabled, then sequencer will skip to the next step. If all steps are disabled, then the sequencer will remain in the IDLE state and continue to apply the Idle StepConfig settings.

Assuming a touchscreen-only mode (no general-purpose channels) the steps could be configured as HW synchronized triggered (mapped to the PEN event).. The sequencer would wait in the IDLE state until a HW pen down event occurred and then begin the HW step conversions. The touchscreen Charge step occurs after the last HW step before going back to the Idle state (the TS Charge step is needed to charge touch screen capacitance which allows the controller to detect subsequent Pen touch events).

Assuming a mixed mode application (touchscreen and general purpose channels), the user can configure the steps as either HW triggered (mapped to Pen event) or SW enabled. If the sequencer is in the IDLE state and a HW pen event occurs, then the HW steps (from lowest to highest) are always scheduled first, followed by the TS Charge step. If there is no HW event, then the SW enabled steps are scheduled instead

If a HW event occurs while the sequencer is in the middle of scheduling the SW steps, the user can program the scheduler to allow preemption. If the HW preempt control bit is enabled, the sequencer will allow the current SW step to finish and then schedule the HW steps. After the last HW step and TS Charge step are completed, the sequencer will continue from the next SW step (before the preemption occurred). If the HW preemption is disabled, then the touch event will be ignored until the last software step is completed; if the touch event is removed before the last software step is finished, then the touch event will be missed.

Even if a touch screen is not present, the user can still configure the steps to be HW synchronize by mapping to the HW event input signal. This HW event input signal could be driven at the SOC level from a timer interrupt or some other IP and can be used to start a channel conversion.

When mapping is set for the input HW event signal, then the TSC\_ADC\_SS will wait for a rising edge transition (from low to high) before starting. The input HW event signal is captured on the internal L4 OCP clock domain. The HW event input signal should be held for at least 2 TSC\_ADC\_SS OCP clocks (L4 frequency).

An END\_OF\_SEQUENCE interrupt and the PEN\_UP interrupt are generated after the last active step is completed before going back to the IDLE state. The END\_OF\_SEQUENCE interrupt does not mean data is in the FIFO (should use the FIFO interrupts and word count reg)

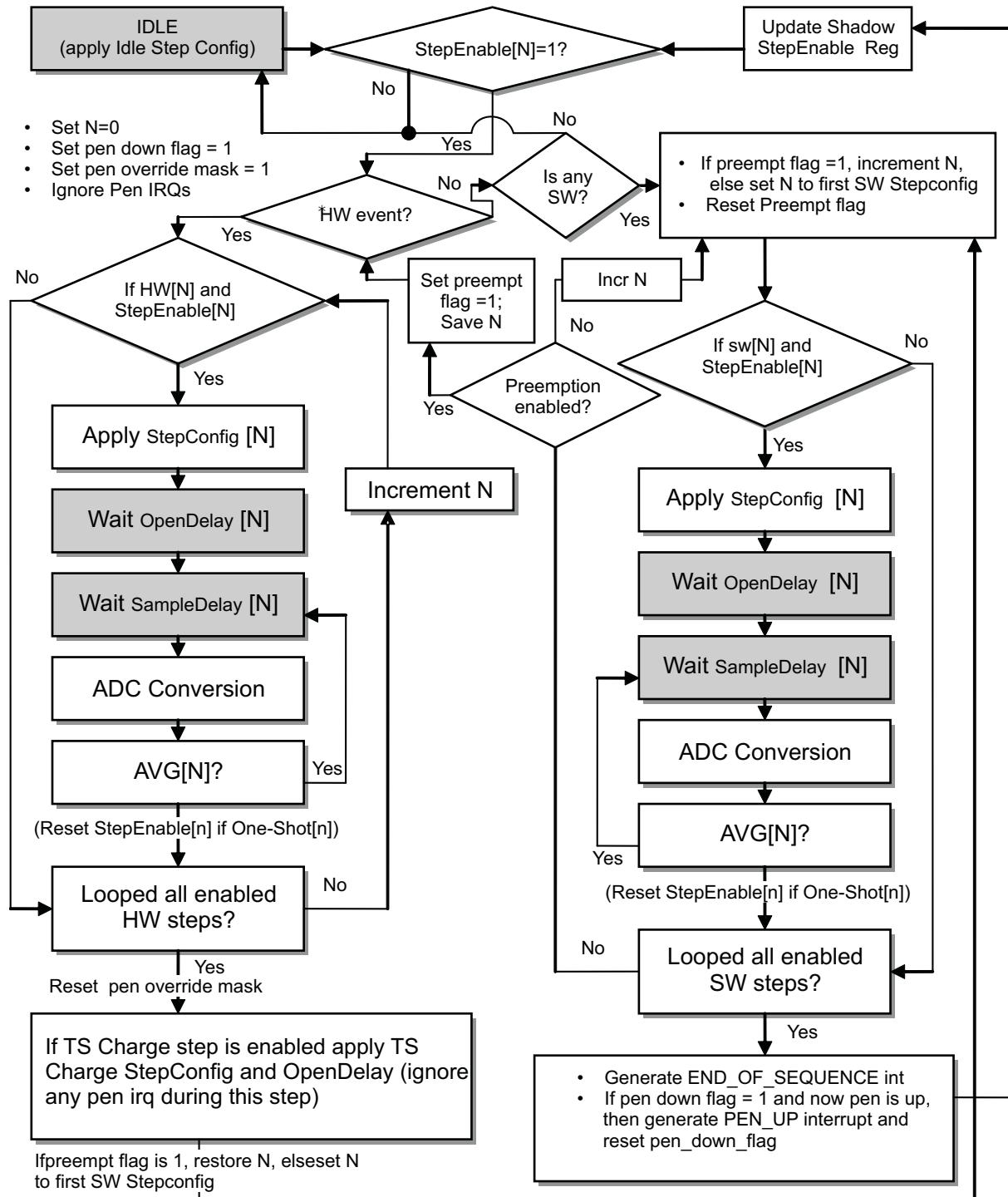
### 12.4.1 PenCtrl and PenIRQ

The Pen IRQ can only occur if the correct Pen Ctrl bits are high (AFE Pen Ctrl bits 6:5 in the TSC\_ADC control register) and if the correct ground transistor biasing is set in the StepConfig [N] Register. Refer to the application notes for the correct settings.

If a step is configured as HW synchronized, the sequencer will override the PenCtrl bits set by the software (bits 6 and 5) once it transitions from the IdleStep. The sequencer will automatically mask the PenCtrl bits (override them and turn them off) so that the ADC can get an accurate measurement from the x and y channels. After the last HW synchronized step, the sequence will go to the TS Charge step and the pen\_overide mask is removed and the values set by the SW (bits 6:5) will have control. The pen\_irq will be temporarily ignored during the TS Charge step (HW will mask any potential glitches that may occur)

If the sequencer is not using the HW synchronized approach, (all the steps are configured as software enabled), then it is the software programmer's responsibility to correctly turn on and off the pen ctrl bits to the AFE in order get the correct measures from the touch screen. It is the responsibility of the software to enable the charge step and ignore any potential glitches by using pen\_irq register.

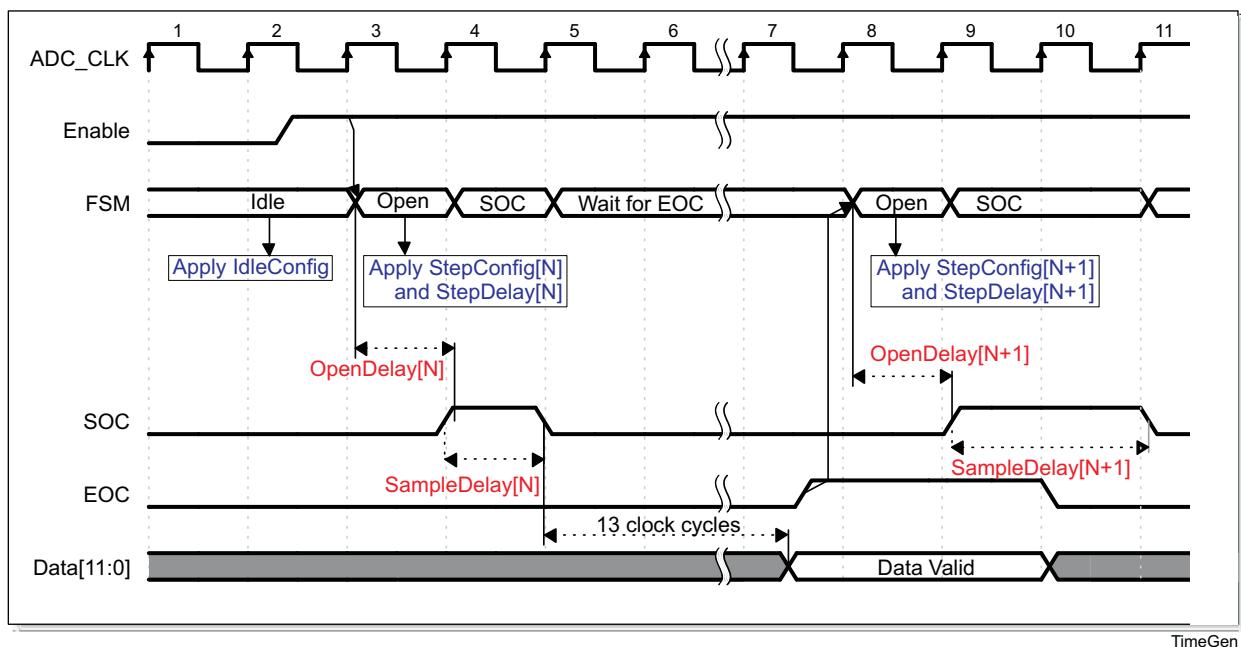
It is also possible to detect the HW Pen event even if all the StepEnable[n] bits are off. By setting the PenCtrl bit to 1, and configuring the IdleStep Configuration register to correctly bias the transistor to grnd, the HW\_PEN event will be generated. The flowchart for the sequencer is shown in [Figure 12-3](#) and an example timing diagram in [Figure 12-4](#).

**Figure 12-3. Sequencer FSM**


\* HW event can either be Pen touch, or input HW event, but not both

The previous diagram does not actually represent clock cycles but instead illustrates how the scheduler will work. However, each shaded box above does represent a FSM state and will use a clock cycle. Using the minimum delay values, the ADC can sample at 15 ADC clocks per sample. Below is an example timing diagram illustrating the states of the sequencer and also the showing when the StepConfig and the StepDelay registers values are applied. The below example assumes the steps are software controlled, and averaging is turned off.

Figure 12-4. Example Timing Diagram for Sequencer



The Idle StepConfig is always enabled and applied when the FSM is in the IDLE state (i.e., either waiting for a HW event or waiting for a step to be enabled). The Idle StepConfig can not be disabled.

Once the TSC\_ADC\_SS is enabled and assuming at least one StepEnable[N] is active, the FSM will transition from the Idle state and apply the first active StepConfig[N] and StepDelay[N] register settings. It is possible for the OpenDelay[N] value to be 0, and therefore the FSM will immediately skip to the SampleDelay[N] state (which is minimum 1 clock cycle). The ADC will begin the sampling on the falling edge of the SOC signal. After the ADC is finished converting the channel data (13 cycles later), the EOC signal is sent and the FSM will then apply the next active step [N+1].

This process is repeated and continued (from step 1 to step 16) until the last active step is completed.

## 12.5 Touchscreen Controller Registers

### 12.5.1 TSC\_ADC\_SS Registers

**Table 12-4** lists the memory-mapped registers for the TSC\_ADC\_SS. All register offset addresses not listed in **Table 12-4** should be considered as reserved locations and the register contents should not be modified.

**Table 12-4. TSC\_ADC\_SS REGISTERS**

| Offset | Acronym              | Register Name | Section                           |
|--------|----------------------|---------------|-----------------------------------|
| 0h     | REVISION             |               | <a href="#">Section 12.5.1.1</a>  |
| 10h    | SYSCONFIG            |               | <a href="#">Section 12.5.1.2</a>  |
| 24h    | IRQSTATUS_RAW        |               | <a href="#">Section 12.5.1.3</a>  |
| 28h    | IRQSTATUS            |               | <a href="#">Section 12.5.1.4</a>  |
| 2Ch    | IRQENABLE_SET        |               | <a href="#">Section 12.5.1.5</a>  |
| 30h    | IRQENABLE_CLR        |               | <a href="#">Section 12.5.1.6</a>  |
| 34h    | IRQWAKEUP            |               | <a href="#">Section 12.5.1.7</a>  |
| 38h    | DMAENABLE_SET        |               | <a href="#">Section 12.5.1.8</a>  |
| 3Ch    | DMAENABLE_CLR        |               | <a href="#">Section 12.5.1.9</a>  |
| 40h    | CTRL                 |               | <a href="#">Section 12.5.1.10</a> |
| 44h    | ADCSTAT              |               | <a href="#">Section 12.5.1.11</a> |
| 48h    | ADCRANGE             |               | <a href="#">Section 12.5.1.12</a> |
| 4Ch    | ADC_CLKDIV           |               | <a href="#">Section 12.5.1.13</a> |
| 50h    | ADC_MISC             |               | <a href="#">Section 12.5.1.14</a> |
| 54h    | STEPENABLE           |               | <a href="#">Section 12.5.1.15</a> |
| 58h    | IDLECONFIG           |               | <a href="#">Section 12.5.1.16</a> |
| 5Ch    | TS_CHARGE_STEPCONFIG |               | <a href="#">Section 12.5.1.17</a> |
| 60h    | TS_CHARGE_DELAY      |               | <a href="#">Section 12.5.1.18</a> |
| 64h    | STEPCONFIG1          |               | <a href="#">Section 12.5.1.19</a> |
| 68h    | STEPDELAY1           |               | <a href="#">Section 12.5.1.20</a> |
| 6Ch    | STEPCONFIG2          |               | <a href="#">Section 12.5.1.21</a> |
| 70h    | STEPDELAY2           |               | <a href="#">Section 12.5.1.22</a> |
| 74h    | STEPCONFIG3          |               | <a href="#">Section 12.5.1.23</a> |
| 78h    | STEPDELAY3           |               | <a href="#">Section 12.5.1.24</a> |
| 7Ch    | STEPCONFIG4          |               | <a href="#">Section 12.5.1.25</a> |
| 80h    | STEPDELAY4           |               | <a href="#">Section 12.5.1.26</a> |
| 84h    | STEPCONFIG5          |               | <a href="#">Section 12.5.1.27</a> |
| 88h    | STEPDELAY5           |               | <a href="#">Section 12.5.1.28</a> |
| 8Ch    | STEPCONFIG6          |               | <a href="#">Section 12.5.1.29</a> |
| 90h    | STEPDELAY6           |               | <a href="#">Section 12.5.1.30</a> |
| 94h    | STEPCONFIG7          |               | <a href="#">Section 12.5.1.31</a> |
| 98h    | STEPDELAY7           |               | <a href="#">Section 12.5.1.32</a> |
| 9Ch    | STEPCONFIG8          |               | <a href="#">Section 12.5.1.33</a> |
| A0h    | STEPDELAY8           |               | <a href="#">Section 12.5.1.34</a> |
| A4h    | STEPCONFIG9          |               | <a href="#">Section 12.5.1.35</a> |
| A8h    | STEPDELAY9           |               | <a href="#">Section 12.5.1.36</a> |
| ACh    | STEPCONFIG10         |               | <a href="#">Section 12.5.1.37</a> |
| B0h    | STEPDELAY10          |               | <a href="#">Section 12.5.1.38</a> |
| B4h    | STEPCONFIG11         |               | <a href="#">Section 12.5.1.39</a> |
| B8h    | STEPDELAY11          |               | <a href="#">Section 12.5.1.40</a> |
| BCh    | STEPCONFIG12         |               | <a href="#">Section 12.5.1.41</a> |

**Table 12-4. TSC\_ADC\_SS REGISTERS (continued)**

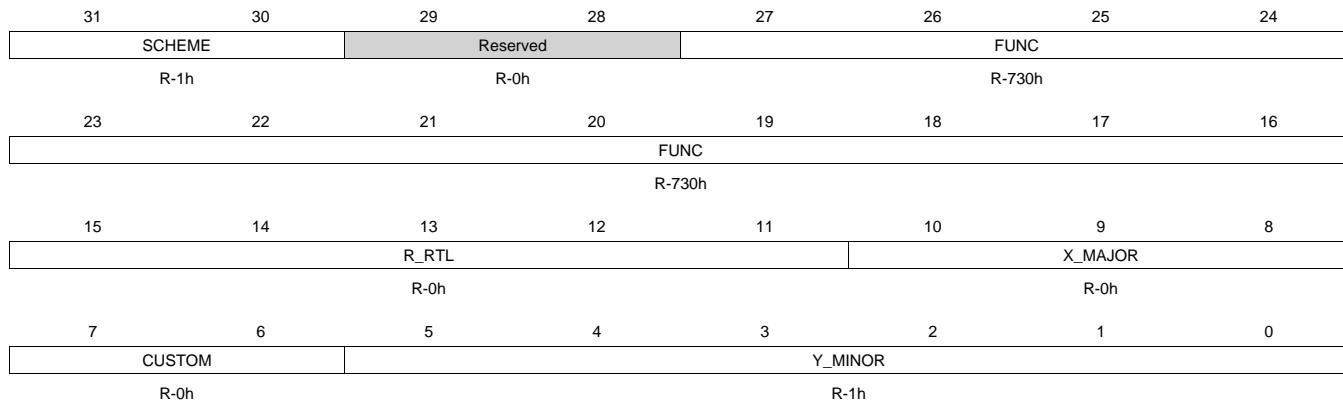
| Offset          | Acronym        | Register Name | Section           |
|-----------------|----------------|---------------|-------------------|
| C0h             | STEPDELAY12    |               | Section 12.5.1.42 |
| C4h             | STEPCONFIG13   |               | Section 12.5.1.43 |
| C8h             | STEPDELAY13    |               | Section 12.5.1.44 |
| CCh             | STEPCONFIG14   |               | Section 12.5.1.45 |
| D0h             | STEPDELAY14    |               | Section 12.5.1.46 |
| D4h             | STEPCONFIG15   |               | Section 12.5.1.47 |
| D8h             | STEPDELAY15    |               | Section 12.5.1.48 |
| DCh             | STEPCONFIG16   |               | Section 12.5.1.49 |
| E0h             | STEPDELAY16    |               | Section 12.5.1.50 |
| E4h             | FIFO0COUNT     |               | Section 12.5.1.51 |
| E8h             | FIFO0THRESHOLD |               | Section 12.5.1.52 |
| EC <sub>h</sub> | DMA0REQ        |               | Section 12.5.1.53 |
| F0h             | FIFO1COUNT     |               | Section 12.5.1.54 |
| F4h             | FIFO1THRESHOLD |               | Section 12.5.1.55 |
| F8h             | DMA1REQ        |               | Section 12.5.1.56 |
| 100h            | FIFO0DATA      |               | Section 12.5.1.57 |
| 200h            | FIFO1DATA      |               | Section 12.5.1.58 |

### 12.5.1.1 REVISION Register (offset = 0h) [reset = 47300001h]

REVISION is shown in [Figure 12-5](#) and described in [Table 12-5](#).

#### Revision Register

**Figure 12-5. REVISION Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-5. REVISION Register Field Descriptions**

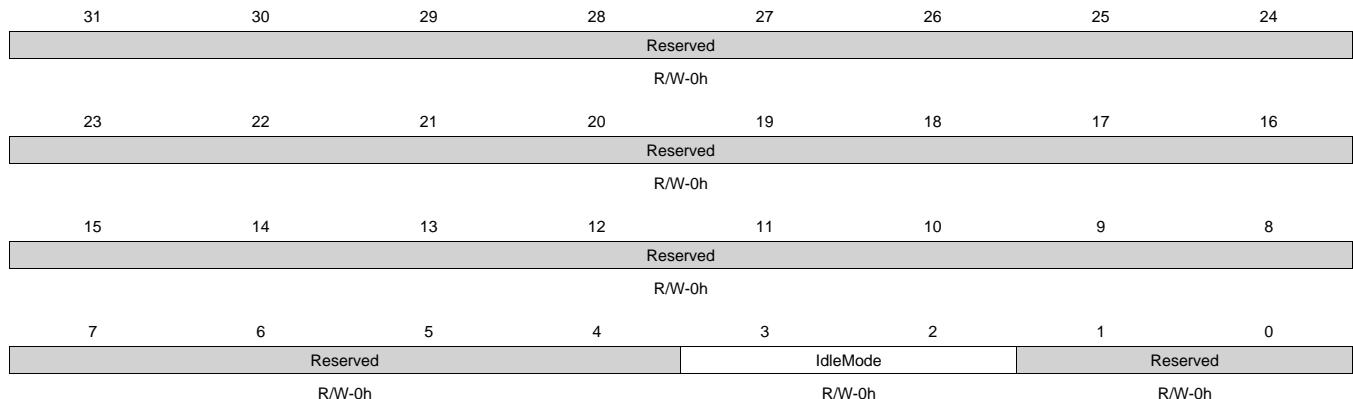
| Bit   | Field    | Type | Reset | Description                                      |
|-------|----------|------|-------|--|
| 31-30 | SCHEME   | R    | 1h    | HL 0.8 scheme                                    |
| 29-28 | Reserved | R    | 0h    | Always read as 0.<br>Writes have no affect.      |
| 27-16 | FUNC     | R    | 730h  | Functional Number                                |
| 15-11 | R RTL    | R    | 0h    | RTL revision.<br>Will vary depending on release. |
| 10-8  | X_MAJOR  | R    | 0h    | Major revision.                                  |
| 7-6   | CUSTOM   | R    | 0h    | Custom revision.                                 |
| 5-0   | Y_MINOR  | R    | 1h    | Minor revision                                   |

### 12.5.1.2 SYSCONFIG Register (offset = 10h) [reset = 0h]

SYSCONFIG is shown in [Figure 12-6](#) and described in [Table 12-6](#).

#### SysConfig Register

**Figure 12-6. SYSCONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-6. SYSCONFIG Register Field Descriptions**

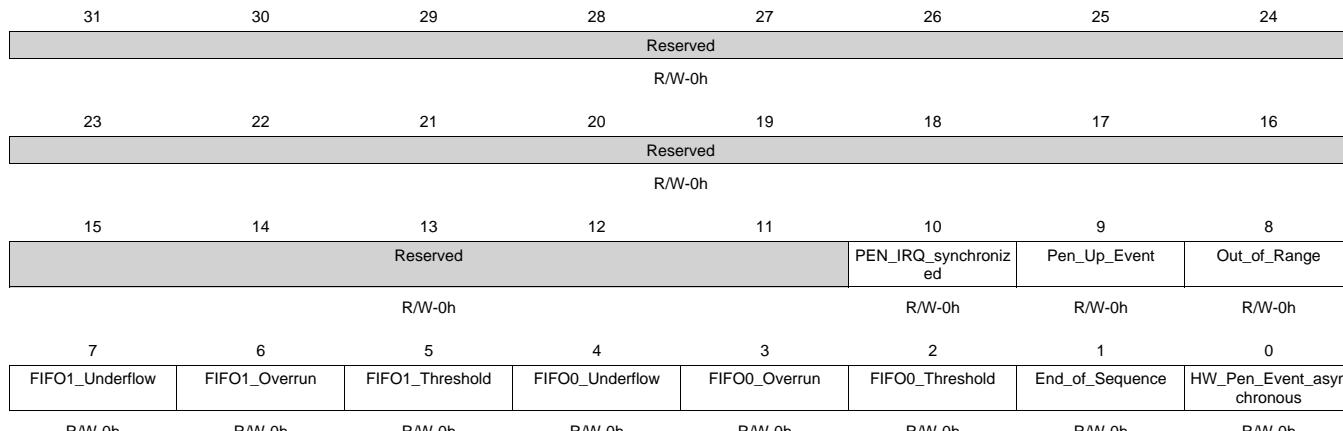
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-4 | Reserved | R/W  | 0h    |  |
| 3-2  | IdleMode | R/W  | 0h    | 00 = Force Idle (always acknowledges).<br>01 = No Idle Mode (never acknowledges).<br>10 = Smart-Idle Mode.<br>11 = Smart Idle with Wakeup. |
| 1-0  | Reserved | R/W  | 0h    |  |

### 12.5.1.3 IRQSTATUS\_RAW Register (offset = 24h) [reset = 0h]

IRQSTATUS\_RAW is shown in [Figure 12-7](#) and described in [Table 12-7](#).

IRQ status (unmasked)

**Figure 12-7. IRQSTATUS\_RAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-7. IRQSTATUS\_RAW Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description   |
|-------|----------------------|------|-------|---|
| 31-11 | Reserved             | R/W  | 0h    |   |
| 10    | PEN_IRQ_synchronized | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 9     | Pen_Up_Event         | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 8     | Out_of_Range         | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 7     | FIFO1_Underflow      | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 6     | FIFO1_OVERRUN        | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 5     | FIFO1_Threshold      | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 4     | FIFO0_Underflow      | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 3     | FIFO0_OVERRUN        | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |

**Table 12-7. IRQSTATUS\_RAW Register Field Descriptions (continued)**

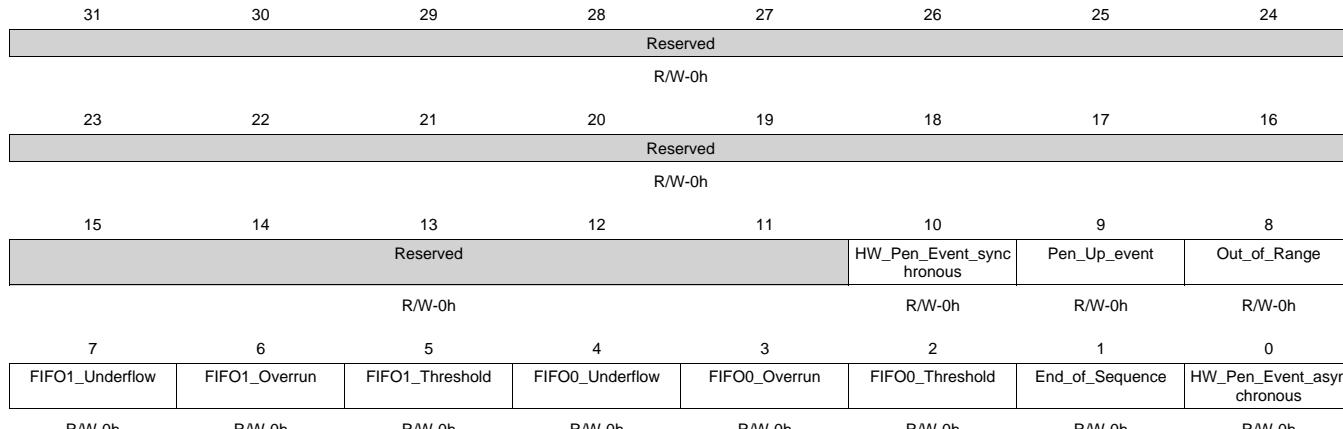
| Bit | Field                     | Type | Reset | Description   |
|-----|---------------------------|------|-------|---|
| 2   | FIFO0_Threshold           | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 1   | End_of_Sequence           | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |
| 0   | HW_Pen_Event_asynchronous | R/W  | 0h    | Write 0 = No action.<br>Write 1 = Set event (debug).<br>Read 0 = No event pending.<br>Read 1 = Event pending. |

#### 12.5.1.4 IRQSTATUS Register (offset = 28h) [reset = 0h]

IRQSTATUS is shown in [Figure 12-8](#) and described in [Table 12-8](#).

IRQ status (masked)

**Figure 12-8. IRQSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-8. IRQSTATUS Register Field Descriptions**

| Bit   | Field                        | Type | Reset | Description   |
|-------|------------------------------|------|-------|---|
| 31-11 | Reserved                     | R/W  | 0h    |   |
| 10    | HW_Pen_Event_sync<br>hronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 9     | Pen_Up_event                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 8     | Out_of_Range                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 7     | FIFO1_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 6     | FIFO1_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 5     | FIFO1_Threshold              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 4     | FIFO0_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 3     | FIFO0_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |

**Table 12-8. IRQSTATUS Register Field Descriptions (continued)**

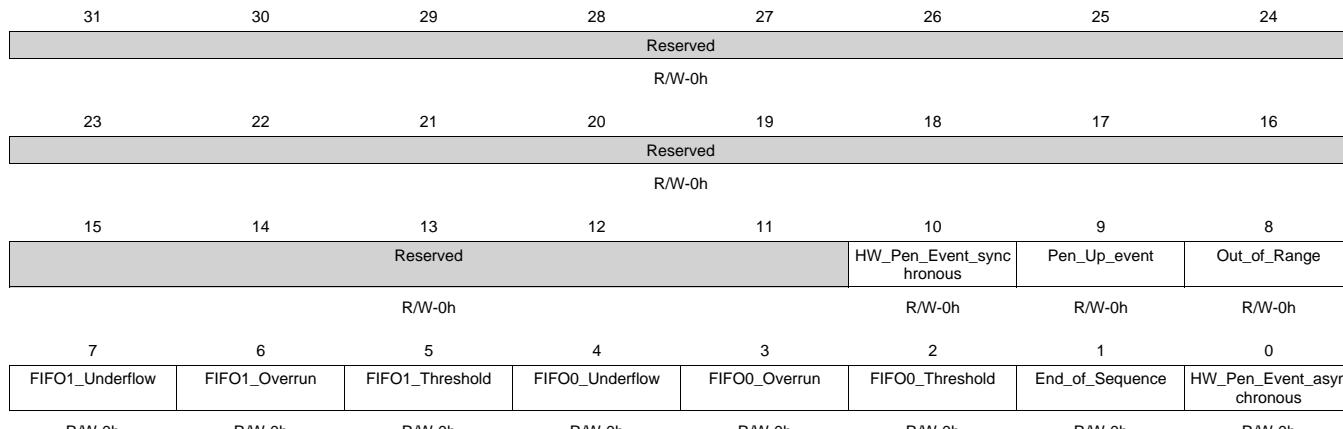
| Bit | Field                     | Type | Reset | Description   |
|-----|---------------------------|------|-------|---|
| 2   | FIFO0_Threshold           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 1   | End_of_Sequence           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |
| 0   | HW_Pen_Event_asynchronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = No (enabled) event pending.<br>Read 1 = Event pending.<br>Write 1 = Clear (raw) event. |

### 12.5.1.5 IRQENABLE\_SET Register (offset = 2Ch) [reset = 0h]

IRQENABLE\_SET is shown in [Figure 12-9](#) and described in [Table 12-9](#).

IRQ enable set bits

**Figure 12-9. IRQENABLE\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-9. IRQENABLE\_SET Register Field Descriptions**

| Bit   | Field                        | Type | Reset | Description   |
|-------|------------------------------|------|-------|---|
| 31-11 | Reserved                     | R/W  | 0h    |   |
| 10    | HW_Pen_Event_sync<br>hronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 9     | Pen_Up_event                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 8     | Out_of_Range                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 7     | FIFO1_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 6     | FIFO1_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 5     | FIFO1_Threshold              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 4     | FIFO0_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 3     | FIFO0_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |

**Table 12-9. IRQENABLE\_SET Register Field Descriptions (continued)**

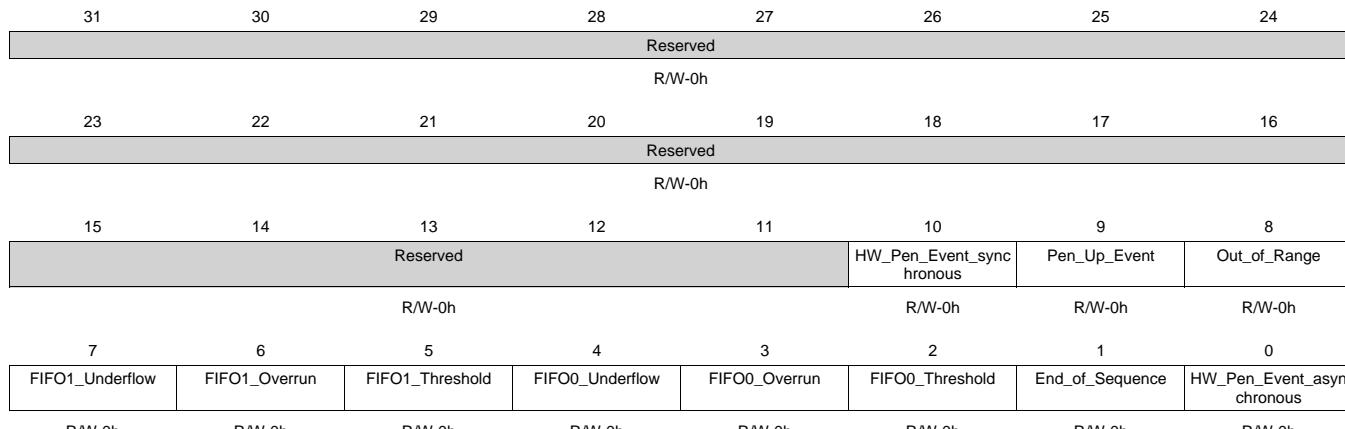
| Bit | Field                     | Type | Reset | Description   |
|-----|---------------------------|------|-------|---|
| 2   | FIFO0_Threshold           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 1   | End_of_Sequence           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |
| 0   | HW_Pen_Event_asynchronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Enable interrupt. |

### 12.5.1.6 IRQENABLE\_CLR Register (offset = 30h) [reset = 0h]

IRQENABLE\_CLR is shown in [Figure 12-10](#) and described in [Table 12-10](#).

IRQ enable clear bits

**Figure 12-10. IRQENABLE\_CLR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-10. IRQENABLE\_CLR Register Field Descriptions**

| Bit   | Field                        | Type | Reset | Description  |
|-------|------------------------------|------|-------|--|
| 31-11 | Reserved                     | R/W  | 0h    |  |
| 10    | HW_Pen_Event_sync<br>hronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 9     | Pen_Up_Event                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 8     | Out_of_Range                 | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 7     | FIFO1_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 6     | FIFO1_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 5     | FIFO1_Threshold              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 4     | FIFO0_Underflow              | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 3     | FIFO0_OVERRUN                | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |

**Table 12-10. IRQENABLE\_CLR Register Field Descriptions (continued)**

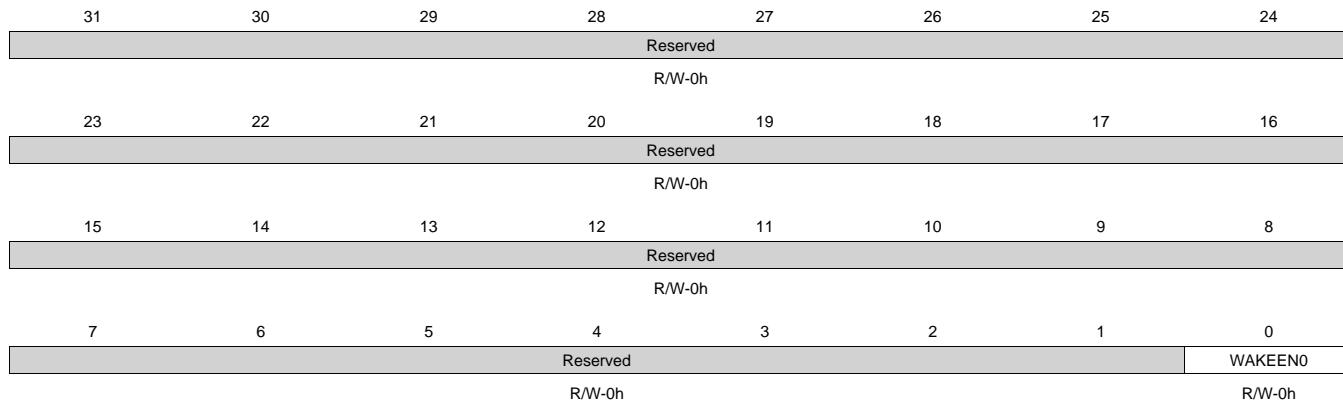
| Bit | Field                     | Type | Reset | Description  |
|-----|---------------------------|------|-------|--|
| 2   | FIFO0_Threshold           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 1   | End_of_Sequence           | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |
| 0   | HW_Pen_Event_asynchronous | R/W  | 0h    | Write 0 = No action.<br>Read 0 = Interrupt disabled (masked).<br>Read 1 = Interrupt enabled.<br>Write 1 = Disable interrupt. |

### 12.5.1.7 IRQWAKEUP Register (offset = 34h) [reset = 0h]

IRQWAKEUP is shown in [Figure 12-11](#) and described in [Table 12-11](#).

IRQ wakeup enable

**Figure 12-11. IRQWAKEUP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-11. IRQWAKEUP Register Field Descriptions**

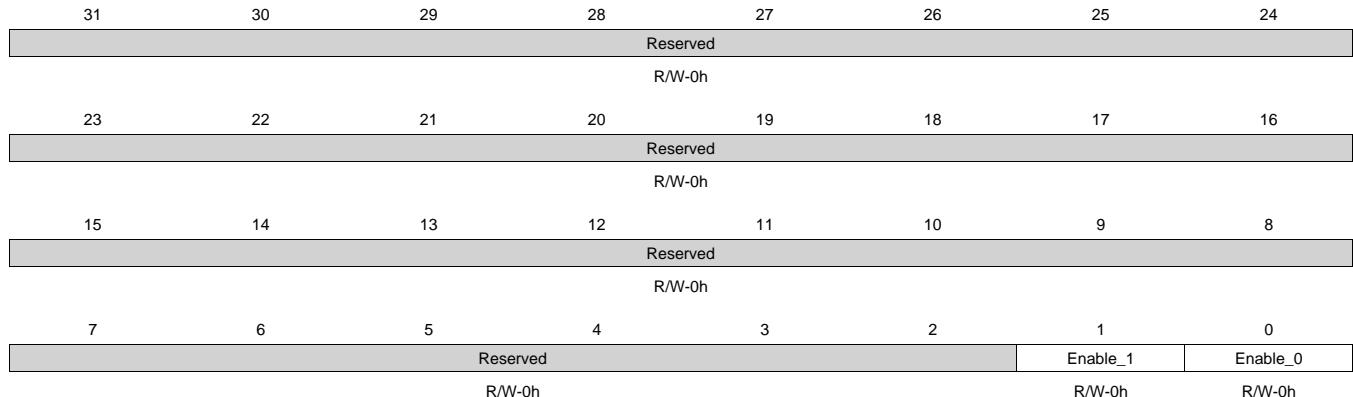
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-1 | Reserved | R/W  | 0h    |  |
| 0    | WAKEEN0  | R/W  | 0h    | Wakeup generation for HW Pen event.<br>0 = Wakeup disabled.<br>1 = Wakeup enabled. |

### 12.5.1.8 DMAENABLE\_SET Register (offset = 38h) [reset = 0h]

DMAENABLE\_SET is shown in [Figure 12-12](#) and described in [Table 12-12](#).

Per-Line DMA set

**Figure 12-12. DMAENABLE\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-12. DMAENABLE\_SET Register Field Descriptions**

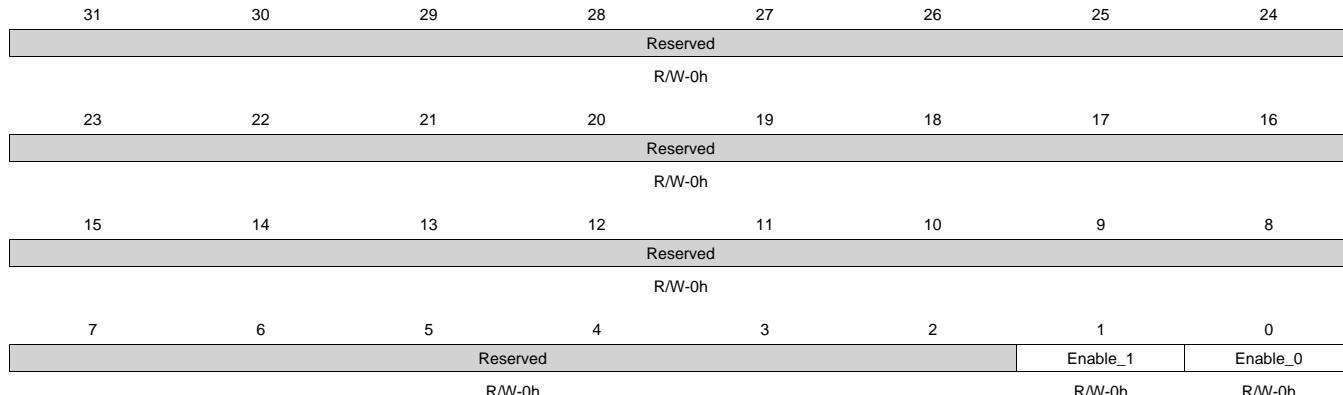
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-2 | Reserved | R/W  | 0h    |   |
| 1    | Enable_1 | R/W  | 0h    | Enable DMA request FIFO 1.<br>Write 0 = No action.<br>Read 0 = DMA line disabled.<br>Read 1 = DMA line enabled.<br>Write 1 = Enable DMA line. |
| 0    | Enable_0 | R/W  | 0h    | Enable DMA request FIFO 0.<br>Write 0 = No action.<br>Read 0 = DMA line disabled.<br>Read 1 = DMA line enabled.<br>Write 1 = Enable DMA line. |

### 12.5.1.9 DMAENABLE\_CLR Register (offset = 3Ch) [reset = 0h]

DMAENABLE\_CLR is shown in [Figure 12-13](#) and described in [Table 12-13](#).

Per-Line DMA clr

**Figure 12-13. DMAENABLE\_CLR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-13. DMAENABLE\_CLR Register Field Descriptions**

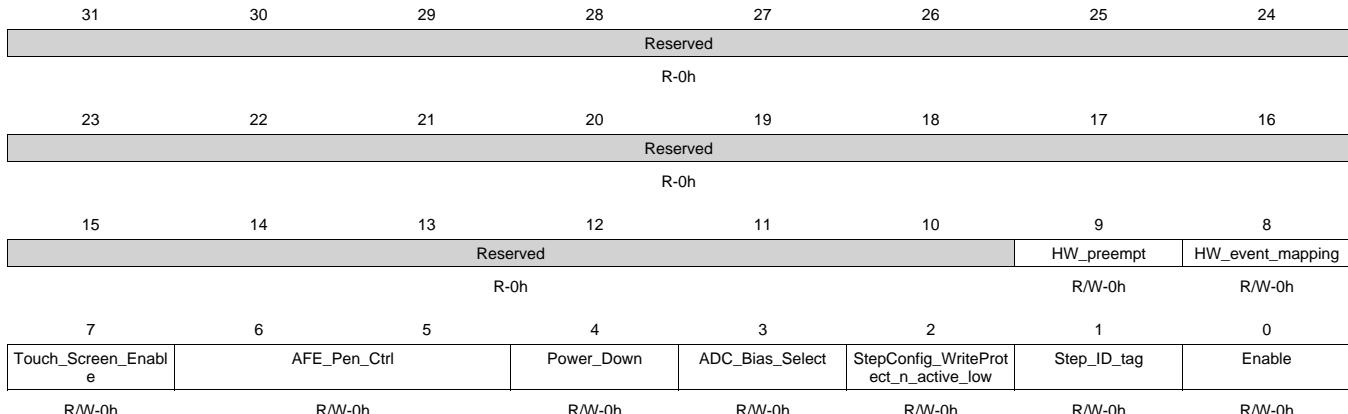
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-2 | Reserved | R/W  | 0h    |   |
| 1    | Enable_1 | R/W  | 0h    | Disable DMA request FIFO 1.<br>Write 0 = No action.<br>Read 0 = DMA line disabled.<br>Read 1 = DMA line enabled.<br>Write 1 = Disable DMA line. |
| 0    | Enable_0 | R/W  | 0h    | Disable DMA request FIFO 0.<br>Write 0 = No action.<br>Read 0 = DMA line disabled.<br>Read 1 = DMA line enabled.<br>Write 1 = Disable DMA line. |

### 12.5.1.10 CTRL Register (offset = 40h) [reset = 0h]

CTRL is shown in [Figure 12-14](#) and described in [Table 12-14](#).

@TSC\_ADC\_SS Control Register

**Figure 12-14. CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-14. CTRL Register Field Descriptions**

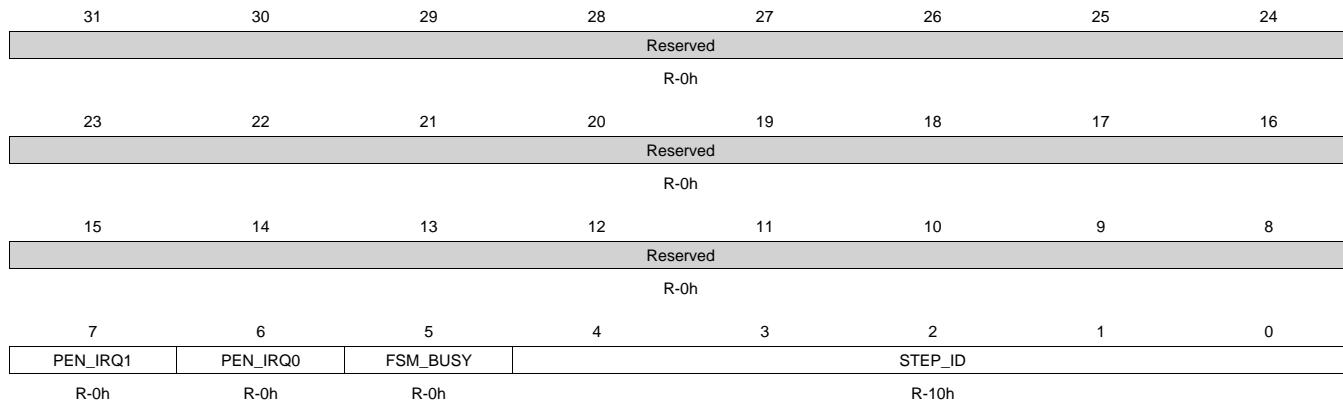
| Bit   | Field                                | Type | Reset | Description   |
|-------|--------------------------------------|------|-------|---|
| 31-10 | Reserved                             | R    | 0h    |   |
| 9     | HW_preempt                           | R/W  | 0h    | 0 = SW steps are not pre-empted by HW events.<br>1 = SW steps are pre-empted by HW events   |
| 8     | HW_event_mapping                     | R/W  | 0h    | 0 = Map HW event to Pen touch irq (from AFE).<br>1 = Map HW event to HW event input.  |
| 7     | Touch_Screen_Enable                  | R/W  | 0h    | 0 = Touchscreen transistors disabled.<br>1 = Touchscreen transistors enabled  |
| 6-5   | AFE_Pen_Ctrl                         | R/W  | 0h    | These two bits are sent directly to the AFE Pen Ctrl inputs.<br>Bit 6 controls the Wiper touch (5 wire modes)Bit 5 controls the X+ touch (4 wire modes)User also needs to make sure the ground path is connected properly for pen interrupt to occur (using the StepConfig registers)Refer to section 4 interrupts for more information |
| 4     | Power_Down                           | R/W  | 0h    | ADC Power Down control.<br>0 = AFE is powered up (default).<br>1 = Write 1 to power down AFE (the tsc_adc_ss enable (bit 0) should also be set to off)  |
| 3     | ADC_Bias_Select                      | R/W  | 0h    | Select Bias to AFE.<br>0 = Internal.<br>1 = Reserved.   |
| 2     | StepConfig_WriteProtect_n_active_low | R/W  | 0h    | 0 = Step configuration registers are protected (not writable).<br>1 = Step configuration registers are not protected (writable).  |
| 1     | Step_ID_tag                          | R/W  | 0h    | Writing 1 to this bit will store the Step ID number with the captured ADC data in the FIFO.<br>0 = Write zeroes.<br>1 = Store the channel ID tag.   |
| 0     | Enable                               | R/W  | 0h    | TSC_ADC_SS module enable bit.<br>After programming all the steps and configuration registers, write a 1to this bit to turn on TSC_ADC_SS.<br>Writing a 0 will disable the module (after the current conversion).  |

### 12.5.1.11 ADCSTAT Register (offset = 44h) [reset = 10h]

ADCSTAT is shown in [Figure 12-15](#) and described in [Table 12-15](#).

General Status bits @TSC\_ADC\_SS\_Sequencer\_Status Register

**Figure 12-15. ADCSTAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-15. ADCSTAT Register Field Descriptions**

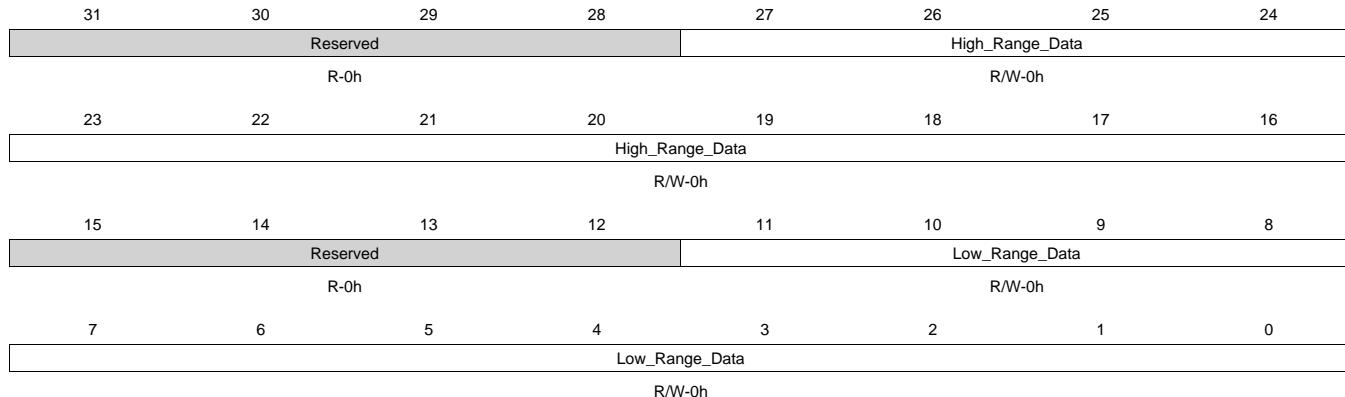
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    | -   |
| 7    | PEN_IRQ1 | R    | 0h    | PEN_IRQ[1] status   |
| 6    | PEN_IRQ0 | R    | 0h    | PEN_IRQ[0] status   |
| 5    | FSM_BUSY | R    | 0h    | Status of OCP FSM and ADC FSM.<br>0 = Idle.<br>1 = Busy.  |
| 4-0  | STEP_ID  | R    | 10h   | Encoded values::<br>10000 = Idle.<br>10001 = Charge.<br>00000 = Step 1.<br>00001 = Step 2.<br>00010 = Step 3.<br>00011 = Step 4.<br>00100 = Step 5.<br>00101 = Step 6.<br>00110 = Step 7.<br>00111 = Step 8.<br>01000 = Step 9.<br>01001 = Step 10.<br>01010 = Step 11.<br>01011 = Step 12.<br>01100 = Step 13.<br>01101 = Step 14.<br>01110 = Step 15.<br>01111 = Step 16. |

### 12.5.1.12 ADCRANGE Register (offset = 48h) [reset = 0h]

ADCRANGE is shown in [Figure 12-16](#) and described in [Table 12-16](#).

High and Low Range Threshold@TSC\_ADC\_SS\_Range\_Check Register

**Figure 12-16. ADCRANGE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-16. ADCRANGE Register Field Descriptions**

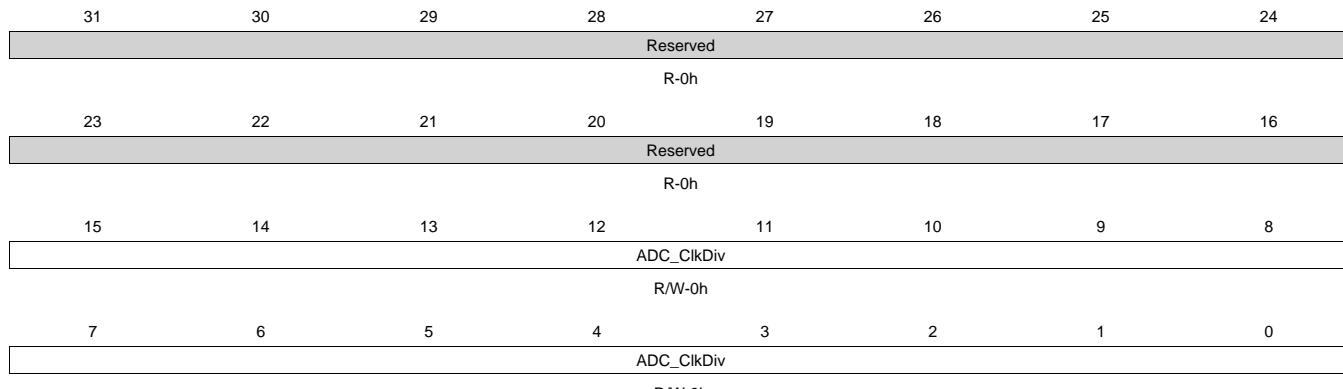
| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-28 | Reserved        | R    | 0h    |   |
| 27-16 | High_Range_Data | R/W  | 0h    | Sampled ADC data is compared to this value.<br>If the sampled data is greater than the value, then an interrupt is generated. |
| 15-12 | Reserved        | R    | 0h    | Reserved.   |
| 11-0  | Low_Range_Data  | R/W  | 0h    | Sampled ADC data is compared to this value.<br>If the sampled data is less than the value, then an interrupt is generated.    |

### 12.5.1.13 ADC\_CLKDIV Register (offset = 4Ch) [reset = 0h]

ADC\_CLKDIV is shown in [Figure 12-17](#) and described in [Table 12-17](#).

ADC clock divider register@TSC\_ADC\_SS\_Clock\_Divider Register

**Figure 12-17. ADC\_CLKDIV Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-17. ADC\_CLKDIV Register Field Descriptions**

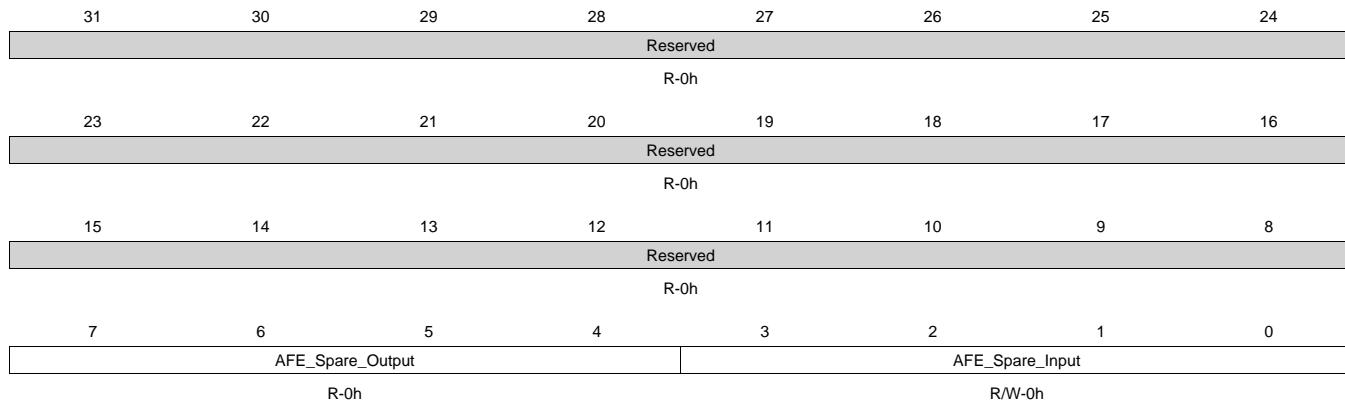
| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-16 | Reserved   | R    | 0h    |  |
| 15-0  | ADC_ClkDiv | R/W  | 0h    | The input ADC clock will be divided by this value and sent to the AFE.<br>Program to the value minus 1 |

### 12.5.1.14 ADC\_MISC Register (offset = 50h) [reset = 0h]

ADC\_MISC is shown in [Figure 12-18](#) and described in [Table 12-18](#).

AFE misc debug@TSC\_ADC\_SS\_MISC Register

**Figure 12-18. ADC\_MISC Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-18. ADC\_MISC Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description  |
|------|------------------|------|-------|--|
| 31-8 | Reserved         | R    | 0h    | RESERVED.  |
| 7-4  | AFE_Spare_Output | R    | 0h    | Connected to AFE Spare Output pins.<br>Reserved in normal operation. |
| 3-0  | AFE_Spare_Input  | R/W  | 0h    | Connected to AFE Spare Input pins.<br>Reserved in normal operation.  |

### 12.5.1.15 STEPENABLE Register (offset = 54h) [reset = 0h]

STEPENABLE is shown in [Figure 12-19](#) and described in [Table 12-19](#).

Step Enable

**Figure 12-19. STEPENABLE Register**

| 31       | 30     | 29     | 28     | 27     | 26     | 25     | 24        |
|----------|--------|--------|--------|--------|--------|--------|-----------|
| Reserved |        |        |        |        |        |        |           |
| R-0h     |        |        |        |        |        |        |           |
| 23       | 22     | 21     | 20     | 19     | 18     | 17     | 16        |
| Reserved |        |        |        |        |        |        |           |
| R-0h     |        |        |        |        |        |        |           |
| 15       | 14     | 13     | 12     | 11     | 10     | 9      | 8         |
| STEP15   | STEP14 | STEP13 | STEP12 | STEP11 | STEP10 | STEP9  | STEP8     |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h    |
| 7        | 6      | 5      | 4      | 3      | 2      | 1      | 0         |
| STEP7    | STEP6  | STEP5  | STEP4  | STEP3  | STEP2  | STEP1  | TS_Charge |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-19. STEPENABLE Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description           |
|-------|-----------|------|-------|-----------------------|
| 31-17 | Reserved  | R    | 0h    | RESERVED.             |
| 16    | STEP16    | R/W  | 0h    | Enable step 16        |
| 15    | STEP15    | R/W  | 0h    | Enable step 15        |
| 14    | STEP14    | R/W  | 0h    | Enable step 14        |
| 13    | STEP13    | R/W  | 0h    | Enable step 13        |
| 12    | STEP12    | R/W  | 0h    | Enable step 12        |
| 11    | STEP11    | R/W  | 0h    | Enable step 11        |
| 10    | STEP10    | R/W  | 0h    | Enable step 10        |
| 9     | STEP9     | R/W  | 0h    | Enable step 9         |
| 8     | STEP8     | R/W  | 0h    | Enable step 8         |
| 7     | STEP7     | R/W  | 0h    | Enable step 7         |
| 6     | STEP6     | R/W  | 0h    | Enable step 6         |
| 5     | STEP5     | R/W  | 0h    | Enable step 5         |
| 4     | STEP4     | R/W  | 0h    | Enable step 4         |
| 3     | STEP3     | R/W  | 0h    | Enable step 3         |
| 2     | STEP2     | R/W  | 0h    | Enable step 2         |
| 1     | STEP1     | R/W  | 0h    | Enable step 1         |
| 0     | TS_Charge | R/W  | 0h    | Enable TS Charge step |

### 12.5.1.16 IDLECONFIG Register (offset = 58h) [reset = 0h]

IDLECONFIG is shown in [Figure 12-20](#) and described in [Table 12-20](#).

Idle Step configuration@TSC\_ADC\_SS\_IDLE\_StepConfig Register

**Figure 12-20. IDLECONFIG Register**

|                  |                 |           |          |           |           |            |                  |
|------------------|-----------------|-----------|----------|-----------|-----------|------------|------------------|
| 31               | 30              | 29        | 28       | 27        | 26        | 25         | 24               |
| Reserved         |                 |           |          |           |           | Diff_CNTRL | SEL_RFMs_SWC_1_0 |
| R/W-0h           |                 |           |          |           |           | R/W-0h     | R/W-0h           |
| 23               | 22              | 21        | 20       | 19        | 18        | 17         | 16               |
| SEL_RFMs_SWC_1_0 | SEL_INP_SWC_3_0 |           |          |           |           |            | SEL_INM_SWM3_0   |
| R/W-0h           |                 |           |          |           |           | R/W-0h     |                  |
| 15               | 14              | 13        | 12       | 11        | 10        | 9          | 8                |
| SEL_INM_SWM3_0   | SEL_RFP_SWC_2_0 |           |          | WPNSW_SWC | YPNSW_SWC | XNPSW_SWC  | YNNSW_SWC        |
| R/W-0h           |                 |           |          |           |           | R/W-0h     | R/W-0h           |
| 7                | 6               | 5         | 4        | 3         | 2         | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC       | XPPSW_SWC | Reserved |           |           |            |                  |
| R/W-0h           |                 |           |          |           |           | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-20. IDLECONFIG Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-26 | Reserved         | R/W  | 0h    |  |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin.<br>0 = Single Ended.<br>1 = Differential Pair Enable.                                    |
| 24-23 | SEL_RFMs_SWC_1_0 | R/W  | 0h    | SEL_RFMs pins SW configuration.<br>00 = VSSA_ADC.<br>01 = XNUR.<br>10 = YNLR.<br>11 = VREFN.                       |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration.<br>0000 = Channel 1.<br>0111 = Channel 8.<br>1xxx = VREFN.                          |
| 18-15 | SEL_INM_SWM3_0   | R/W  | 0h    | SEL_INM pins for neg differential.<br>0000 = Channel 1.<br>0111 = Channel 8.<br>1xxx = VREFN.                      |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration.<br>000 = VDDA_ADC.<br>001 = XPUL.<br>010 = YPLL.<br>011 = VREFP.<br>1xx = Reserved. |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration   |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration   |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration   |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration   |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration   |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration   |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration   |
| 4-0   | Reserved         | R/W  | 0h    |  |

**12.5.1.17 TS\_CHARGE\_STEPCONFIG Register (offset = 5Ch) [reset = 0h]**

 TS\_CHARGE\_STEPCONFIG is shown in [Figure 12-21](#) and described in [Table 12-21](#).

TS Charge StepConfiguration@TSC\_ADC\_SS\_TS\_Charge\_StepConfig Register

**Figure 12-21. TS\_CHARGE\_STEPCONFIG Register**

| 31               | 30              | 29        | 28       | 27        | 26        | 25         | 24               |
|------------------|-----------------|-----------|----------|-----------|-----------|------------|------------------|
| Reserved         |                 |           |          |           |           | Diff_CNTRL | SEL_RFMs_SWC_1_0 |
| R/W-0h           |                 |           |          |           |           | R/W-0h     | R/W-0h           |
| 23               | 22              | 21        | 20       | 19        | 18        | 17         | 16               |
| SEL_RFMs_SWC_1_0 | SEL_INP_SWC_3_0 |           |          |           |           |            | SEL_INM_SWM3_0   |
| R/W-0h           |                 |           |          |           |           | R/W-0h     |                  |
| 15               | 14              | 13        | 12       | 11        | 10        | 9          | 8                |
| SEL_INM_SWM3_0   | SEL_RFP_SWC_2_0 |           |          | WPNSW_SWC | YPNSW_SWC | XNPSW_SWC  | YNNSW_SWC        |
| R/W-0h           |                 |           |          |           |           | R/W-0h     |                  |
| 7                | 6               | 5         | 4        | 3         | 2         | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC       | XPPSW_SWC | Reserved |           |           |            |                  |
| R/W-0h           |                 |           |          |           |           | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-21. TS\_CHARGE\_STEPCONFIG Register Field Descriptions**

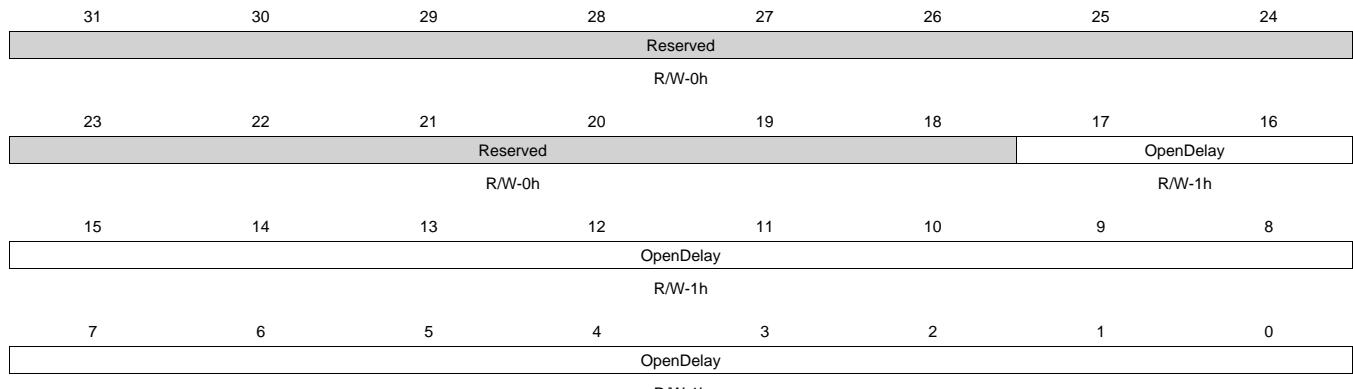
| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-26 | Reserved         | R/W  | 0h    |  |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin.<br>0 = Single Ended.<br>1 = Differential Pair Enable.                              |
| 24-23 | SEL_RFMs_SWC_1_0 | R/W  | 0h    | SEL_RFMs pins SW configuration.<br>00 = VSSA.<br>01 = XNUR.<br>10 = YNLR.<br>11 = VREFN.                     |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration.<br>0000 = Channel 1.<br>0111 = Channel 8.<br>1xxx = VREFN.                    |
| 18-15 | SEL_INM_SWM3_0   | R/W  | 0h    | SEL_INM pins for neg differential.<br>0000 = Channel 1.<br>0111 = Channel 8.<br>1xxx = VREFN.                |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration.<br>000 = VDDA.<br>001 = XPUL.<br>010 = YPLL.<br>011 = VREFP.<br>1xx = INTREF. |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration   |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration   |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration   |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration   |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration   |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration   |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration   |
| 4-0   | Reserved         | R/W  | 0h    |  |

### 12.5.1.18 TS\_CHARGE\_DELAY Register (offset = 60h) [reset = 1h]

TS\_CHARGE\_DELAY is shown in [Figure 12-22](#) and described in [Table 12-22](#).

TS Charge Delay Register@TSC\_ADC\_SS\_TS\_Charge\_StepDelay Register

**Figure 12-22. TS\_CHARGE\_DELAY Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-22. TS\_CHARGE\_DELAY Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-18 | Reserved  | R/W  | 0h    |  |
| 17-0  | OpenDelay | R/W  | 1h    | Program the # of ADC clock cycles to wait between applying the step configuration registers and going back to the IDLE state.<br>(Value must be greater than 0.) |

### 12.5.1.19 STEPCONFIG1 Register (offset = 64h) [reset = 0h]

STEPCONFIG1 is shown in Figure 12-23 and described in Table 12-23.

Step Configuration 1

**Figure 12-23. STEPCONFIG1 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-23. STEPCONFIG1 Register Field Descriptions**

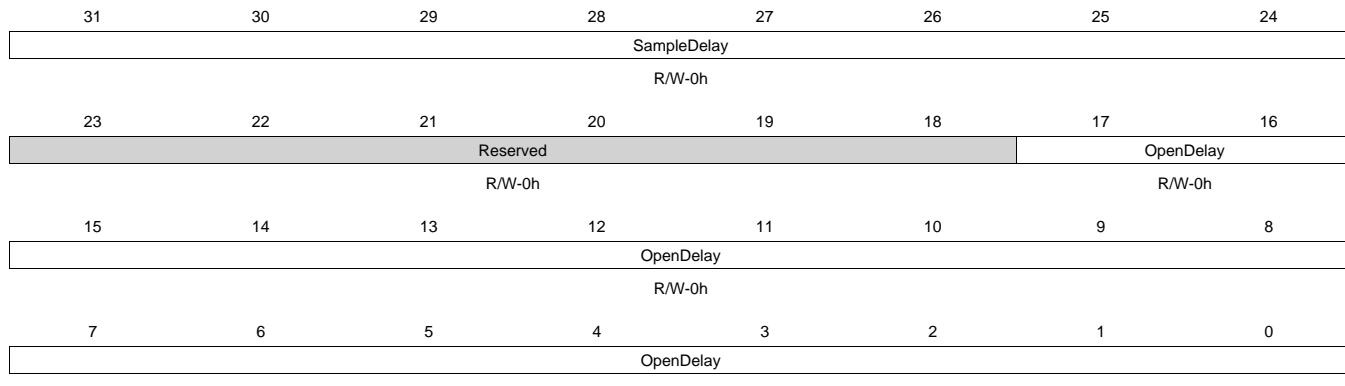
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO 1.  |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samplings to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                    |

### 12.5.1.20 STEPDELAY1 Register (offset = 68h) [reset = 0h]

STEPDELAY1 is shown in [Figure 12-24](#) and described in [Table 12-24](#).

Step Delay Register 1

**Figure 12-24. STEPDELAY1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-24. STEPDELAY1 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.21 STEPCONFIG2 Register (offset = 6Ch) [reset = 0h]

STEPCONFIG2 is shown in Figure 12-25 and described in Table 12-25.

Step Configuration 2

**Figure 12-25. STEPCONFIG2 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-25. STEPCONFIG2 Register Field Descriptions**

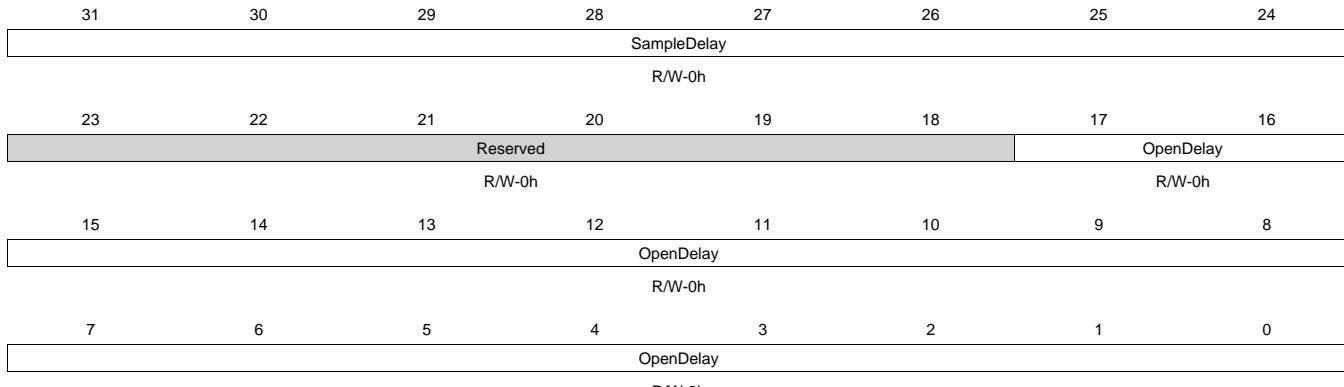
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.22 STEPDELAY2 Register (offset = 70h) [reset = 0h]

STEPDELAY2 is shown in [Figure 12-26](#) and described in [Table 12-26](#).

Step Delay Register 2

**Figure 12-26. STEPDELAY2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-26. STEPDELAY2 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.23 STEPCONFIG3 Register (offset = 74h) [reset = 0h]

STEPCONFIG3 is shown in [Figure 12-27](#) and described in [Table 12-27](#).

Step Configuration 3

**Figure 12-27. STEPCONFIG3 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-27. STEPCONFIG3 Register Field Descriptions**

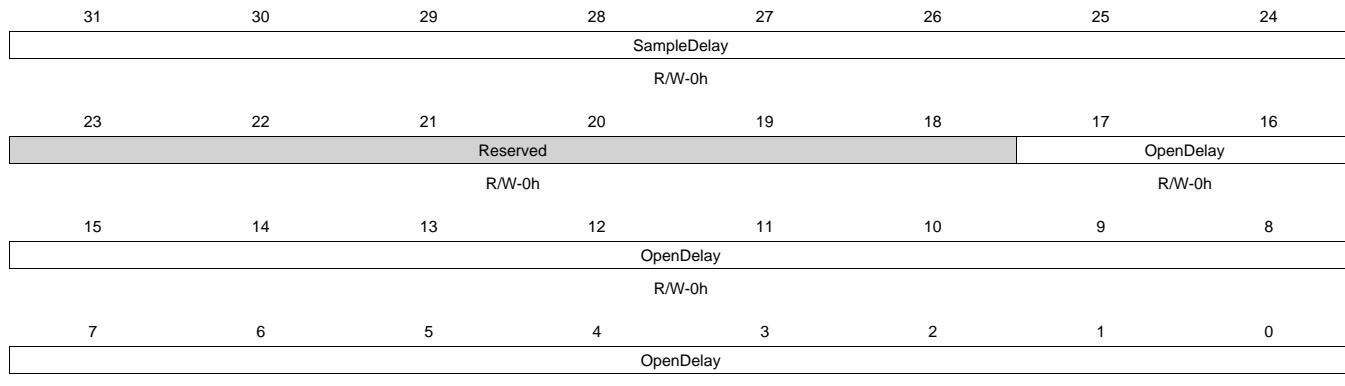
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.24 STEPDELAY3 Register (offset = 78h) [reset = 0h]

STEPDELAY3 is shown in [Figure 12-28](#) and described in [Table 12-28](#).

Step Delay Register 3

**Figure 12-28. STEPDELAY3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-28. STEPDELAY3 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.25 STEPCONFIG4 Register (offset = 7Ch) [reset = 0h]

STEPCONFIG4 is shown in Figure 12-29 and described in Table 12-29.

Step Configuration 4

**Figure 12-29. STEPCONFIG4 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-29. STEPCONFIG4 Register Field Descriptions**

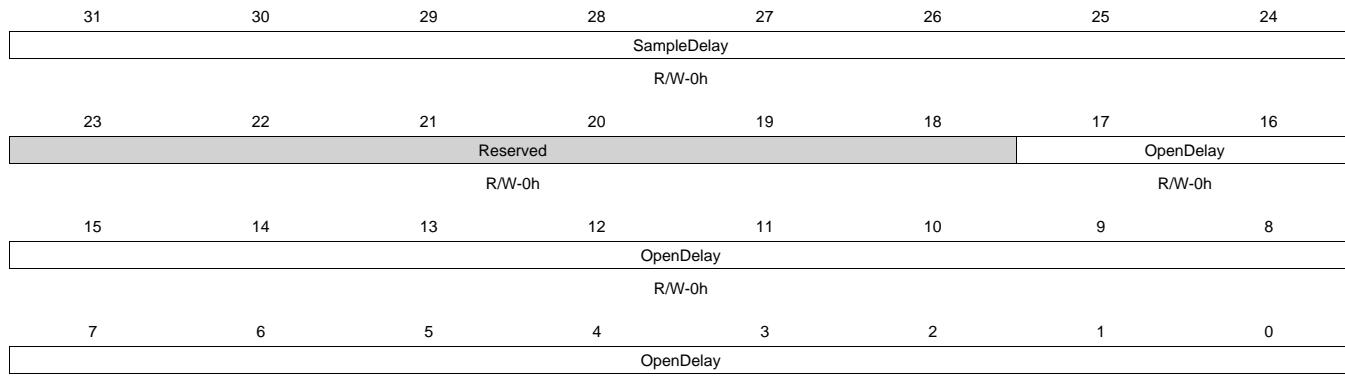
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.26 STEPDELAY4 Register (offset = 80h) [reset = 0h]

STEPDELAY4 is shown in [Figure 12-30](#) and described in [Table 12-30](#).

Step Delay Register 4

**Figure 12-30. STEPDELAY4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-30. STEPDELAY4 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.27 STEPCONFIG5 Register (offset = 84h) [reset = 0h]

STEPCONFIG5 is shown in Figure 12-31 and described in Table 12-31.

Step Configuration 5

**Figure 12-31. STEPCONFIG5 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-31. STEPCONFIG5 Register Field Descriptions**

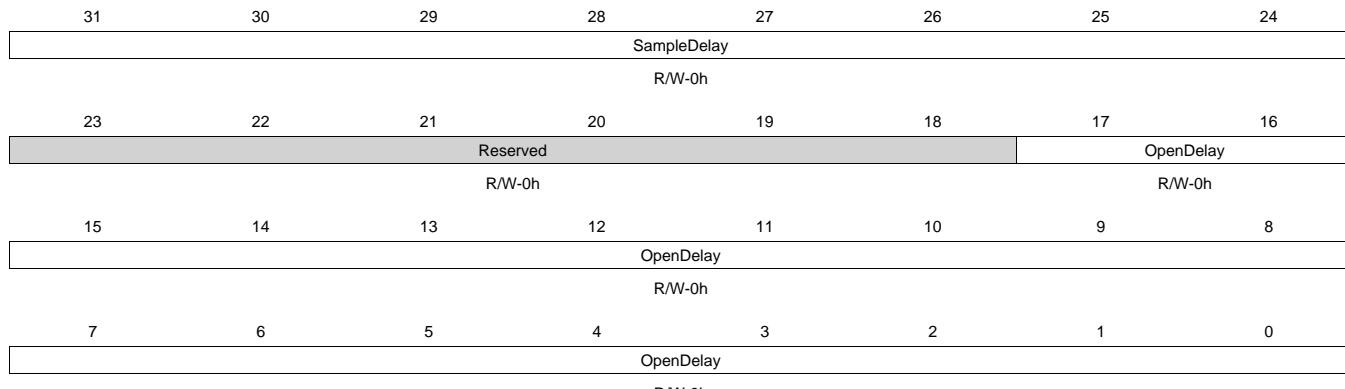
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.28 STEPDELAY5 Register (offset = 88h) [reset = 0h]

STEPDELAY5 is shown in [Figure 12-32](#) and described in [Table 12-32](#).

Step Delay Register 5

**Figure 12-32. STEPDELAY5 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-32. STEPDELAY5 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.29 STEPCONFIG6 Register (offset = 8Ch) [reset = 0h]

STEPCONFIG6 is shown in Figure 12-33 and described in Table 12-33.

Step Configuration 6

**Figure 12-33. STEPCONFIG6 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-33. STEPCONFIG6 Register Field Descriptions**

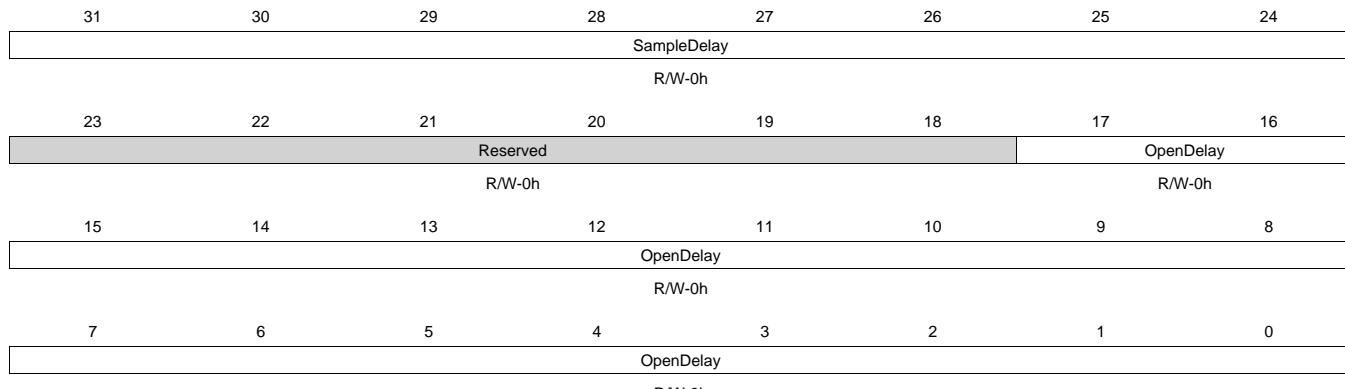
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.30 STEPDELAY6 Register (offset = 90h) [reset = 0h]

STEPDELAY6 is shown in [Figure 12-34](#) and described in [Table 12-34](#).

Step Delay Register 6

**Figure 12-34. STEPDELAY6 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-34. STEPDELAY6 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.31 STEPCONFIG7 Register (offset = 94h) [reset = 0h]

STEPCONFIG7 is shown in Figure 12-35 and described in Table 12-35.

Step Configuration 7

**Figure 12-35. STEPCONFIG7 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-35. STEPCONFIG7 Register Field Descriptions**

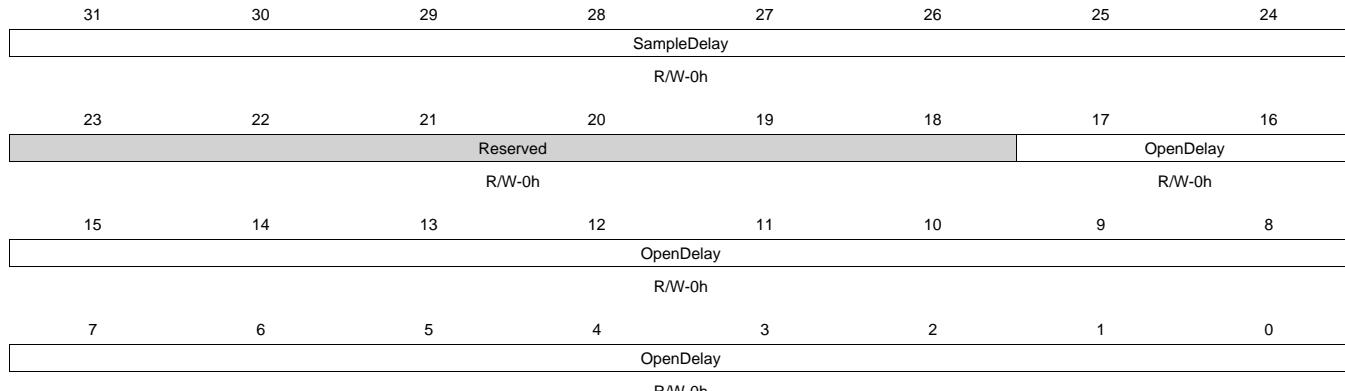
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.32 STEPDELAY7 Register (offset = 98h) [reset = 0h]

STEPDELAY7 is shown in [Figure 12-36](#) and described in [Table 12-36](#).

Step Delay Register 7

**Figure 12-36. STEPDELAY7 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-36. STEPDELAY7 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.33 STEPCONFIG8 Register (offset = 9Ch) [reset = 0h]

STEPCONFIG8 is shown in Figure 12-37 and described in Table 12-37.

Step Configuration 8

**Figure 12-37. STEPCONFIG8 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-37. STEPCONFIG8 Register Field Descriptions**

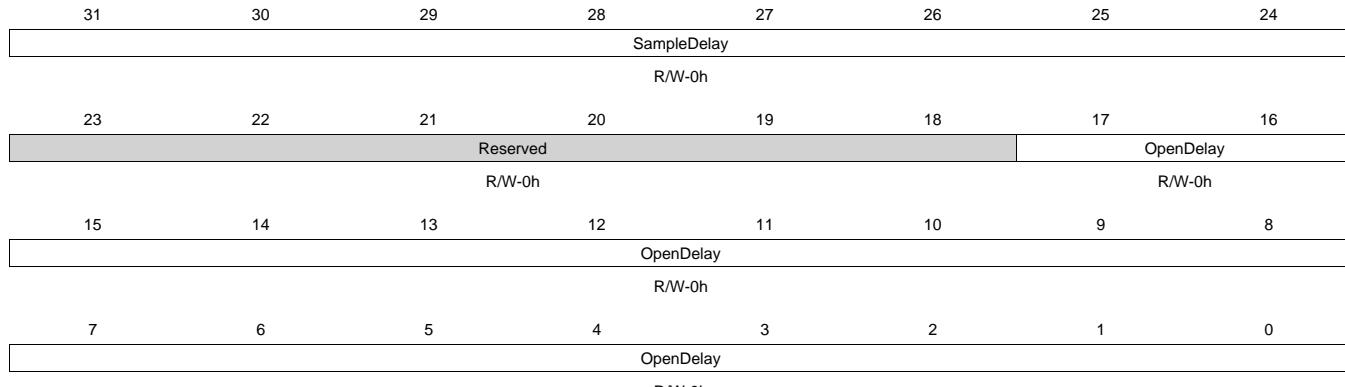
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.34 STEPDELAY8 Register (offset = A0h) [reset = 0h]

STEPDELAY8 is shown in [Figure 12-38](#) and described in [Table 12-38](#).

Step Delay Register 8

**Figure 12-38. STEPDELAY8 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-38. STEPDELAY8 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.35 STEPCONFIG9 Register (offset = A4h) [reset = 0h]

STEPCONFIG9 is shown in Figure 12-39 and described in Table 12-39.

Step Configuration 9

**Figure 12-39. STEPCONFIG9 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-39. STEPCONFIG9 Register Field Descriptions**

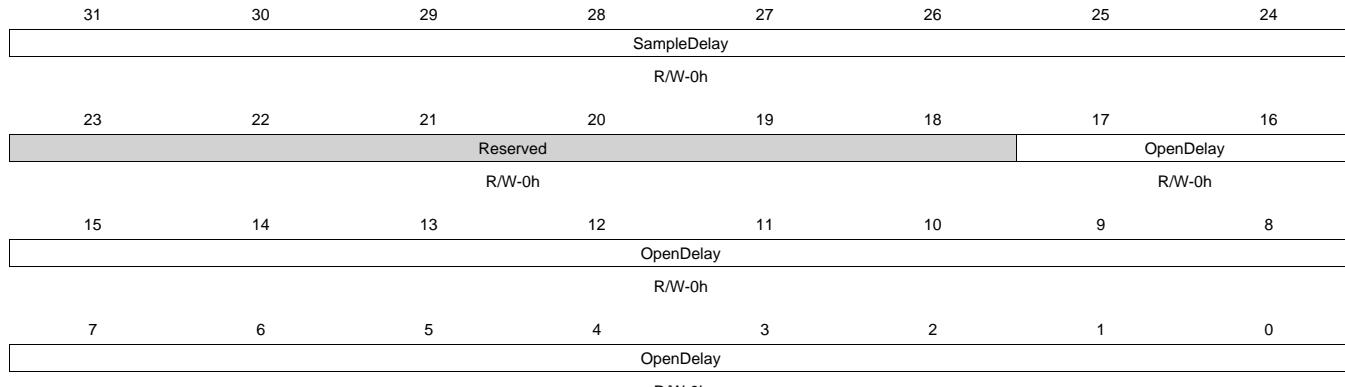
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO 1.<br>= FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.36 STEPDELAY9 Register (offset = A8h) [reset = 0h]

STEPDELAY9 is shown in [Figure 12-40](#) and described in [Table 12-40](#).

Step Delay Register 9

**Figure 12-40. STEPDELAY9 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-40. STEPDELAY9 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.37 STEPCONFIG10 Register (offset = ACh) [reset = 0h]

STEPCONFIG10 is shown in [Figure 12-41](#) and described in [Table 12-41](#).

Step Configuration 10

**Figure 12-41. STEPCONFIG10 Register**

| 31               | 30        | 29              | 28     | 27          | 26              | 25         | 24               |
|------------------|-----------|-----------------|--------|-------------|-----------------|------------|------------------|
|                  |           | Reserved        |        | Range_check | FIFO_select     | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19          | 18              | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        |             | SEL_INM_SWC_3_0 |            |                  |
|                  |           | R/W-0h          |        | R/W-0h      |                 | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11          | 10              | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC   | YPNSW_SWC       | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h      | R/W-0h          | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3           | 2               | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging   |                 | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h      |                 | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-41. STEPCONFIG10 Register Field Descriptions**

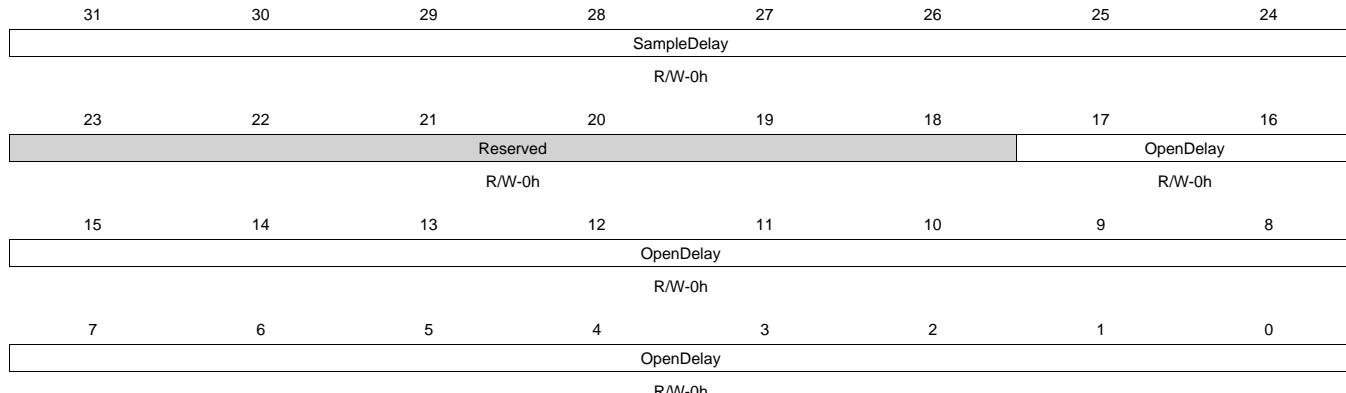
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.38 STEPDELAY10 Register (offset = B0h) [reset = 0h]

STEPDELAY10 is shown in [Figure 12-42](#) and described in [Table 12-42](#).

Step Delay Register 10

**Figure 12-42. STEPDELAY10 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-42. STEPDELAY10 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.39 STEPCONFIG11 Register (offset = B4h) [reset = 0h]

STEPCONFIG11 is shown in [Figure 12-43](#) and described in [Table 12-43](#).

Step Configuration 11

**Figure 12-43. STEPCONFIG11 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-43. STEPCONFIG11 Register Field Descriptions**

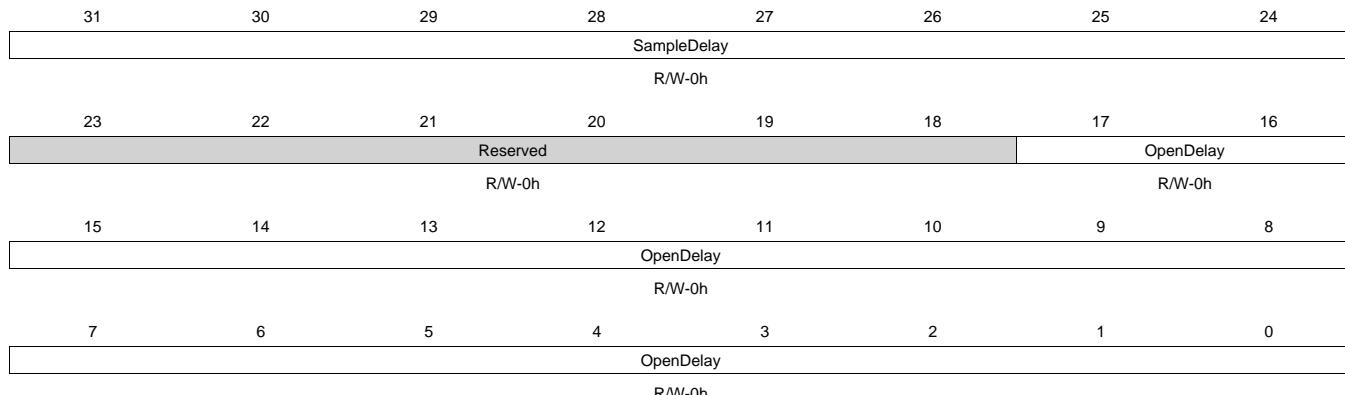
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.40 STEPDELAY11 Register (offset = B8h) [reset = 0h]

STEPDELAY11 is shown in [Figure 12-44](#) and described in [Table 12-44](#).

Step Delay Register 11

**Figure 12-44. STEPDELAY11 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-44. STEPDELAY11 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.41 STEPCONFIG12 Register (offset = BCh) [reset = 0h]

STEPCONFIG12 is shown in [Figure 12-45](#) and described in [Table 12-45](#).

Step Configuration 12

**Figure 12-45. STEPCONFIG12 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-45. STEPCONFIG12 Register Field Descriptions**

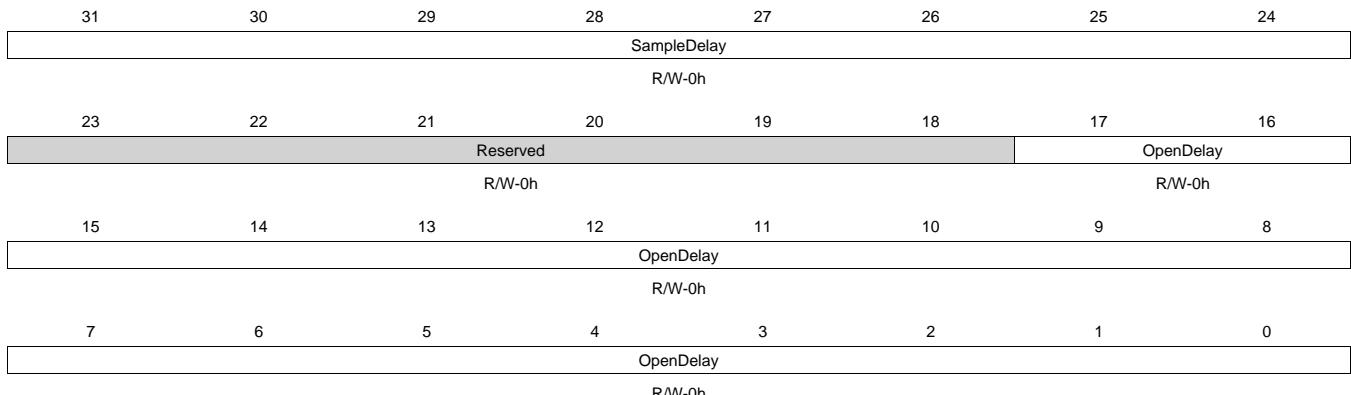
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.42 STEPDELAY12 Register (offset = C0h) [reset = 0h]

STEPDELAY12 is shown in [Figure 12-46](#) and described in [Table 12-46](#).

Step Delay Register 12

**Figure 12-46. STEPDELAY12 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-46. STEPDELAY12 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.43 STEPCONFIG13 Register (offset = C4h) [reset = 0h]

STEPCONFIG13 is shown in [Figure 12-47](#) and described in [Table 12-47](#).

Step Configuration 13

**Figure 12-47. STEPCONFIG13 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-47. STEPCONFIG13 Register Field Descriptions**

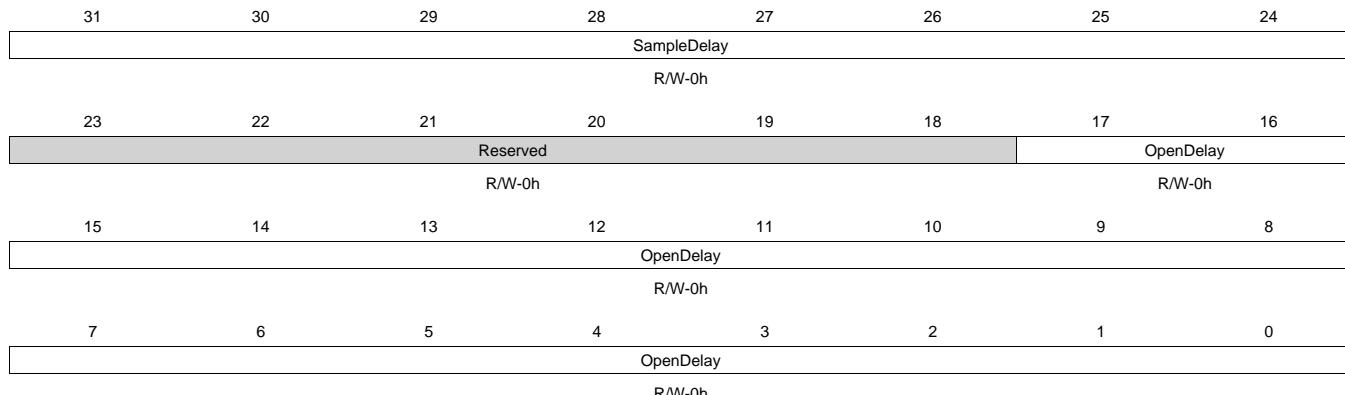
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.44 STEPDELAY13 Register (offset = C8h) [reset = 0h]

STEPDELAY13 is shown in [Figure 12-48](#) and described in [Table 12-48](#).

Step Delay Register 13

**Figure 12-48. STEPDELAY13 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-48. STEPDELAY13 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.45 STEPCONFIG14 Register (offset = CCh) [reset = 0h]

STEPCONFIG14 is shown in [Figure 12-49](#) and described in [Table 12-49](#).

Step Configuration 14

**Figure 12-49. STEPCONFIG14 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-49. STEPCONFIG14 Register Field Descriptions**

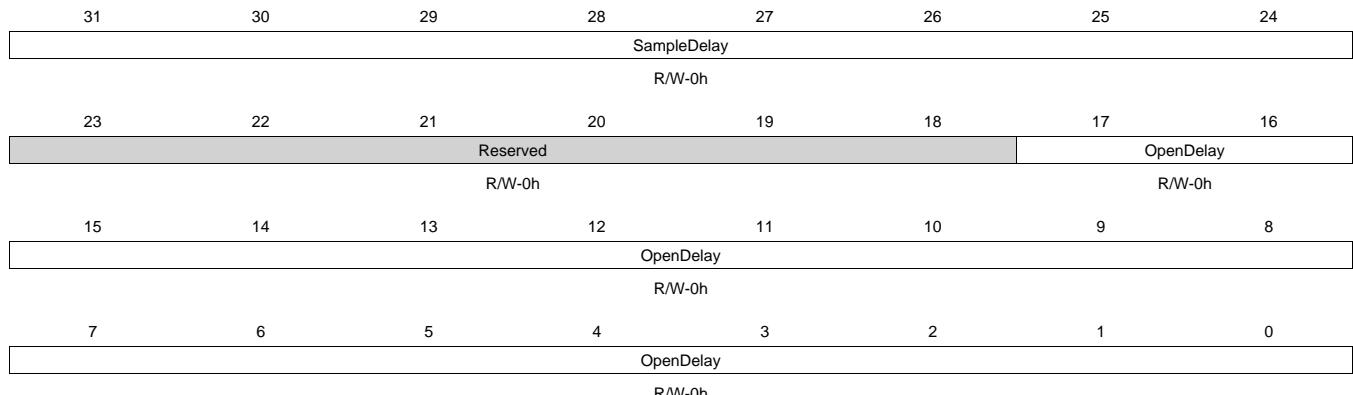
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.46 STEPDELAY14 Register (offset = D0h) [reset = 0h]

STEPDELAY14 is shown in [Figure 12-50](#) and described in [Table 12-50](#).

Step Delay Register 14

**Figure 12-50. STEPDELAY14 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-50. STEPDELAY14 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.47 STEPCONFIG15 Register (offset = D4h) [reset = 0h]

STEPCONFIG15 is shown in [Figure 12-51](#) and described in [Table 12-51](#).

Step Configuration 15

**Figure 12-51. STEPCONFIG15 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-51. STEPCONFIG15 Register Field Descriptions**

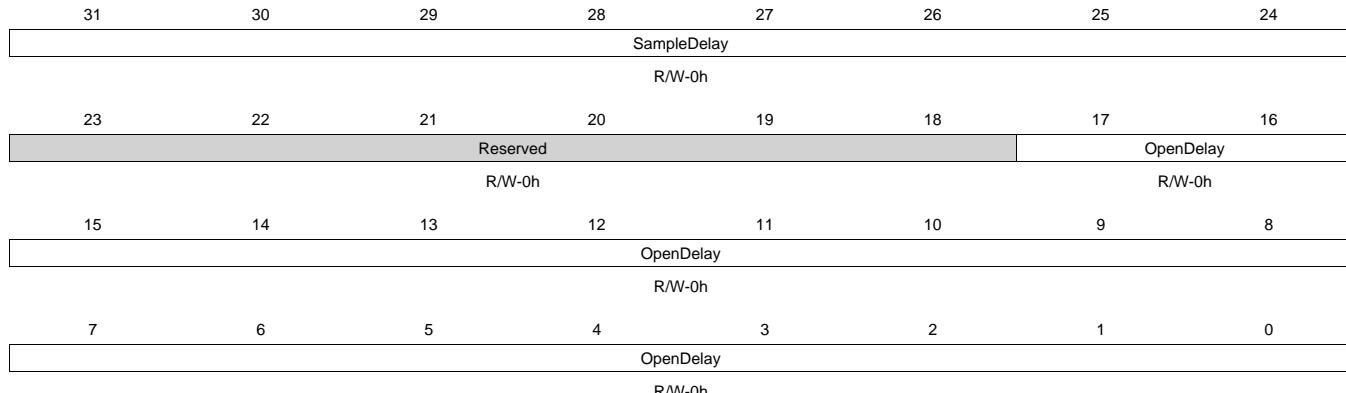
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.48 STEPDELAY15 Register (offset = D8h) [reset = 0h]

STEPDELAY15 is shown in [Figure 12-52](#) and described in [Table 12-52](#).

Step Delay Register 15

**Figure 12-52. STEPDELAY15 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-52. STEPDELAY15 Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.49 STEPCONFIG16 Register (offset = DCh) [reset = 0h]

STEPCONFIG16 is shown in [Figure 12-53](#) and described in [Table 12-53](#).

Step Configuration 16

**Figure 12-53. STEPCONFIG16 Register**

| 31               | 30        | 29              | 28     | 27              | 26          | 25         | 24               |
|------------------|-----------|-----------------|--------|-----------------|-------------|------------|------------------|
|                  |           | Reserved        |        | Range_check     | FIFO_select | Diff_CNTRL | SEL_RFIM_SWC_1_0 |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 23               | 22        | 21              | 20     | 19              | 18          | 17         | 16               |
| SEL_RFIM_SWC_1_0 |           | SEL_INP_SWC_3_0 |        | SEL_INM_SWC_3_0 |             |            |                  |
|                  |           | R/W-0h          |        | R/W-0h          |             | R/W-0h     |                  |
| 15               | 14        | 13              | 12     | 11              | 10          | 9          | 8                |
| SEL_INM_SWC_3_0  |           | SEL_RFP_SWC_2_0 |        | WPNSW_SWC       | YPNSW_SWC   | XNPSW_SWC  | YNNSW_SWC        |
|                  |           | R/W-0h          |        | R/W-0h          | R/W-0h      | R/W-0h     | R/W-0h           |
| 7                | 6         | 5               | 4      | 3               | 2           | 1          | 0                |
| YPPSW_SWC        | XNNSW_SWC | XPPSW_SWC       |        | Averaging       |             | Mode       |                  |
|                  | R/W-0h    | R/W-0h          | R/W-0h | R/W-0h          |             | R/W-0h     |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-53. STEPCONFIG16 Register Field Descriptions**

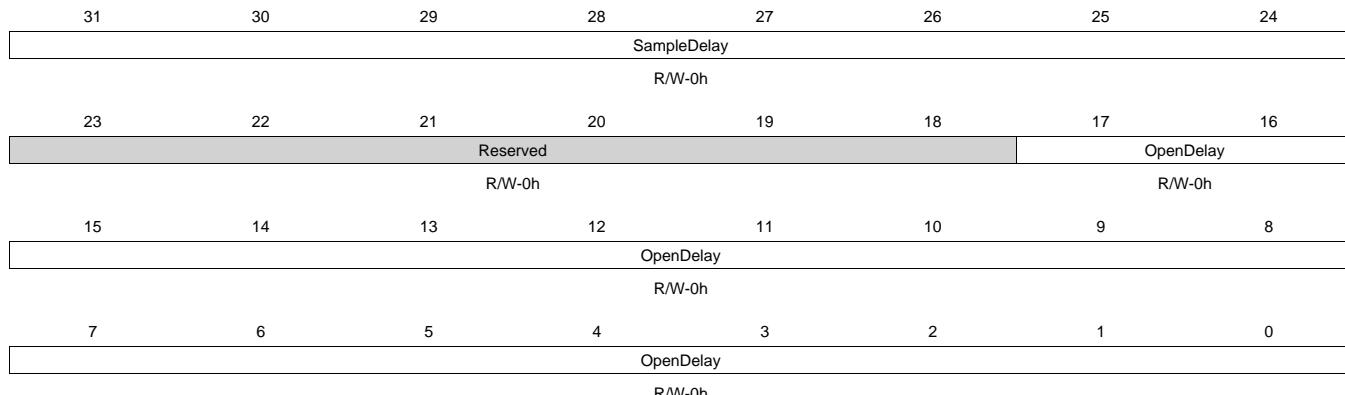
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-28 | Reserved         | R/W  | 0h    |   |
| 27    | Range_check      | R/W  | 0h    | 0 = Disable out-of-range check.<br>1 = Compare ADC data with range check register.  |
| 26    | FIFO_select      | R/W  | 0h    | Sampled data will be stored in FIFO.<br>0 = FIFO.<br>1 = FIFO1.   |
| 25    | Diff_CNTRL       | R/W  | 0h    | Differential Control Pin  |
| 24-23 | SEL_RFIM_SWC_1_0 | R/W  | 0h    | SEL_RFIM pins SW configuration  |
| 22-19 | SEL_INP_SWC_3_0  | R/W  | 0h    | SEL_INP pins SW configuration   |
| 18-15 | SEL_INM_SWC_3_0  | R/W  | 0h    | SEL_INM pins for negative differential  |
| 14-12 | SEL_RFP_SWC_2_0  | R/W  | 0h    | SEL_RFP pins SW configuration   |
| 11    | WPNSW_SWC        | R/W  | 0h    | WPNSW pin SW configuration  |
| 10    | YPNSW_SWC        | R/W  | 0h    | YPNSW pin SW configuration  |
| 9     | XNPSW_SWC        | R/W  | 0h    | XNPSW pin SW configuration  |
| 8     | YNNSW_SWC        | R/W  | 0h    | YNNSW pin SW configuration  |
| 7     | YPPSW_SWC        | R/W  | 0h    | YPPSW pin SW configuration  |
| 6     | XNNSW_SWC        | R/W  | 0h    | XNNSW pin SW configuration  |
| 5     | XPPSW_SWC        | R/W  | 0h    | XPPSW pin SW configuration  |
| 4-2   | Averaging        | R/W  | 0h    | Number of samples to average:<br>000 = No average.<br>001 = 2 samples average.<br>010 = 4 samples average.<br>011 = 8 samples average.<br>100 = 16 samples average. |
| 1-0   | Mode             | R/W  | 0h    | 00 = SW enabled, one-shot.<br>01 = SW enabled, continuous.<br>10 = HW synchronized, one-shot.<br>11 = HW synchronized, continuous.                                  |

### 12.5.1.50 STEPDELAY16 Register (offset = E0h) [reset = 0h]

STEPDELAY16 is shown in [Figure 12-54](#) and described in [Table 12-54](#).

Step Delay Register 16

**Figure 12-54. STEPDELAY16 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-54. STEPDELAY16 Register Field Descriptions**

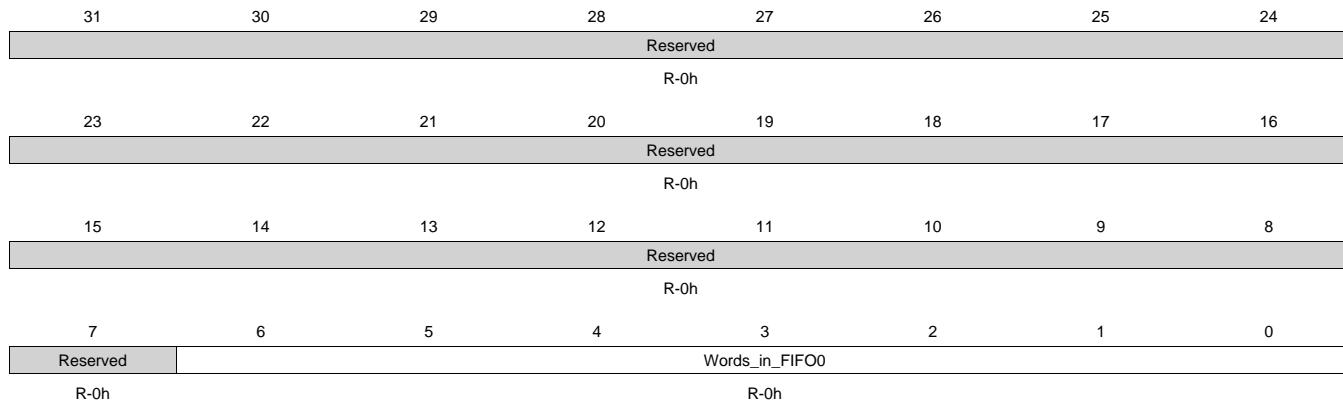
| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-24 | SampleDelay | R/W  | 0h    | This register will control the number of ADC clock cycles to sample (hold SOC high). Any value programmed here will be added to the minimum requirement of 1 clock cycle. |
| 23-18 | Reserved    | R/W  | 0h    |   |
| 17-0  | OpenDelay   | R/W  | 0h    | Program the number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion                             |

### 12.5.1.51 FIFO0COUNT Register (offset = E4h) [reset = 0h]

FIFO0COUNT is shown in [Figure 12-55](#) and described in [Table 12-55](#).

FIFO0 word count@TSC\_ADC\_SS\_FIFO0 Word Count Register

**Figure 12-55. FIFO0COUNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-55. FIFO0COUNT Register Field Descriptions**

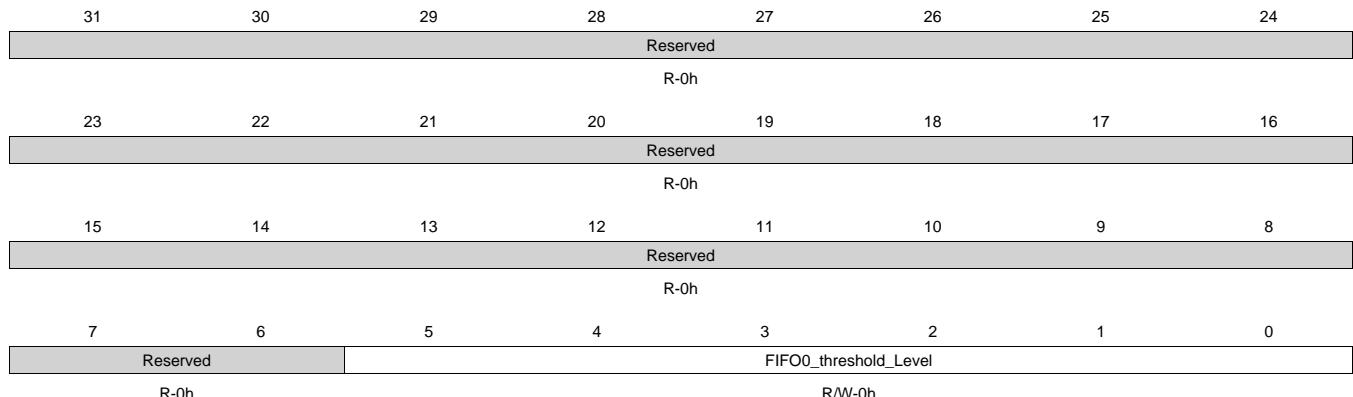
| Bit  | Field          | Type | Reset | Description                            |
|------|----------------|------|-------|--|
| 31-7 | Reserved       | R    | 0h    | RESERVED                               |
| 6-0  | Words_in_FIFO0 | R    | 0h    | Number of words currently in the FIFO0 |

### 12.5.1.52 FIFO0THRESHOLD Register (offset = E8h) [reset = 0h]

FIFO0THRESHOLD is shown in [Figure 12-56](#) and described in [Table 12-56](#).

FIFO0 Threshold trigger@TSC\_ADC\_SS\_FIFO0 Threshold Level Register

**Figure 12-56. FIFO0THRESHOLD Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-56. FIFO0THRESHOLD Register Field Descriptions**

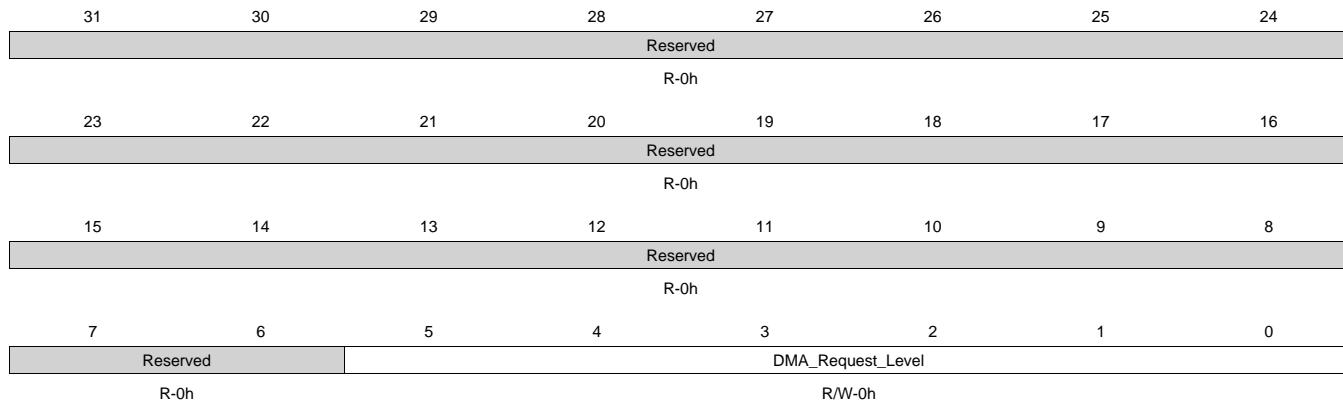
| Bit  | Field                 | Type | Reset | Description  |
|------|-----------------------|------|-------|--|
| 31-6 | Reserved              | R    | 0h    |  |
| 5-0  | FIFO0_threshold_Level | R/W  | 0h    | Program the desired FIFO0 data sample level to reach before generating interrupt to CPU (program to value minus 1) |

### 12.5.1.53 DMA0REQ Register (offset = ECh) [reset = 0h]

DMA0REQ is shown in [Figure 12-57](#) and described in [Table 12-57](#).

FIFO0 DMA req0 trigger@TSC\_ADC\_SS\_FIFO0 DMA REQUEST Register

**Figure 12-57. DMA0REQ Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-57. DMA0REQ Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-6 | Reserved          | R    | 0h    | RESERVED  |
| 5-0  | DMA_Request_Level | R/W  | 0h    | Number of words in FIFO0 before generating a DMA request (program to value minus 1) |

### 12.5.1.54 FIFO1COUNT Register (offset = F0h) [reset = 0h]

FIFO1COUNT is shown in [Figure 12-58](#) and described in [Table 12-58](#).

FIFO1 word count@TSC\_ADC\_SS\_FIFO1 Word Count Register

**Figure 12-58. FIFO1COUNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-58. FIFO1COUNT Register Field Descriptions**

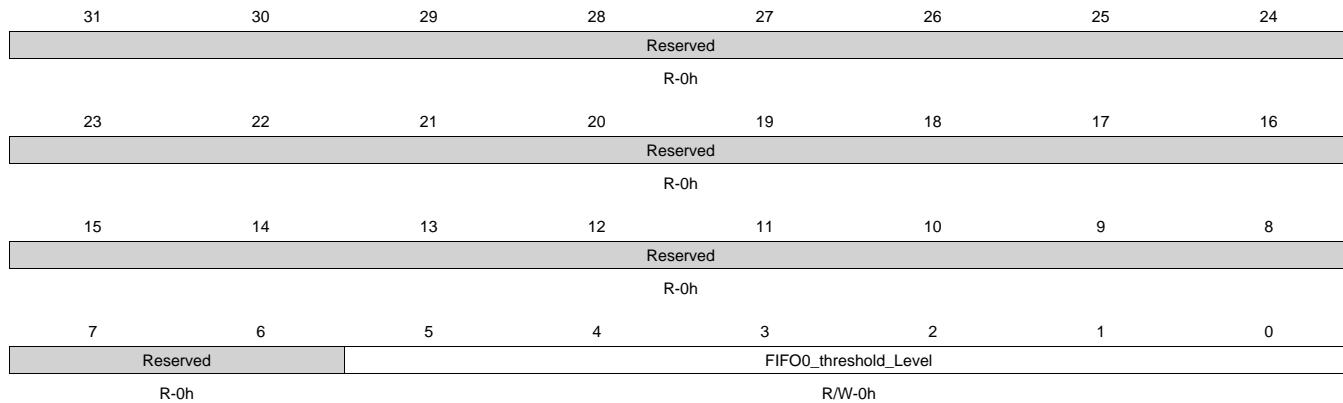
| Bit  | Field          | Type | Reset | Description                            |
|------|----------------|------|-------|--|
| 31-7 | Reserved       | R    | 0h    | RESERVED                               |
| 6-0  | Words_in_FIFO0 | R    | 0h    | Number of words currently in the FIFO0 |

### 12.5.1.55 FIFO1THRESHOLD Register (offset = F4h) [reset = 0h]

FIFO1THRESHOLD is shown in [Figure 12-59](#) and described in [Table 12-59](#).

FIFO1 Threshold trigger@TSC\_ADC\_SS\_FIFO1 Threshold Level Register

**Figure 12-59. FIFO1THRESHOLD Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-59. FIFO1THRESHOLD Register Field Descriptions**

| Bit  | Field                 | Type | Reset | Description  |
|------|-----------------------|------|-------|--|
| 31-6 | Reserved              | R    | 0h    |  |
| 5-0  | FIFO0_threshold_Level | R/W  | 0h    | Program the desired FIFO0 data sample level to reach before generating interrupt to CPU (program to value minus 1) |

### 12.5.1.56 DMA1REQ Register (offset = F8h) [reset = 0h]

DMA1REQ is shown in [Figure 12-60](#) and described in [Table 12-60](#).

FIFO1 DMA req1 trigger@TSC\_ADC\_SS\_FIFO1 DMA Request Register

**Figure 12-60. DMA1REQ Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-60. DMA1REQ Register Field Descriptions**

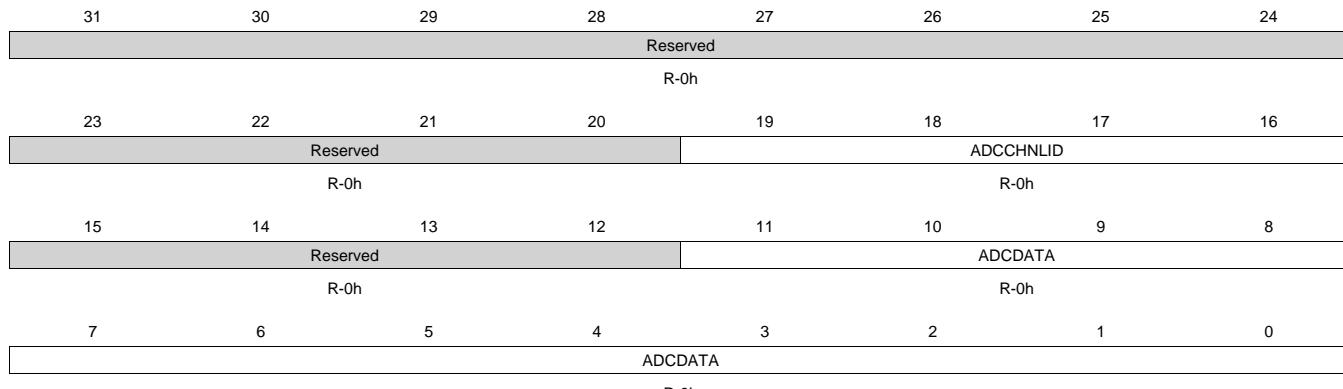
| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-6 | Reserved          | R    | 0h    | RESERVED  |
| 5-0  | DMA_Request_Level | R/W  | 0h    | Number of words in FIFO0 before generating a DMA request (program to value minus 1) |

### 12.5.1.57 FIFO0DATA Register (offset = 100h) [reset = 0h]

FIFO0DATA is shown in [Figure 12-61](#) and described in [Table 12-61](#).

ADC\_FIFO0 \_READ Data @TSC\_ADC\_SS\_FIFO0 READ Register

**Figure 12-61. FIFO0DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-61. FIFO0DATA Register Field Descriptions**

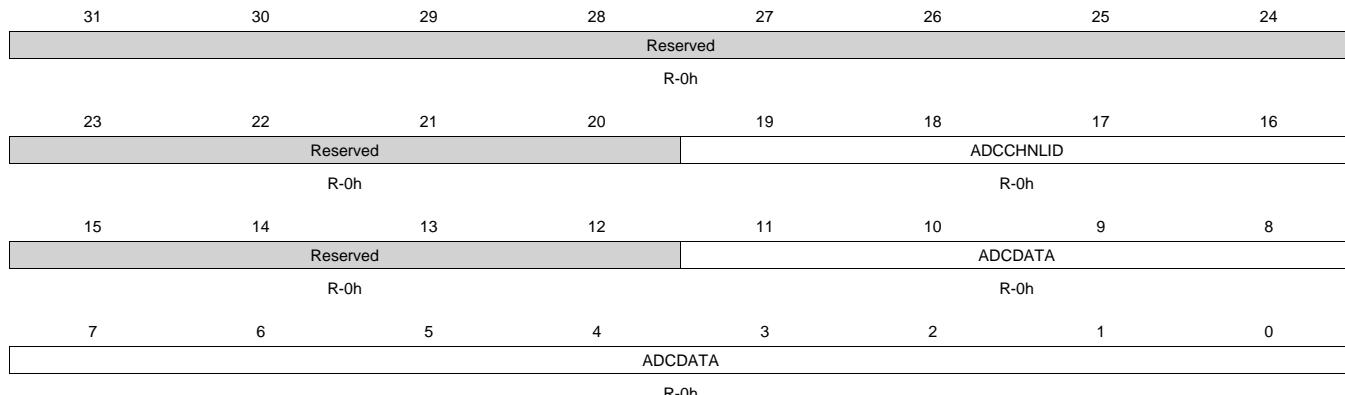
| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-20 | Reserved  | R    | 0h    | RESERVED.  |
| 19-16 | ADCCHNLID | R    | 0h    | Optional ID tag of channel that captured the data.<br>If tag option is disabled, these bits will be 0. |
| 15-12 | Reserved  | R    | 0h    |  |
| 11-0  | ADCDATA   | R    | 0h    | 12 bit sampled ADC converted data value stored in FIFO 0.  |

### 12.5.1.58 FIFO1DATA Register (offset = 200h) [reset = 0h]

FIFO1DATA is shown in [Figure 12-62](#) and described in [Table 12-62](#).

ADC FIFO1\_READ Data@TSC\_ADC\_SS\_FIFO1 READ Register

**Figure 12-62. FIFO1DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 12-62. FIFO1DATA Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-20 | Reserved  | R    | 0h    | RESERVED   |
| 19-16 | ADCCHNLID | R    | 0h    | Optional ID tag of channel that captured the data.<br>If tag option is disabled, these bits will be 0. |
| 15-12 | Reserved  | R    | 0h    | RESERVED   |
| 11-0  | ADCDATA   | R    | 0h    | 12 bit sampled ADC converted data value stored in FIFO 1.  |

## LCD Controller

This chapter describes the LCD controller of the device.

| Topic                             | Page |
|-----------------------------------|------|
| 13.1 Introduction .....           | 1090 |
| 13.2 Integration .....            | 1092 |
| 13.3 Functional Description ..... | 1094 |
| 13.4 Programming Model .....      | 1114 |
| 13.5 LCD Registers .....          | 1120 |

## 13.1 Introduction

### 13.1.1 Purpose of the Peripheral

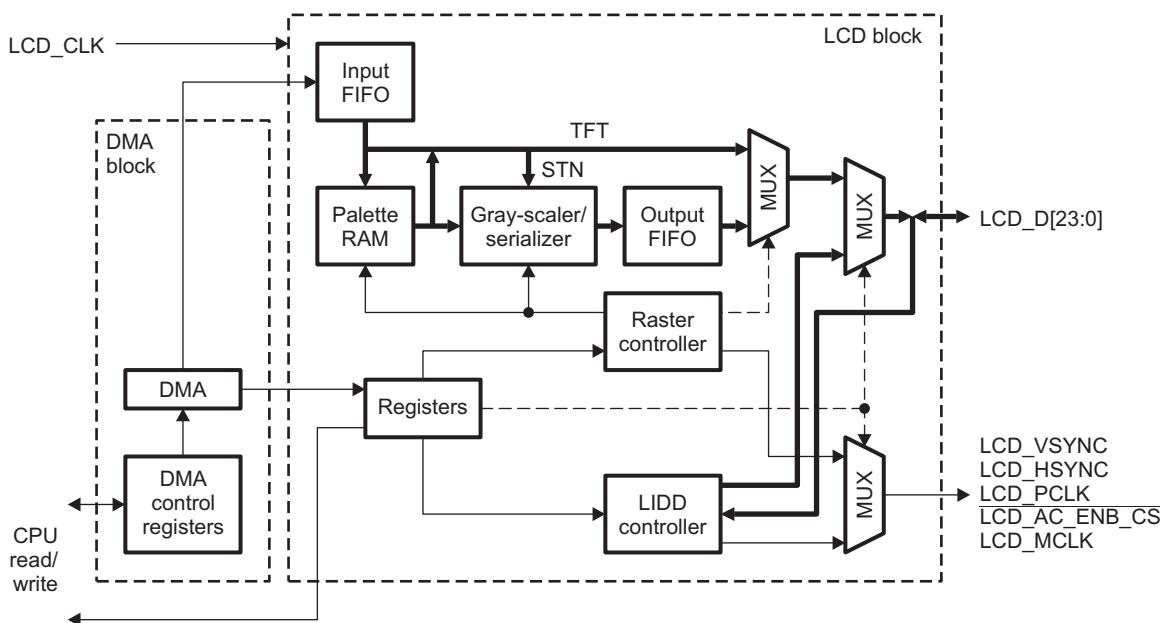
The LCD controller consists of two independent controllers, the Raster Controller and the LCD Interface Display Driver (LIDD) controller. Each controller operates independently from the other and only one of them is active at any given time.

- The Raster Controller handles the synchronous LCD interface. It provides timing and data for constant graphics refresh to a passive display. It supports a wide variety of monochrome and full-color display types and sizes by use of programmable timing controls, a built-in palette, and a gray-scale/serializer. Graphics data is processed and stored in frame buffers. A frame buffer is a contiguous memory block in the system. A built-in DMA engine supplies the graphics data to the Raster engine which, in turn, outputs to the external LCD device.
- The LIDD Controller supports the asynchronous LCD interface. It provides full-timing programmability of control signals (CS, WE, OE, ALE) and output data.

**Figure 13-1** shows the LCD controller details. The raster and LIDD Controllers are responsible for generating the correct external timing. The DMA engine provides a constant flow of data from the frame buffer(s) to the external LCD panel via the Raster and LIDD Controllers. In addition, CPU access is provided to read and write registers.

The solid, thick lines in **Figure 13-1** indicate the data path. The Raster Controller's data path is fairly complicated, for a thorough description of the Raster Controller data path, see [Section 13.3.5](#).

**Figure 13-1. LCD Controller**



### 13.1.2 Features

General features of the LCD Controller include:

- Supports up to 24-bit data output; 8 bits-per-pixel (RGB).
- Supports up to WXGA (1366x768) resolution.
- Integrated DMA engine to pull data from the external frame buffer without burdening the processor via interrupts or a firmware timer.
- 512 word deep internal FIFO with programmable threshold values.
- Character Based Panels
  - Supports 2 Character Panels (CS0 and CS1) with independent and programmable bus timing parameters when in asynchronous Hitachi, Motorola and Intel modes
  - Supports 1 Character Panel (CS0) with programmable bus timing parameters when in synchronous Motorola and Intel modes
  - Can be used as a generic 16 bit address/data interleaved MPU bus master with no external stall
- Passive Matrix LCD Panels
  - Panel types including STN, DSTN, and C-DSTN
  - AC Bias Control
- Active Matrix LCD Panels
  - Panel types including TN TFT
- OLED Panels
  - Passive Matrix (PM OLED) with frame buffer and controller IC inside the Panel
  - Active Matrix (AM OLED)

## 13.2 Integration

The device includes an LCD Controller that reads display data from external memory and drives several different types of LCD displays. The LCD Controller integration is shown in [Figure 13-2](#).

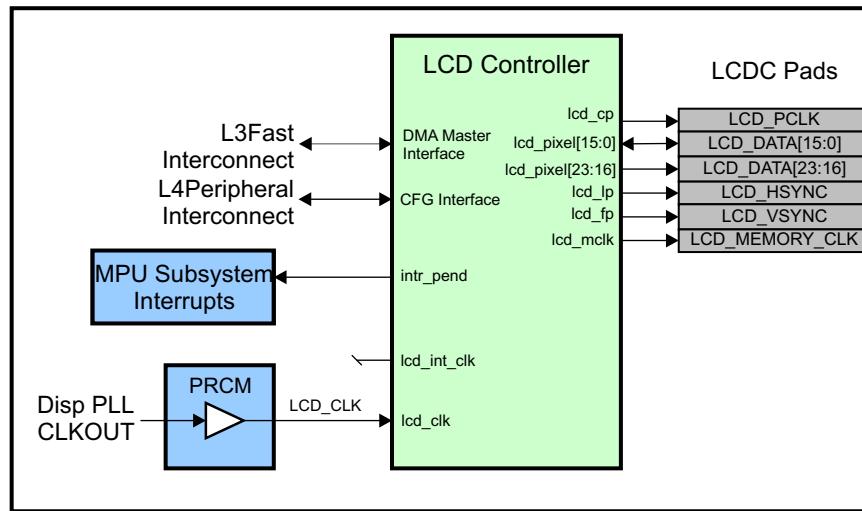


Figure 13-2. LCD Controller Integration

### 13.2.1 LCD Controller Connectivity Attributes

The general connectivity attributes for the LCDC subsystems are shown in [Table 13-1](#).

Table 13-1. LCD Controller Connectivity Attributes

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain   |
| Clock Domain        | PD_PER_LCD_L3_GCLK (OCP Master Clock)<br>PD_PER_LCD_L3_GCLK (OCP Slave Clock)<br>PD_PER_LCD_GCLK (Functional Clock) |
| Reset Signals       | PER_DOM_RST_N   |
| Idle/Wakeup Signals | Standby Smart Idle  |
| Interrupt Requests  | 1 Interrupt to MPU Subsystem (LCDCINT)  |
| DMA Requests        | None  |
| Physical Address    | L4 Peripheral Slave Port  |

### 13.2.2 LCD Controller Clock and Reset Management

The LCDC module uses the following functional and OCP interface clocks. The L4 Slave interface runs at half the frequency of the L3 Master interface but uses the same clock. The clock is divided within the LCDC through the I4\_clkdiv input using the pd\_per\_lcd\_I4s\_gclk\_iens signal from the PRCM. The functional clock comes from the Display PLL. When the Display PLL is in bypass mode, its output is sourced by either CORE\_CLKOUTM6 or PER\_CLKOUTM2.

**Table 13-2. LCD Controller Clock Signals**

| Clock Signal                     | Max Freq | Reference / Source                     | Comments                        |
|----------------------------------|----------|--|---------------------------------|
| I3_clk<br>Master Interface Clock | 200 MHz  | CORE_CLKOUTM4                          | pd_per_lcd_I3_gclk<br>From PRCM |
| I4_clk<br>Slave Interface Clock  | 100 MHz  | CORE_CLKOUTM4<br>(divided within LCDC) | pd_per_lcd_I3_gclk<br>From PRCM |
| Lcd_clk<br>Functional Clock      | 200 MHz  | Display PLL CLKOUT                     | pd_per_lcd_gclk<br>From PRCM    |

### 13.2.3 LCD Controller Pin List

The LCD Controller external interface signals are shown in [Table 13-3](#).

**Table 13-3. LCD Controller Pin List**

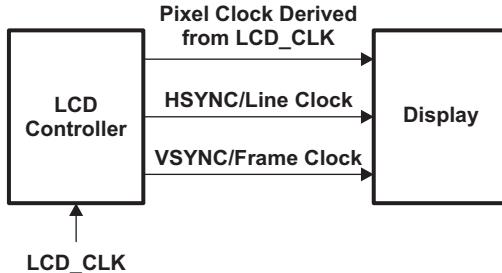
| Pin               | Type | Description  |
|-------------------|------|--|
| lcd_cp            | O    | Pixel Clock in Raster mode Read Strobe or Read/Write Strobe in LIDD mode   |
| lcd_pixel_i[15:0] | I    | LCD Data Bus input (for LIDD mode only)  |
| lcd_pixel_o[23:0] | O    | LCD Data Bus output  |
| lcd_lp            | O    | Line Clock or HSYNC in Raster mode; Write Strobe or Direction bit in LIDD mode                                     |
| lcd_fp            | O    | Frame Clock or VSYNC in Raster mode; Address Latch Enable in LIDD mode   |
| lcd_ac            | O    | AC bias or Latch Enable in Raster mode; Primary Chip Select/Primary Enable in LIDD MPU/Hitachi mode                |
| lcd_mclk          | O    | N/A in Raster mode; Memory Clock/Secondary chip Select/Secondary Enable in LIDD Synchronous/Async MPU/Hitachi mode |

## 13.3 Functional Description

### 13.3.1 Clocking

This section details the various clocks and signals. [Figure 13-3](#) shows input and output LCD controller clocks.

**Figure 13-3. Input and Output Clocks**



#### 13.3.1.1 Pixel Clock (LCD\_PCLK)

The pixel clock (LCD\_PCLK) frequency is derived from LCD\_CLK, the reference clock to this LCD module (see [Figure 13-3](#)). The pixel clock is used by the LCD display to clock the pixel data into the line shift register.

$$\text{LCD\_PCLK} = \frac{\text{LCD\_CLK}}{\text{CLKDIV}}$$

where CLKDIV is a field in the LCD\_CTRL register and should not be 0 or 1.

- **Passive (STN) mode.** LCD\_PCLK only transitions when valid data is available for output. It does not transition when the horizontal clock (HSYNC) is asserted or during wait state insertion.
- **Active (TFT) mode.** LCD\_PCLK continuously toggles as long as the Raster Controller is enabled.

### 13.3.1.2 Horizontal Clock (LCD\_HSYNC)

LCD\_HSYNC toggles after all pixels in a horizontal line have been transmitted to the LCD and a programmable number of pixel clock wait states has elapsed both at the beginning and end of each line.

The RASTER\_TIMING\_0 register fully defines the behavior of this signal.

LCD\_HSYNC can be programmed to be synchronized with the rising or falling edge of LCD\_PCLK. The configuration field is bits 24 and 25 in the RASTER\_TIMING\_2 register.

**Active (TFT) mode:** The horizontal clock or the line clock is also used by TFT displays as the horizontal synchronization signal (LCD\_HSYNC).

The timings of the horizontal clock (line clock) pins are programmable to support:

- Delay insertion both at the beginning and end of each line.
- Line clock polarity.
- Line clock pulse width, driven on rising or falling edge of pixel clock.

### 13.3.1.3 Vertical Clock (LCD\_VSYNC)

LCD\_VSYNC toggles after all lines in a frame have been transmitted to the LCD and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame.

The RASTER\_TIMING\_1 register fully defines the behavior of this signal.

LCD\_VSYNC can be programmed to be synchronized with the rising or falling edge of LCD\_PCLK. The configuration field is bits 24 and 25 in the RASTER\_TIMING\_2 register.

- **Passive (STN) mode.** The vertical, or frame, clock toggles during the first line of the screen.
- **Active (TFT) mode.** The vertical, or frame, clock is also used by TFT displays as the vertical synchronization signal (LCD\_VSYNC).

The timings of the vertical clock pins are programmable to support:

- Delay insertion both at the beginning and end of each frame
- Frame clock polarity

### 13.3.1.4 LCD\_AC\_BIAS\_EN

- **Passive (STN) mode.** To prevent a dc charge within the screen pixels, the power and ground supplies of the display are periodically switched. The Raster Controller signals the LCD to switch the polarity by toggling this pin (LCD\_AC\_BIAS\_EN).
- **Active (TFT) mode.** This signal acts as an output enable (OE) signal. It is used to signal the external LCD that the data is valid on the data bus (LCD\_DATA).

### 13.3.2 LCD External I/O Signals

Table 13-4 shows the details of the LCD controller external signals.

**Table 13-4. LCD External I/O Signals**

| Signal         | Type        | Description   |
|----------------|-------------|---|
| LCD_VSYNC      | OUT         | Raster controller: Frame clock the LCD uses to signal the start of a new frame of pixels.<br>Also used by TFT displays as the vertical synchronization signal.<br><br>LIDD character: Register select (RS) or address latch enable (ALE)<br><br>LIDD graphics: Address bit 0 (A0) or command/data select (C/D)  |
| LCD_HSYNC      | OUT         | Raster controller: Line clock the LCD uses to signal the end of a line of pixels that transfers line data from the shift register to the screen and to increment the line pointer(s). Also used by TFT displays as the horizontal synchronization signal.<br><br>LIDD character: not used.<br><br>LIDD graphics: <ul style="list-style-type: none"><li>• 6800 mode = read or write enable</li><li>• 8080 mode = not write strobe</li></ul>  |
| LCD_PCLK       | OUT         | Raster controller: Pixel clock the LCD uses to clock the pixel data into the line shift register.<br>In passive mode, the pixel clock transitions only when valid data is available on the data lines. In active mode, the pixel clock transitions continuously, and the ac-bias pin is used as an output enable to signal when data is available on the LCD pin.<br><br>LIDD character: not used.<br><br>LIDD graphics: <ul style="list-style-type: none"><li>• 6800 mode = enable strobe</li><li>• 8080 mode = not read strobe</li></ul>  |
| LCD_AC_BIAS_EN | OUT         | Raster controller: ac-bias used to signal the LCD to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. Used in TFT mode as the output enable to signal when data is latched from the data pins using the pixel clock.<br><br>LIDD character: Primary enable strobe<br><br>LIDD graphics: Chip select 0 (CS0)  |
| LCD_MCLK       | OUT         | Raster controller: not used.<br><br>LIDD character: Secondary enable strobe<br><br>LIDD graphics: Chip select 1 (CS1)   |
| LCD_D[23:0]    | Raster: OUT | LCD data bus, providing a 4-, 8-, 16- or 24-bit data path.<br><br>Raster controller: For monochrome displays, each signal represents a pixel; for passive color displays, groupings of three signals represent one pixel (red, green, and blue). LCD_DATA[3:0] is used for monochrome displays of 2, 4, and 8 BPP; LCD_DATA[7:0] is used for color STN displays and LCD_DATA[15:0] or LCD_DATA[23:0] is used for active (TFT) mode.<br><br>LIDD: OUT/IN LIDD character: LCD_DATA[15:0] Read and write the command and data registers.<br>LIDD graphics: LCD_DATA[15:0] Read and write the command and data registers. |

### 13.3.3 DMA Engine

The DMA engine provides the capability to output graphics data to constantly refresh LCDs, without burdening the CPU, via interrupts or a firmware timer. It operates on one or two frame buffers, which are set up during initialization. Using two frame buffers (ping-pong buffers) enables the simultaneous operation of outputting the current video frame to the external display and updating the next video frame. The ping-pong buffering approach is preferred in most applications.

When the Raster Controller is used, the DMA engine reads data from a frame buffer and writes it to the input FIFO (as shown in [Figure 13-1](#)). The Raster Controller requests data from the FIFO for frame refresh; as a result, the DMA's job is to ensure that the FIFO is always kept full.

When the LIDD Controller is used, the DMA engine accesses the LIDD Controller's address and/or data registers.

To program the DMA engine, configure the following registers, as shown in [Table 13-5](#).

**Table 13-5. Register Configuration for DMA Engine Programming**

| Register           | Configuration   |
|--------------------|---|
| LCDDMA_CTRL        | Configure DMA data format   |
| LCDDMA_FB0_BASE    | Configure frame buffer 0  |
| LCDDMA_FB0_CEILING |   |
| LCDDMA_FB1_BASE    | Configure frame buffer 1. (If only one frame buffer is used, these two registers will not be used.) |
| LCDDMA_FB1_CEILING |   |

In addition, the LIDD\_CTRL register (for LIDD Controller) or the RASTER\_CTRL register (for Raster Controller) should also be configured appropriately, along with all the timing registers.

To enable DMA transfers, the LIDD\_DMA\_EN bit (in the LIDD\_CTRL register) or the LCDEN bit (in the RASTER\_CTRL register) should be written with 1.

#### 13.3.3.1 Interrupts

Interrupts in this LCD module are related to DMA engine operation. Three registers are used to control and monitor the interrupts:

- The LIDD\_CTRL and RASTER\_CTRL registers enable or disable each individual interrupt sources.
- The IRQSTATUS\_RAW register collects all the interrupt status information.

##### 13.3.3.1.1 LIDD Mode

When operating in LIDD mode, the DMA engine generates one interrupt signal every time the specified frame buffer has been transferred completely.

- The DONE\_INT\_EN bit in the LIDD\_CTRL register specifies if the interrupt signal is delivered to the system interrupt controller, which in turn may or may not generate an interrupt to CPU.
- The EOF1, EOF0, and DONE bits in the IRQSTATUS\_RAW register reflect the interrupt signal, regardless of being delivered to the system interrupt controller or not.

##### 13.3.3.1.2 Raster Mode

When operating in Raster mode, the DMA engine can generate the interrupts in the following scenarios:

1. **Output FIFO under-run.** This occurs when the DMA engine cannot keep up with the data rate consumed by the LCD (which is determined by the LCD\_PCLK.) This is likely due to a system memory throughput issue or an incorrect LCD\_PCLK setting. The FUF bit in IRQSTATUS\_RAW is set when this error occurs. This bit is cleared by disabling the Raster Controller (i.e., clearing the LCDEN bit in RASTER\_CTRL).

2. **Frame synchronization lost.** This error happens when the DMA engine attempts to read what it believes to be the first word of the video buffer but it cannot be recognized as such. This could be caused by an invalid frame buffer address or an invalid BPP value (for more details, see [Section 13.3.5.2](#)). The SYNC bit in the IRQSTATUS\_RAW register is set when such an error is detected. This field is cleared by disabling the Raster Controller (clearing the LCDEN bit in the RASTER\_CTRL register).
3. **Palette loaded.** This interrupt can be generated when the palette is loaded into the memory by the DMA engine. At the same time, the PL bit in the IRQSTATUS\_RAW register is set. In data-only (PLM = 2h) and palette-plus-data (PLM = 00) modes, writing 0 to this bit clears the interrupt. In the palette-only (PLM = 1) mode, this bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in the RASTER\_CTRL register).
4. **AC bias transition.** If the ACB\_I bit in the RASTER\_TIMING\_2 register is programmed with a non-zero value, an internal counter will be loaded with this value and starts to decrement each time LCD\_AC\_BIAS\_EN (AC-bias signal) switches its state. When the counter reaches zero, the ACB bit in the IRQSTATUS\_RAW register is set, which will deliver an interrupt signal to the system interrupt controller (if the interrupt is enabled.) The counter reloads the value in field ACB\_I, but does not start to decrement until the ABC bit is cleared by writing 0 to this bit.
5. **Frame transfer completed.** When one frame of data is transferred completely, the DONE bit in the IRQSTATUS\_RAW register is set. This bit is cleared by disabling the Raster Controller (i.e., clearing the LCDEN bit in the RASTER\_CTRL register). Note that the EOF0 and EOF1 bits in the IRQSTATUS\_RAW register will be set accordingly.

Note that the interrupt enable bits are in the RASTER\_CTRL register. The corresponding enable bit must be set in order to generate an interrupt to the CPU. However, the IRQSTATUS\_RAW register reflects the interrupt signal regardless of the interrupt enable bits settings.

#### 13.3.3.1.3 Interrupt Handling

See [Chapter 6, Interrupts](#), for information about LCD interrupt number to CPU. The interrupt service routine needs to determine the interrupt source by examining the IRQSTATUS\_RAW register and clearing the interrupt properly.

#### 13.3.4 LIDD Controller

The LIDD Controller is designed to support LCD panels with a memory-mapped interface. The types of displays range from low-end character monochrome LCD panels to high-end TFT smart LCD panels.

LIDD mode (and the use of this logic) is enabled by clearing the MODESEL bit in the LCD control register (LCD\_CTRL).

LIDD Controller operation is summarized as follows:

- During initialization, the LCD LIDD CS0/CS1 configuration registers (LIDD\_CS0\_CONF and LIDD\_CS1\_CONF) are configured to match the requirements of the LCD panel being used.
- During normal operation, the CPU writes display data to the LCD data registers (LIDD\_CS0\_DATA and LIDD\_CS1\_DATA). The LIDD interface converts the CPU write into the proper signal transition sequence for the display, as programmed earlier. Note that the first CPU write should send the beginning address of the update to the LCD panel and the subsequent writes update data at display locations starting from the first address and continuing sequentially. Note that DMA may be used instead of CPU.
- The LIDD Controller is also capable of reading back status or data from the LCD panel, if the latter has this capability. This is set up and activated in a similar manner to the write function described above.

---

**NOTE:** If an LCD panel is not used, this interface can be used to control any MCU-like peripheral.

---

See your device-specific data manual to check the LIDD features supported by the LCD controller.

[Table 13-6](#) describes how the signals are used to interface external LCD modules, which are configured by the LIDD\_CTRL register.

**Table 13-6. LIDD I/O Name Map**

| Display Type                    | Interface Type | Data Bits | LIDD_CTRL [2:0] | I/O Name   | Display I/O Name                                      | Comment  |
|---------------------------------|----------------|-----------|-----------------|--|---|--|
| Character Display               | HD44780 Type   | 4         | 100             | LCD_DATA[7:4]<br><br>LCD_AC_BIAS_EN<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_MCLK              | DATA[7:4]<br><br>E (or E0)<br>R/W<br>RS<br>E1         | Data Bus (length defined by Instruction)<br><br>Enable Strobe (first display)<br>Read/Write<br>Register Select (Data/not Instruction)<br>Enable Strobe (second display optional) |
| Character Display               | HD44780 Type   | 8         | 100             | LCD_DATA[7:0]<br><br>LCD_AC_BIAS_EN<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_MCLK              | DATA[7:0]<br><br>E (or E0)<br>R/W<br>RS<br>E1         | Data Bus (length defined by Instruction)<br><br>Enable Strobe (first display)<br>Read/Write<br>Register Select (Data/not Instruction)<br>Enable Strobe (second display optional) |
| Micro Interface Graphic Display | 6800 Family    | Up to 16  | 001             | LCD_DATA[15:0]<br><br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK | DATA[7:0]<br><br>E<br>R/W<br>A0<br>CS (or CS0)<br>CS1 | Data Bus (16 bits always available)<br><br>Enable Clock<br>Read/Write<br>Address/Data Select<br>Chip Select (first display)<br>Chip Select (second display optional)             |
|                                 |                |           | 000             | LCD_MCLK   | None  | Synchronous Clock (optional)   |
| Micro Interface Graphic Display | 8080 Family    | Up to 16  | 011             | LCD_DATA[15:0]<br><br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK | DATA[7:0]<br><br>RD<br>WR<br>A0<br>CS (or CS0)<br>CS1 | Data Bus (16 bits always available)<br><br>Read Strobe<br>Write Strobe<br>Address/Data Select<br>Chip Select (first display)<br>Chip Select (second display optional)            |
|                                 |                |           | 010             | LCD_MCLK   | None  | Synchronous Clock (optional)   |

The timing parameters are defined by the LIDD\_CS0\_CONF and LIDD\_CS1\_CONF registers, which are described in [Section 13.5.4](#) and [Section 13.5.7](#).

The timing configuration is based on an internal reference clock, MCLK. The MCLK is generated out of LCD\_CLK, which is determined by the CLKDIV bit in the LCD\_CTRL register.

$$\text{MCLK} = \text{LCD\_CLK} \text{ when } \text{CLKDIV} = 0.$$

$$\text{MCLK} = \frac{\text{LCD\_CLK}}{\text{CLKDIV}} \text{ when } \text{CLKDIV} \neq 0.$$

See your device-specific data manual for the timing configurations supported by the LCD controller.

### 13.3.5 Raster Controller

Raster mode (and the use of this logic) is enabled by setting the MODESEL bit in the LCD control register (LCD\_CTRL). [Table 13-7](#) shows the active external signals when this mode is active.

**Table 13-7. Operation Modes Supported by Raster Controller**

| Interface                | Data Bus Width | Register Bits<br>RASTER_CTRL[9, 7, 1] | Signal Name  | Description  |
|--------------------------|----------------|---------------------------------------|--|--|
| Passive (STN) Mono 4-bit | 4              | 001                                   | LCD_DATA[3:0]<br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK  | Data bus<br>Pixel clock<br>Horizontal clock(Line Clock)<br>Vertical clock (Frame Clock)<br>AC Bias<br>Not used       |
| Passive (STN) Mono 8-bit | 8              | 101                                   | LCD_DATA[7:0]<br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK  | Data bus<br>Pixel clock<br>Horizontal clock(Line Clock)<br>Vertical clock (Frame Clock)<br>AC Bias<br>Not used       |
| Passive (STN) Color      | 8              | 100                                   | LCD_DATA[7:0]<br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK  | Data bus<br>Pixel clock<br>Horizontal clock(Line Clock)<br>Vertical clock (Frame Clock)<br>AC Bias<br>Not used       |
| Active (TFT) Color       | 16             | x10                                   | LCD_DATA[15:0]<br>LCD_PCLK<br>LCD_HSYNC<br>LCD_VSYNC<br>LCD_AC_BIAS_EN<br>LCD_MCLK | Data bus<br>Pixel clock<br>Horizontal clock(Line Clock)<br>Vertical clock (Frame Clock)<br>Output enable<br>Not used |

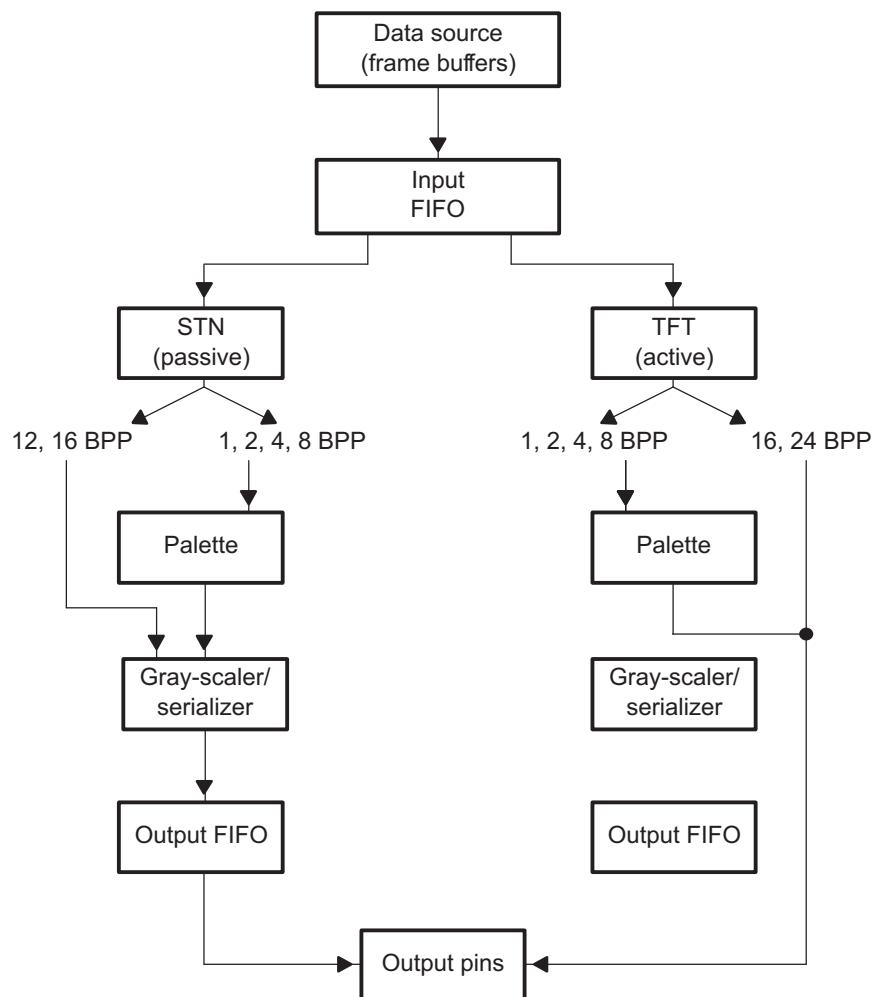
### 13.3.5.1 Logical Data Path

The block diagram of the Raster Controller is shown in [Figure 13-1](#). [Figure 13-4](#) illustrates its logical data path for various operation modes (passive (STN) versus active (TFT), various BPP size).

[Figure 13-4](#) shows that:

- The gray-scaler/serializer and output FIFO blocks are bypassed in active (TFT) modes.
- The palette is bypassed in both 12- and 16-BPP modes.

**Figure 13-4. Logical Data Path for Raster Controller**



In summary:

- The display image is stored in frame buffers.
- The built-in DMA engine constantly transfers the data stored in the frame buffers to the Input FIFO.
- The Raster Controller relays data to the external pins according to the specified format.

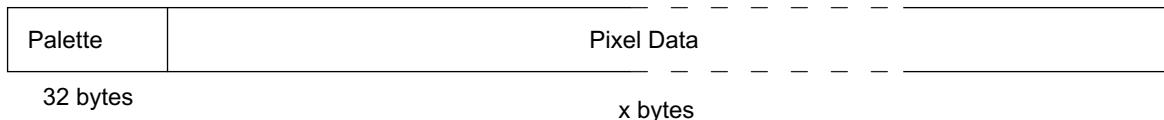
The remainder of this section describes the functioning blocks in [Figure 13-4](#), including frame buffers, palette, and gray-scaler(serializer). Their operation and programming techniques are covered in detail. The output format is also described in [Section 13.3.5.5](#).

### 13.3.5.2 Frame Buffer

A frame buffer is a contiguous memory block, storing enough data to fill a full LCD screen. For this device, external memory needs to be used for the frame buffer. For specific details on which external memory interface (EMIF) controller can be accessed by the LCD controller, see your device-specific data manual. The data in the frame buffer consists of pixel values as well as a look-up palette. [Figure 13-5](#) shows the frame buffer structure.

**Figure 13-5. Frame Buffer Structure**

#### 1, 2, 4, 12, 16, 24 BPP Modes



#### 8 BPP Mode



#### NOTE:

- 8-BPP mode uses the first 512 bytes in the frame buffer as the palette while the other modes use 32 bytes.
- 12-, 16-, and 24-BPP modes do not need a palette; i.e., the pixel data is the desired RGB value. However, the first 32 bytes are still considered a palette. The first entry should be 4000h (bit 14 is 1) while the remaining entries must be filled with 0. (For details, see [Table 13-8](#).)
- Each entry in a palette occupies 2 bytes. As a result, 8-BPP mode palette has 256 color entries while the other palettes have up to 16 color entries.
- 4-BPP mode uses up all the 16 entries in a palette.
- 1-BPP mode uses the first 2 entries in a palette while 2-BPP mode uses the first 4 entries. The remaining entries are not used and must be filled with 0.
- In 12- and 16-BPP modes, pixel data is RGB data. For all the other modes, pixel data is actually an index of the palette entry.

**Table 13-8. Bits-Per-Pixel Encoding for Palette Entry 0 Buffer**

| Bit   | Name | Value | Description  | (1) | (2) | (3) |
|-------|------|-------|--|-----|-----|-----|
| 14-12 | BPP  |       | Bits-per-pixel.  |     |     |     |
|       |      | 000   | 1 BPP  |     |     |     |
|       |      | 001   | 2 BPP  |     |     |     |
|       |      | 010   | 4 BPP  |     |     |     |
|       |      | 011   | 8 BPP  |     |     |     |
|       |      | 1xx   | 12 BPP in passive mode (TFT_STN = 0 and STN_565 = 0 in RASTER_CTRL)<br>16 BPP in passive mode (TFT_STN = 0 and STN_565 = 1 in RASTER_CTRL)<br>16 BPP in active mode (LCDTFT = 1 and TFT24 = 0 in RASTER_CTRL)<br>24 BPP in active mode (LCDTFT = 1 and TFT24 = 1 in RASTER_CTRL) |     |     |     |

(1) Eight 1-bit pixels, four 2-bit pixels, and two 4-bit pixels are packed into each byte, and 12-bit pixels are right justified on (16-bit) word boundaries (in the same format as palette entry).

(2) For STN565, see the 16 BPP STN mode bit ([Section 13.5.10](#)).

(3) For Raw Data (12/16/24 bpp) framebuffers, no Palette lookup is employed therefore PALMODE = 0x10 in RASTER\_CTRL.

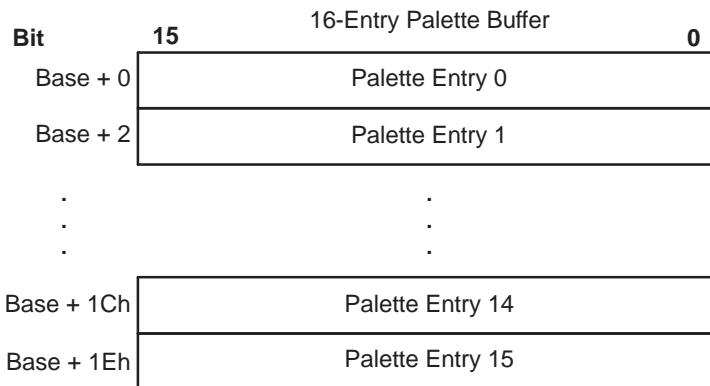
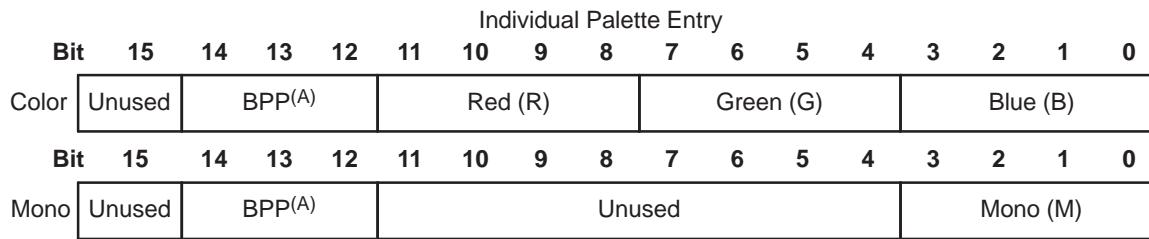
The equations shown in [Table 13-9](#) are used to calculate the total frame buffer size (in bytes) based on varying pixel size encoding and screen sizes.

[Figure 13-6](#) and [Figure 13-7](#) show more detail of the palette entry organization.

**Table 13-9. Frame Buffer Size According to BPP**

| BPP   | Frame Buffer Size                                    |
|-------|--|
| 1     | $32 + (\text{Lines} \times \text{Columns})/8$        |
| 2     | $32 + (\text{Lines} \times \text{Columns})/4$        |
| 4     | $32 + (\text{Lines} \times \text{Columns})/2$        |
| 8     | $512 + (\text{Lines} \times \text{Columns})$         |
| 12/16 | $32 + 2 \times (\text{Lines} \times \text{Columns})$ |

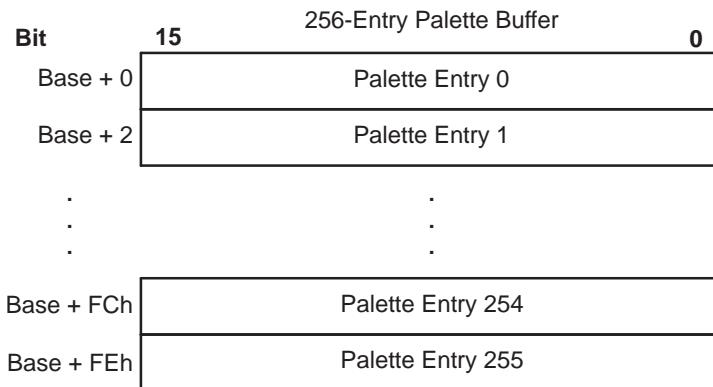
**Figure 13-6. 16-Entry Palette/Buffer Format (1, 2, 4, 12, 16 BPP)**



A. Bits-per-pixels (BPP) is only contained within the first palette entry (palette entry 0).

**Figure 13-7. 256-Entry Palette/Buffer Format (8 BPP)**

| Bit   | 15                       | 14                 | 13      | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3         | 2 | 1 | 0 |
|-------|--------------------------|--------------------|---------|----|----|----|---|---|---|---|---|---|-----------|---|---|---|
|       | Individual Palette Entry |                    |         |    |    |    |   |   |   |   |   |   |           |   |   |   |
| Color | Unused                   | BPP <sup>(A)</sup> | Red (R) |    |    |    |   |   |   |   |   |   | Green (G) |   |   |   |
| Mono  | Unused                   | BPP <sup>(A)</sup> | Unused  |    |    |    |   |   |   |   |   |   | Blue (B)  |   |   |   |



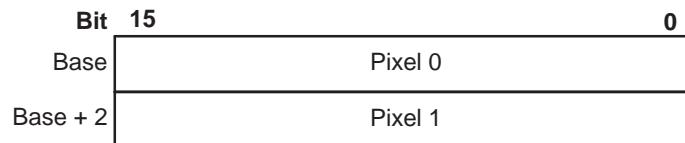
A. Bits-per-pixels (BPP) is only contained within the first palette entry (palette entry 0).

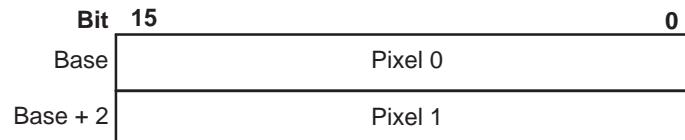
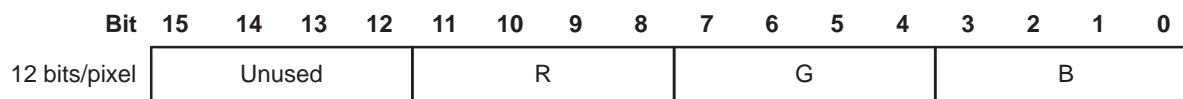
Bits 12, 13, and 14 of the first palette entry select the number of bits-per-pixel to be used in the following frame and thus the number of palette RAM entries. The palette entry is used by the Raster Controller to correctly unpack pixel data.

Figure 13-8 through Figure 13-13 show the memory organization within the frame buffer for each pixel encoding size.

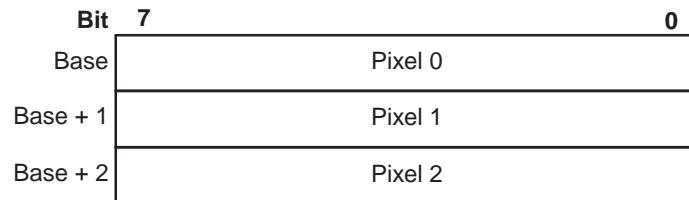
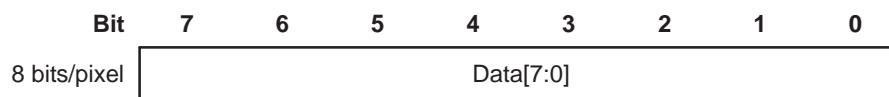
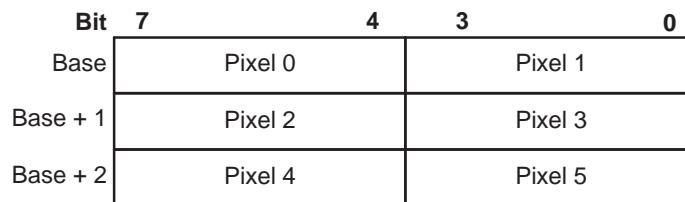
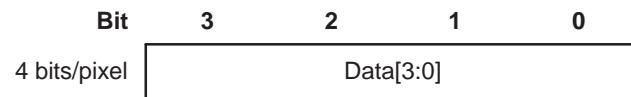
**Figure 13-8. 16-BPP Data Memory Organization (TFT Mode Only)—LittleEndian**

| Bit           | 15    | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------|-------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 16 bits/pixel | R G B |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |



**Figure 13-9. 12-BPP Data Memory Organization—Little Endian**


Unused [15-12] bits are filled with zeroes in TFT mode.

**Figure 13-10. 8-BPP Data Memory Organization**

**Figure 13-11. 4-BPP Data Memory Organization**


**Figure 13-12. 2-BPP Data Memory Organization**

| Bit      | 7       | 6       | 5       | 4       | 3       | 2        | 1       | 0        |
|----------|---------|---------|---------|---------|---------|----------|---------|----------|
| Base     | Pixel 0 |         | Pixel 1 |         | Pixel 2 |          | Pixel 3 |          |
| Base + 1 |         | Pixel 4 |         | Pixel 5 |         | Pixel 6  |         | Pixel 7  |
| Base + 2 |         | Pixel 8 |         | Pixel 9 |         | Pixel 10 |         | Pixel 11 |

**Figure 13-13. 1-BPP Data Memory Organization**

| Bit      | 7  | 6  | 5   | 4   | 3   | 2   | 1   | 0   |
|----------|----|----|-----|-----|-----|-----|-----|-----|
| Base     | P0 | P1 | P2  | P3  | P4  | P5  | P6  | P7  |
| Base + 1 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 |

### 13.3.5.3 Palette

As explained in the previous section, the pixel data is an index of palette entry (when palette is used). The number of colors supported is given by  $2^{\text{number of BPP}}$ . However, due to a limitation of the gray-scaler/serializer block, fewer grayscales or colors may be supported.

The PLM field (in RASTER\_CTRL) affects the palette loading:

- If PLM is 00b (palette-plus-data mode) or 01b (palette-only mode), the palette is loaded by the DMA engine at the very beginning, which is followed by the loading of pixel data.
- If PLM is 10b (data-only mode), the palette is not loaded. Instead, the DMA engine loads the pixel data immediately.

### 13.3.5.4 Gray-Scaler/Serializer

#### 13.3.5.4.1 Passive (STN) Mode

Once a palette entry is selected from the look-up palette by the pixel data, its content is sent to the gray-scaler/serializer. If it is monochrome data, it is encoded as 4 bits. If it is color data, it is encoded as 4 bits (Red), 4 bits (Green), and 4 bits (Blue).

These 4-bit values are used to select one of the 16 intensity levels, as shown in [Table 13-10](#). A patented algorithm is used during this processing to provide an optimized intensity value that matches the eye's visual perception of color/gray gradations.

#### 13.3.5.4.2 Active (TFT) Mode

The gray-scaler(serializer is bypassed.

**Table 13-10. Color/Grayscale Intensities and Modulation Rates**

| Dither Value<br>(4-Bit Value from Palette) | Intensity<br>(0% is White) | Modulation Rate<br>(Ratio of ON to ON+OFF Pixels) |
|--|----------------------------|---|
| 0000                                       | 0.0%                       | 0   |
| 0001                                       | 14.3%                      | 1/7   |
| 0010                                       | 20.0%                      | 1/5   |
| 0011                                       | 25%                        | 1/4   |
| 0100                                       | 33.3%                      | 3/9   |
| 0101                                       | 40.0%                      | 2/5   |
| 0110                                       | 44.4%                      | 4/9   |
| 0111                                       | 50.0%                      | 1/2   |
| 1000                                       | 55.6%                      | 5/9   |
| 1001                                       | 60.0%                      | 3/5   |
| 1010                                       | 66.6%                      | 6/9   |
| 1011                                       | 75%                        | 3/4   |
| 1100                                       | 80.0%                      | 4/5   |
| 1101                                       | 85.7%                      | 6/7   |
| 1110                                       | 93.3%                      | 14/15   |
| 1111                                       | 100.0%                     | 1   |

### 13.3.5.4.3 Summary of Color Depth

**Table 13-11. Number of Colors/Shades of Gray Available on Screen**

| Number of BPP | Passive Mode (LCDTFT = 0)  |  | Active Mode (LCDTFT = 1)                                  |
|---------------|--|--|---|
|               | Monochrome (LCDBW = 1)   | Color (LCDBW = 0)  | Color Only (LCDBW = 0)                                    |
| 1             | 2 palette entries to select within 15 grayscales   | 2 palette entries to select within 3375 possible colors  | 2 palette entries to select within 4096 possible colors   |
| 2             | 4 palette entries to select within 15 grayscales   | 4 palette entries to select within 3375 possible colors  | 4 palette entries to select within 4096 possible colors   |
| 4             | 16 palette entries to select within 15 grayscales  | 16 palette entries to select within 3375 possible colors | 16 palette entries to select within 4096 possible colors  |
| 8             | Not relevant since it would consist in 256 palette entries to select within 15 grayscales, but exists anyway | 256 palette entries to select 3375 possible colors       | 256 palette entries to select within 4096 possible colors |
| 12            | x  | 3375 possible colors                                     | 4096 possible colors                                      |
| 16            | x  | 3375 possible colors (STN_565 = 1)                       | Up to 65536 possible colors                               |
| 24            | X  | X  | Up to 16.7 million colors                                 |

### 13.3.5.5 Output Format

#### 13.3.5.5.1 Passive (STN) Mode

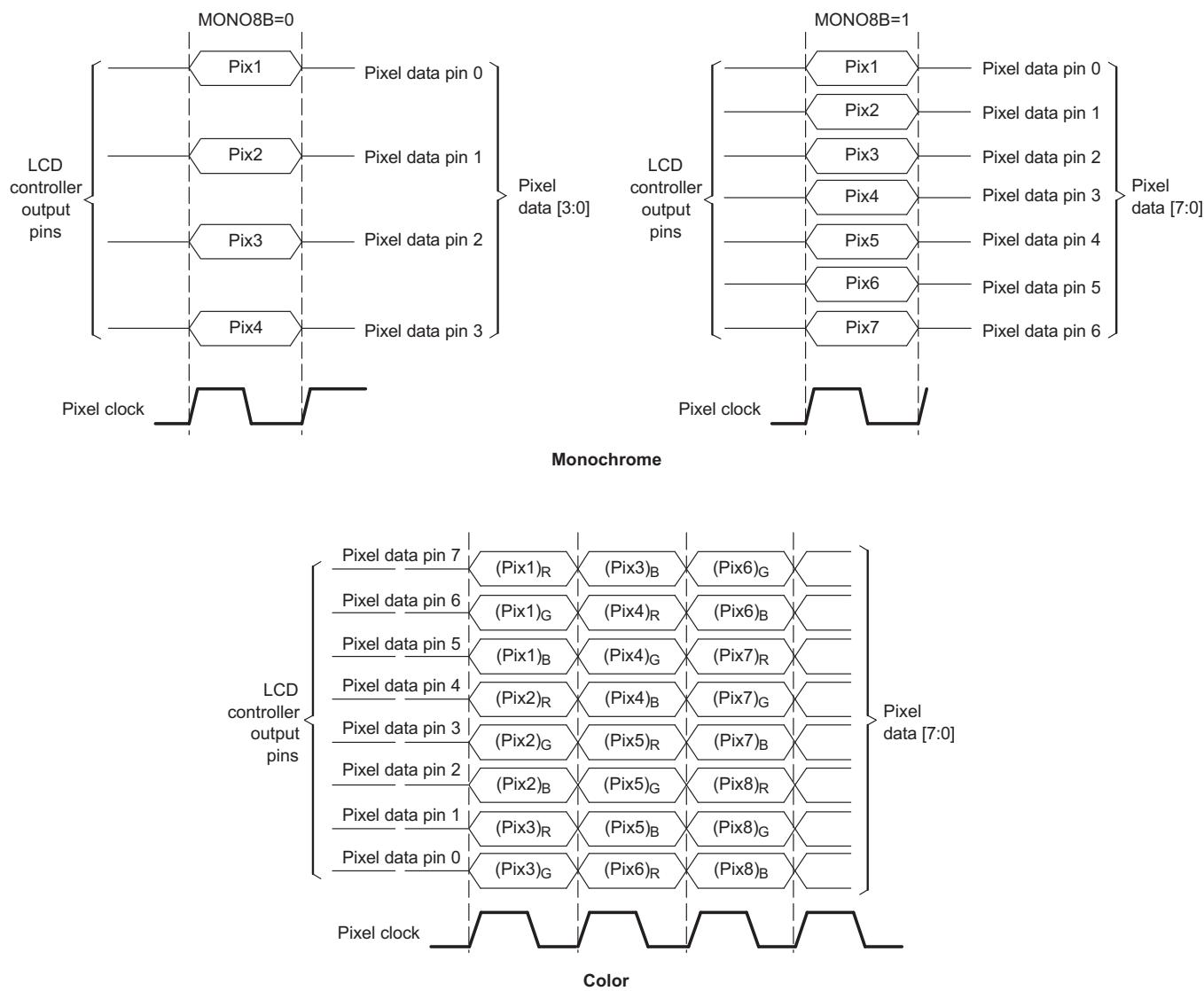
As shown in [Figure 13-4](#), the pixel data stored in frame buffers go through palette (if applicable) and gray-scaler/serializer before reaching the Output FIFO. As a result, it is likely that the data fed to the Output FIFO is numerically different from the data in the frame buffers. (However, they represent the same color or grayscale.)

The output FIFO formats the received data according to display modes (see [Table 13-7](#)). [Figure 13-14](#) shows the actual data output on the external pins.

#### 13.3.5.5.2 Active (TFT) Mode

As shown in [Figure 13-4](#), the gray-scaler/serializer and output FIFO are bypassed in active (TFT) mode. Namely, at each pixel clock, one pixel data (16 bits) is output to the external LCD.

**Figure 13-14. Monochrome and Color Output**



### 13.3.5.6 Subpicture Feature

A feature exists in the LCD to cover either the top or lower portion of the display with a default color. This feature is called a subpicture and is illustrated in [Figure 13-15](#). Subpictures are only allowed for Active Matrix mode (`cfg_lcdtft = '1'`).

Subpictures reduce the bandwidth to the DDR since lines containing default pixel data are not read from memory. For example, suppose the panel has 100 lines of which 50 are default pixel data lines. Then, only 50 lines of data are DMAed from DDR for this subpicture setup. That is, the `cfg_fbx_base` and `cfg_fbx_ceiling` registers only encompass 50 lines of data instead of 100.

**Figure 13-15. Example of Subpicture**



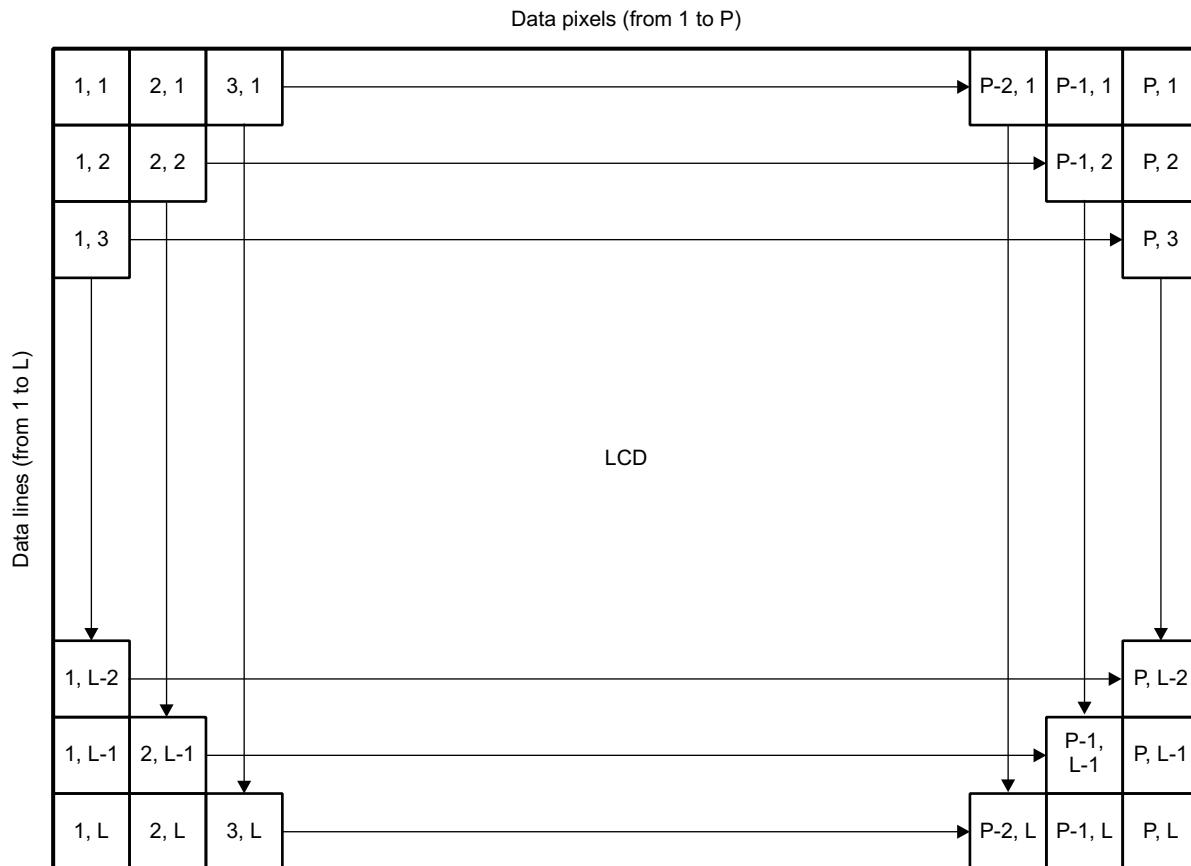
The subpicture feature is enabled when the `spen` MMR control bit is set to '1'. The `hols` bit, when set to '0,' puts the Default Pixel Data lines at the top of the screen and the active video lines at the bottom of the screen.

When the `hols` bit is set to '1,' Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen. The `hols` bit behavior is shown in [Figure 13-16](#).

**Figure 13-16. Subpicture HOLS Bit**



The lines per panel threshold (LPPT) bitfield defines the number of lines at the bottom of the picture for both `holz = '1'` or `'0'`. LPPT is an encoded value in the range {0:2047} used to represent the number of lines in the range {1:2048}.

**Figure 13-17. Raster Mode Display Format**


### **13.3.6 Interrupt Conditions**

#### **13.3.6.1 Highlander 0.8 Interrupts**

##### **13.3.6.1.1 Highlander Interrupt Basics**

The interrupt mechanism is Highlander 0.8-compliant and relies on the ipgvmmodirq IP Generic. The ipgvmmodirq module supports hardware-initiated interrupts, each of which can also be individually triggered by software. An interrupt mask function allows each interrupt to be masked or enabled. The software can read all of the raw interrupts or only those that are unmasked.

All pending interrupts in the LCD module must be serviced by the Host's Interrupt Service Routine before it exits.

The Interrupt Module registers are described in the following table.

**Table 13-12. Highlander 0.8 Interrupt Module Control Registers**

| Address Offset | Name  | Description                      |
|----------------|-------|----------------------------------|
| 0x58           | Reg22 | Interrupt Raw Status Register    |
| 0x5C           | Reg23 | Interrupt Masked Status Register |
| 0x60           | Reg24 | Interrupt Enable Set (Unmask)    |
| 0x64           | Reg25 | Interrupt Enable Clear (Mask)    |
| 0x68           | Reg26 | End of Interrupt Indicator       |

##### **13.3.6.1.2 Raw Status Register**

Interrupts are associated with a bit position. For instance, Hardware Interrupt 0 is physically connected to bit 0 of the interrupt controller and all Sets, Clears, and Masks to this interrupt will reference the Bit 0 location of the interrupt vector. Likewise, Hardware Interrupt 1 is referenced by bit 1 of the interrupt vector, and so on.

The Host CPU can see all the interrupts that have been set, regardless of the interrupt mask, by reading Reg22, the Raw Status Register.

If the Host CPU writes a '1' to a bit position in Reg 22, it will do a software set for the interrupt associated with that bit position.

##### **13.3.6.1.3 Masked Status Register**

The Masked Status Register contains all the pending interrupts that are unmasked (enabled). The Interrupt Service Routine should read this register to determine which interrupts must be serviced.

##### **13.3.6.1.4 Interrupt Enable Set Register**

To unmask an interrupt, the Host CPU writes a '1' to the appropriate bit position of the Enable Set (Unmask) register.

##### **13.3.6.1.5 Interrupt Enable Clear Register**

To mask an interrupt, the Host CPU writes a '1' to the appropriate bit position of the Enable Clear (Mask) register.

##### **13.3.6.1.6 End of Interrupt Register**

The ipgvmmodirq module supports level or pulse interrupts to the CPU. For pulse interrupts, the Host must write to an end-of-interrupt (EOI), memory-mapped address to indicate that the Interrupt Service Routine has completed and is exiting. Any pending interrupts that have not been serviced will trigger another interrupt pulse to the Host CPU.

### 13.3.6.2 Interrupt Sources

#### 13.3.6.2.1 Overview of Interrupt Sources

The interrupt sources include:

- DMA End of Frame 0
- DMA End of Frame 1
- Palette Loaded
- FIFO Underflow
- AC Bias Count
- Sync Lost
- Recurrent Frame Done
- LIDD or Raster Frame Done

##### 13.3.6.2.1.1 DMA End of Frame 0 and End of Frame 1 Interrupt

The DMA End of Frame 0 and End of Frame 1 interrupts are triggered when the DMA module has completed transferring the contents of a frame buffer bounded by cfg\_fb0\_base/cfg\_bf0\_ceil or cfg\_fb1\_base/cfg\_fb1\_ceil.

##### 13.3.6.2.1.2 Palette Loaded Interrupt

When cfg\_palmode is set to Palette-only or Palette+data, the Palette Loaded interrupt is triggered when the palette portion of the DMA transfer has been stored in the Palette RAM.

##### 13.3.6.2.1.3 FIFO Underflow Interrupt

The FIFO Underflow interrupt is triggered when the real-time output needs to send a value for pixel data but one cannot be found in the FIFO.

##### 13.3.6.2.1.4 AC Bias Count Interrupt

For Passive Matrix displays, a count can be kept of the number of times the AC Bias line toggles. Once the specified number of transitions has been seen, the AC Bias Count interrupt is triggered. The module will not post any further interrupts or keep counting AC Bias transitions until the interrupt has been cleared.

##### 13.3.6.2.1.5 Sync Lost Interrupt

When the DMA module reads a frame buffer and stores it in the FIFO, it sets a start frame and an end frame indicator embedded with the data. On retrieving the data from the FIFO in the lcd\_clk domain, the Sync Lost interrupt is triggered if the start indicator is not found at the first pixel of a new frame.

##### 13.3.6.2.1.6 Recurrent Frame Done Interrupt

In raster mode, the Recurrent Frame Done interrupt is triggered each time a complete frame has been sent to the interface pins.

##### 13.3.6.2.1.7 LIDD or Raster Frame Done Interrupt

In LIDD DMA mode, a frame buffer of data is sent. When the frame buffer has completed, the LIDD Frame Done interrupt is triggered. In order to do another LIDD DMA, the DMA engine must be disabled and then re-enabled.

In Raster mode, the interrupt is triggered after cfg\_lcden is set to '0' and after the last frame is sent to the pins. After the Raster mode DMA is running, the interrupt occurs only once after the module is disabled.

### 13.3.7 DMA

DDR access is handled internally by the DMA module. For Character Displays, the DMA module can transfer a sequence of data transactions from the DDR to LCD panel. By using the DMA instead of the Host CPU, the Host will not be stalled waiting for the slow external peripheral to complete.

For Passive and Active Matrix displays, the DMA is used to read frame buffers with associated palette information from DDR. The DMA parses the frame buffer according to the frame buffer type and supplies the raster processing chain with Palette information and pixel data as needed.

### 13.3.8 Power Management

Power management within the DSS can be accomplished in several ways:

1. L4 OCP MConnect/SConnect can disable the internal L4 clock network.
2. L3 OCP MConnect/Sconnect can disable the internal L3 clock network.
3. Within the Clock Control register, there are clock enable registers to disable the clock networks to all major internal functional paths.
4. Power Compiler clock gates are automatically instantiated within datapaths to minimize active power.

Items 1 and 2 are accomplished using the standard IDLE (for L4) and STANDBY (for L3) IPGeneric modules. When these modules are instructed to disable clocks for the internal L3 or L4 (MMR) clock domains, the internal clock networks will be shut down. This shutdown applies to the external clock pins I3\_clk and I4\_clk.

All other internal clock domains (Item 3) can only be shut down by writing the appropriate register bit within the Clock Enable register. This software clock control applies to all other clock inputs.

Power Compiler clock gating is done automatically as a function of the design. There is no special control required for this operation.

Because the LCD normally drives displays, and because all video is sourced from the L3 clock domain, shutting down the L3 domain using the IPGenerics can cause undesirable display effects. In most circumstances, it will be necessary to hardware/software reset the LCD module after such an event has occurred.

## 13.4 Programming Model

### 13.4.1 LCD Character Displays

#### 13.4.1.1 Configuration Registers, Setup, and Settings

##### 13.4.1.1.1 Configuration Registers

Set the following to appropriate values for the target LCD character panel:

- cfg\_cs1\_e1\_pol
- cfg\_cs0\_e0\_pol
- ws\_dir\_pol
- cfg\_rs\_en\_pol
- cfg\_alepol

cfg\_lidd\_mode\_sel[2:0] defines the type of CPU bus that will be used in interfacing with the LCD character panel. Note that the clocked bus styles only support a single panel using CS0 since the clock pin takes a device pin that is otherwise used for CS1.

Set the following to appropriate bus timing parameters for the target LCD character panel:

- cfg\_w\_su
- cfg\_w\_strobe
- cfg\_w\_hold, cfg\_r\_su
- cfg\_r\_strobe
- cfg\_r\_hold
- cfg\_ta

A set of bus timing parameters are individually available for CS0 and CS1 such that the bus transactions can be customized for each of the two supported LCD character displays.

##### 13.4.1.1.2 Defining Panel Commands and Panel Data

In the Hitachi interface mode used for the example panel, whether the Character Panel understands a data transfer as Command or Data depends on the state of the REGSEL input pin. Writing to the cfg\_adr\_idx register will output a Command transfer. Writing to the cfg\_data register will result in a Data transfer.

Functionally, the ALE (lcd\_fp pin) from the LCD controller is tied to the REGSEL input of the character panel.

For example, to send byte 0xAB as a command to the previously described character panel, the CPU would write 0x00AB to the adr\_idx register. To send byte 0xAB as data, the CPU would write 0x00AB to the data register.

### 13.4.1.2 CPU Initiated Data Bus Transactions

#### 13.4.1.2.1 Initiating Data Bus Transactions

Writing to cfg\_cs0\_data will initiate a write transfer to the CS0 panel. Reading from cfg\_cs0\_data will initiate a read transfer from the CS0 panel.

Writing to cfg\_cs1\_data will initiate a write transfer to the CS1 panel. Reading from cfg\_cs1\_data will initiate a read transfer from the CS0 panel.

---

**NOTE:** Writes to CS1 translate to valid bus transactions only if cfg\_lidd\_mode\_sel[2:0] is configured for an asynchronous mode.

---

### 13.4.1.3 DMA Initiated Data Bus Transactions for LIDD

#### 13.4.1.3.1 DMA Overview for MPU Bus Output

Writing a long sequence of data to the Character Display Panel will ensure that the CPU will be occupied for a long time. However, the DMA module supports a mode in which this sequence of data elements can first be written in DRAM by the CPU.

The DMA can read this sequence of commands or data from the DRAM and send it to the LCD Interface Display Driver (LIDD) module such that each data element becomes a write bus transaction to the external Character Panel/MPU Bus. The data bus write transaction can target either CS0 or CS1 and use the appropriate bus timing parameters.

Functionally, in this DMA LIDD mode, the DMA module sends the sequence of data to the LIDD module by acting as another CPU.

The DMA can only perform write bus transactions. It cannot read from the external character panel a series of data elements and store them in the DRAM.

When the LIDD module is controlled by the DMA module by setting cfg\_lidd\_dma\_en = '1', CPU reads or writes to cfg\_adr\_index and cfg\_data are not allowed.

The fb0\_base and fb0\_ceil registers define the address boundary of data elements to be sent out the character display by the DMA engine. Setting cfg\_lidd\_dma\_en from '0' to '1' will initiate the DMA as if a virtual CPU is reading data from the DDR and writing the values to Reg6 or Reg9. cfg\_dma\_cs0\_cs1 determines whether the virtual CPU writes to Reg6 (CS0) or Reg7 (CS1).

---

**NOTE:** Writes to CS1 translate to valid bus transactions only if cfg\_lidd\_mode\_sel[2:0] is configured for an asynchronous mode.

---

The DMA module requires the start and end DDR addresses to be on word-aligned byte addresses. The MPU/LIDD bus is a halfword (16 bit) output, so both the upper and lower halfwords of the DDR memory will be sent out. Thus, the number of data elements sent to the LIDD by the DMA must always result in an even number of bus MPU bus transactions. In other words, a transfer of three 32-bit words from DDR will result in six 16-bit bus transactions.

### 13.4.1.3.2 MCU/LIDD DMA Setup: Example Pseudo Code

Suppose we want to send by DMA a section of DDR memory from byte address 0x4 to byte address 0x3C to the MCU bus using chip select 0. The pseudo code for such an operation is listed below.

```
// Enable Clocks
wr 006C 0000_0007

// LCD Control Register
wr 0004 0000_8000    // set clock divisor

// LIDD Control Register
wr 000C 0000_000C  // set output bus polarities and lidd_mode_sel

// LIDD CS0 Register
wr 0010 0822_1044    // set bus timing parameters for CS0

// DMA Control Register
wr 0040 0000_0030  // set DMA parameters like burst size, memory layout

// DMA FB0 Base Register
wr 0044 0000_0004  // DMA start byte address

// DMA FB0 Ceiling Register
wr 0048 0000_003C  // DMA end byte address

// LIDD Control Register - enable DMA
wr 000C 0000_010C  // Flip LIDD DMA enable bit
```

Once the DMA completes sending data to the Async FIFO, the Eof0 interrupt will occur. The Done interrupt will occur when the last word is written out the MPU bus.

The CPU must bring cfg\_lidd\_dma\_en low before the CPU can directly initiate MPU bus transactions or for the DMA module to start again.

### 13.4.1.4 Passive Matrix

#### 13.4.1.4.1 Monochrome Bitrate Awareness

In a mostly testbench related note, care must be taken when configuring the module for Passive Matrix (cfg\_lcdtft = '0') monochrome (cfg\_lcdbw = '1') modes. In passive matrix mode, the Blue component of the Grayscaler output is used as the quantized value for each scan order pixel.

When cfg\_mono8b='1', eight pixel values must be sent through the grayscaler before one 8-bit output is ready. This output data represents the passive matrix output states for eight pixels.

Likewise, when cfg\_mono8b='0', four pixel values must be sent through the grayscaler before one 4-bit output is ready. This output data represents the passive matrix output states for four pixels.

The problem arises when the output clock is fast (cfg\_clkdiv=0x2). The data path must output a value every two system clocks. However, it takes four or eight system clocks to generate a data element to be output. In this case, the LCD module returns an underflow interrupt.

In practice, such a situation does not occur because passive matrix panels are slow by design.

## 13.4.2 Active Matrix Displays

### 13.4.2.1 Interfacing to Dual LVDS Transmitters

The pixel clock rate for HD-sized pictures is approximately 148.5 MHz. At this speed, the LVDS link requires a double-wide data bus for transferring the even and odd pixels at half the pixel rate. The LCD Controller outputs one pixel per pixel clock cycle. Some LVDS transmitters accept a high speed, single pixel input and output to dual LVDS drivers, in which case external glue logic is unnecessary. For those LVDS transmitters that require the even and odd pixel to enter the LVDS transmitter at half the pixel clock rate, external logic is required.

## 13.4.3 System Interaction

### 13.4.3.1 DMA End of Frame Interrupts

The LCD module works with the DMA such that data is fetched from DDR and sent to a FIFO memory. The DMA module does this fetching independently of the logic on the output side of the FIFO.

For LIDD mode DMA, the module fetches frame buffer 0. When the last word of frame buffer 0 is stored in the FIFO memory, the Eof0 interrupt is triggered (if `cfg_eof_inten='1'`) and the DMA stops. The CPU has to set `cfg_lidd_dma_en='0'`, followed by a `cfg_lidd_dma_en='1'`, before the next burst from frame buffer 0 is read from DDR.

For Raster mode DMA, the module fetches frame buffer 0. When the last word of frame buffer 0 is stored in the FIFO memory, the Eof0 interrupt is triggered (if `cfg_eof_inten='1'`) but the DMA does not stop. The DMA module ping pongs immediately to frame buffer 1 if `cfg_frame_mode='1'`. Otherwise, the DMA fetches the frame buffer 0 address range from DDR. When the DMA module fetches frame buffer 1, and the last word of frame buffer 1 is stored in the FIFO memory, the Eof1 interrupt is triggered (if `cfg_eof_inten='1'`). This pattern would repeat.

## 13.4.4 Palette Lookup

For Active Matrix and Passive Matrix modes, the 12-bit Palette RAM Lookup can be used. For Active Matrix (`cfg_lcdtft = '1'`), palette lookup is enabled when `cfg_tft24 = '0'` and the bpp field in the Palette RAM is set to "000," "001," "010," or "011" (1, 2, 4, or 8 bpp). Palette lookup cannot be used when the bpp field is set to "100" (12/16 bpp).

For Passive Matrix (`cfg_lcdtft = '0'`), palette lookup is enabled when the bpp field in the Palette RAM is set to "000," "001," "010," or "011" (1, 2, 4, or 8 bpp). Palette lookup cannot be used when the bpp field is set to "100" (12/16 bpp).

Palette lookup scenarios are illustrated in [Figure 13-18](#).

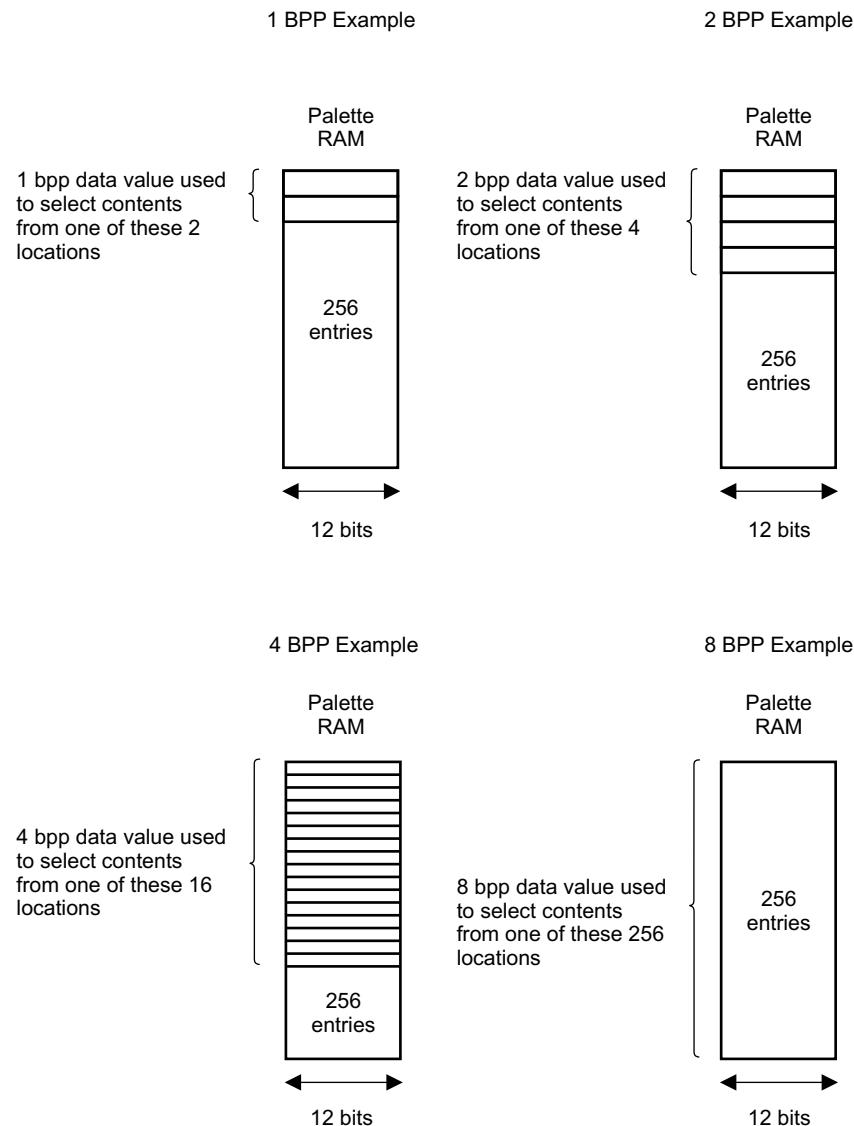
When the bpp encoding is set to 1 bpp, each bit in a 16-bit frame buffer halfword is used to index the two bottom locations of the palette RAM. Suppose the frame buffer bit value is '0', this '0' indicates that the address 0 entry in the Palette RAM should be read. If the frame buffer bit value is '1,' the address 1 entry in the Palette RAM is used. The resulting 12-bit output from the Palette RAM is the quantized pixel value of a 4-bit per color component quantized pixel value.

When the bpp encoding is set to 2 bpp, every two bits in a 16 bit frame buffer halfword is used to index the bottom 4 locations of the palette RAM. Suppose the frame buffer bit value is "00." This "00" indicates that the address 0 entry in the Palette RAM should be read. If the frame buffer bit value is "01," the address 1 entry in the Palette RAM is used. When the frame buffer bit value is "10," the address 2 entry in the Palette RAM is read. Finally, if the frame buffer bit value is "11," the address 3 entry in the Palette RAM is read. The resulting 12 bit output from the Palette RAM is the quantized pixel value of the 4 bit per color component.

The 4 bpp encoding allows every four bits from a frame buffer halfword to address 16 entries in the Palette RAM.

The 8 bpp encoding enables every byte from a frame buffer halfword to address one of the 256 entries in the Palette RAM.

Figure 13-18. Palette Lookup Examples



A 16-bit halfword is read from the DDR frame buffer. This halfword can be byte lane and halfword swapped using the DMA configuration values `cfg_byte_swap` and `cfg_bigendian`. This section will deal with the frame buffer data as it is returned post swapped from the DMA module.

The DMA module actually outputs a 32-bit word. The Palette Lookup logic uses the lower halfword first, followed by the upper halfword. The `cfg_rdorder` and `cfg_nibmode` registers determine the raster read ordering of the frame buffer data to be sent to the palette lookup table.

There are precedence rules for the hardware as it parses each 16-bit word from the frame buffer.

1. If `cfg_rdorder = '0'`, the data halfword is parsed from the least significant bit to the most significant bit.
2. Else, if `cfg_nibmode = '1'`, the data halfword is parsed byte swapped with the scan order going from the most significant bit of each byte to the least significant bit of each byte.
3. Otherwise, the data halfword is parsed from the most significant bit to the least significant bit.

The bitwise scan order for each halfword fetched from the frame buffer is shown in the following lists. The bitfields returned are used to determine the addressing of the Palette RAM.

#### **Frame buffer halfword scan order for 1 bpp**

1. If cfg\_rdorder = 0, scan order is [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11][ 12] [13] [14] [15]
2. Else if cfg\_nibmode=1, scan order is [7] [6] [5] [4] [3] [2] [1] [0] [15] [14] [13] [12] [11] [10] [9] [8]
3. Otherwise, scan order is [15] [14] [13] [12] [11] [10] [9] [8] [7] [6] [5] [4] [3] [2] [1] [0]

#### **Frame buffer halfword scan order for 2 bpp**

1. If cfg\_rdorder = 0, scan order is [1:0] [3:2] [5:4] [7:6] [9:8] [11:10][ 13:12] [15:14]
2. Else if cfg\_nibmode=1, scan order is [7:6] [5:4] [3:2] [1:0] [15:14] [13:12] [11:10] [9:8]
3. Otherwise, scan order is [15:14] [13:12] [11:10] [9:8] [7:6] [5:4] [3:2] [1:0]

#### **Frame buffer halfword scan order for 4 bpp**

1. If cfg\_rdorder = 0, scan order is [3:0] [7:4] [11:8] [15:12]
2. Else if cfg\_nibmode=1, scan order is [7:4] [3:0] [15:12] [11:8]
3. Otherwise, scan order is [15:12] [11:8] [7:4] [3:0]

#### **Frame buffer halfword scan order for 8 bpp**

1. If cfg\_rdorder = 0, scan order is [7:0] [15:8]
2. Else if cfg\_nibmode=1, scan order is [7:0] [15:8]
3. Otherwise, scan order is [15:8] [7:0]

### **13.4.5 Test Logic**

### **13.4.6 Disable and Software Reset Sequence**

In Raster Modes, the module must be disabled before applying a software reset. When cfg\_lcden is set to '0' to disable the module, the output continues to the end of the current frame.

The Done interrupt will trigger once the frame is complete. The software reset can then be applied to the module.

The software reset will clear all the frame information in the FIFO. Upon a restart, the L3 DMA will fetch from the fb0\_base address.

To summarize:

- Set cfg\_lcden='0'.
- Wait for the Done interrupt.
- Set the software reset bits high (cfg\_main\_rst='1' or [cfg\_dma\_rst='1' and cfg\_core\_rst='1']) for several cycles.
- Set the resets back low.
- Set cfg\_lcden='1'.

The disable and reset sequence must be done in this order to properly operate the LCD module and the EMIF.

### 13.4.7 Precedence Order for Determining Frame Buffer Type

The precedence order for determining frame buffer type is specified as follows:

```

If (cfg_lcdtft == 1) // active matrix
    If (cfg_tft24 == 1) // 24 bpp
        If (cfg_tft24_unpacked == 1)
            4 pixels in 4 words
        else
            4 pixels in 3 words
    else // 1/2/4/8/12/16 bpp
        if (bpp[2] == 1)
            12/16 bpp data
        else
            if (bpp == 0)
                1 bpp data
            else if (bpp == 1)
                2 bpp data
            else if (bpp == 2)
                4 bpp data
            else // if (bpp == 3)
                8 bpp data
else // passive matrix
    if (bpp[2] == 1)
        12/16 bpp data
    else
        if (bpp == 0)
            1 bpp data
        else if (bpp == 1)
            2 bpp data
        else if (bpp == 2)
            4 bpp data
        else // if (bpp == 3)
            8 bpp data

```

## 13.5 LCD Registers

Table 13-13 lists the memory-mapped registers for the LCD. All register offset addresses not listed in Table 13-13 should be considered as reserved locations and the register contents should not be modified.

**Table 13-13. LCD REGISTERS**

| Offset | Acronym            | Register Name | Section         |
|--------|--------------------|---------------|-----------------|
| 0h     | PID                |               | Section 13.5.1  |
| 4h     | CTRL               |               | Section 13.5.2  |
| Ch     | LIDD_CTRL          |               | Section 13.5.3  |
| 10h    | LIDD_CS0_CONF      |               | Section 13.5.4  |
| 14h    | LIDD_CS0_ADDR      |               | Section 13.5.5  |
| 18h    | LIDD_CS0_DATA      |               | Section 13.5.6  |
| 1Ch    | LIDD_CS1_CONF      |               | Section 13.5.7  |
| 20h    | LIDD_CS1_ADDR      |               | Section 13.5.8  |
| 24h    | LIDD_CS1_DATA      |               | Section 13.5.9  |
| 28h    | RASTER_CTRL        |               | Section 13.5.10 |
| 2Ch    | RASTER_TIMING_0    |               | Section 13.5.11 |
| 30h    | RASTER_TIMING_1    |               | Section 13.5.12 |
| 34h    | RASTER_TIMING_2    |               | Section 13.5.13 |
| 38h    | RASTER_SUBPANEL    |               | Section 13.5.14 |
| 3Ch    | RASTER_SUBPANEL2   |               | Section 13.5.15 |
| 40h    | LCDDMA_CTRL        |               | Section 13.5.16 |
| 44h    | LCDDMA_FB0_BASE    |               | Section 13.5.17 |
| 48h    | LCDDMA_FB0_CEILING |               | Section 13.5.18 |

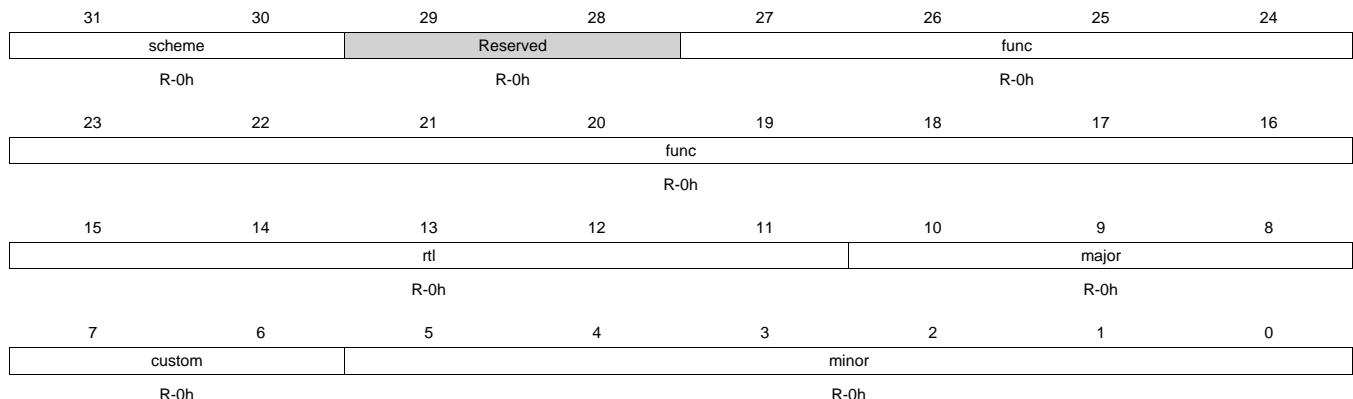
**Table 13-13. LCD REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b>     | <b>Register Name</b> | <b>Section</b>  |
|---------------|--------------------|----------------------|-----------------|
| 4Ch           | LCDDMA_FB1_BASE    |                      | Section 13.5.19 |
| 50h           | LCDDMA_FB1_CEILING |                      | Section 13.5.20 |
| 54h           | SYSCONFIG          |                      | Section 13.5.21 |
| 58h           | IRQSTATUS_RAW      |                      | Section 13.5.22 |
| 5Ch           | IRQSTATUS          |                      | Section 13.5.23 |
| 60h           | IRQENABLE_SET      |                      | Section 13.5.24 |
| 64h           | IRQENABLE_CLEAR    |                      | Section 13.5.25 |
| 6Ch           | CLKC_ENABLE        |                      | Section 13.5.26 |
| 70h           | CLKC_RESET         |                      | Section 13.5.27 |

### 13.5.1 PID Register (offset = 0h) [reset = 0h]

PID is shown in [Figure 13-19](#) and described in [Table 13-14](#).

**Figure 13-19. PID Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

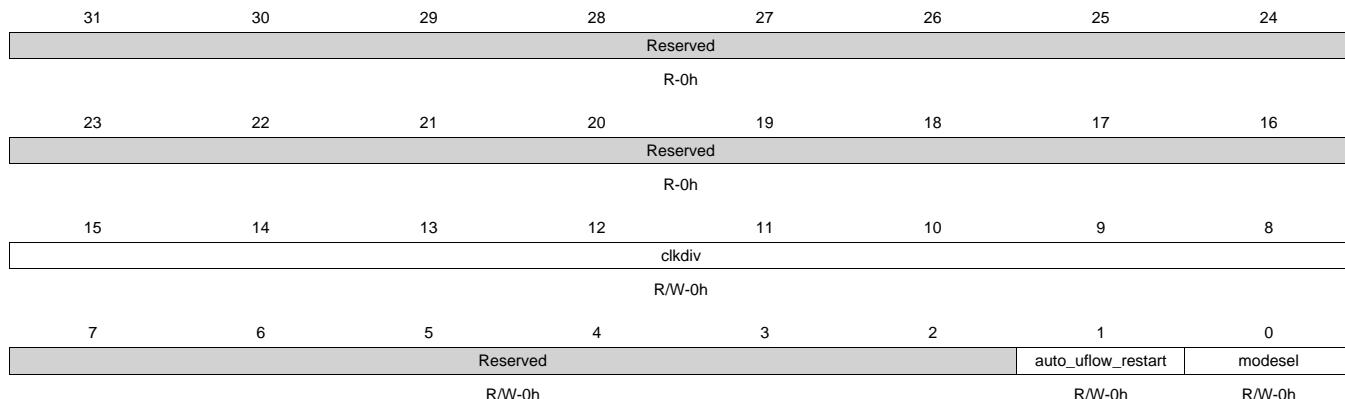
**Table 13-14. PID Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-30 | scheme   | R    | 0h    | The scheme of the register used<br>This field indicates the 3<br>5 Method |
| 29-28 | Reserved | R    | 0h    |   |
| 27-16 | func     | R    | 0h    | The function of the module being used                                     |
| 15-11 | rtl      | R    | 0h    | The Release number for this IP  |
| 10-8  | major    | R    | 0h    | Major Release Number  |
| 7-6   | custom   | R    | 0h    | Custom IP   |
| 5-0   | minor    | R    | 0h    | Minor Release Number  |

### 13.5.2 CTRL Register (offset = 4h) [reset = 0h]

CTRL is shown in [Figure 13-20](#) and described in [Table 13-15](#).

**Figure 13-20. CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

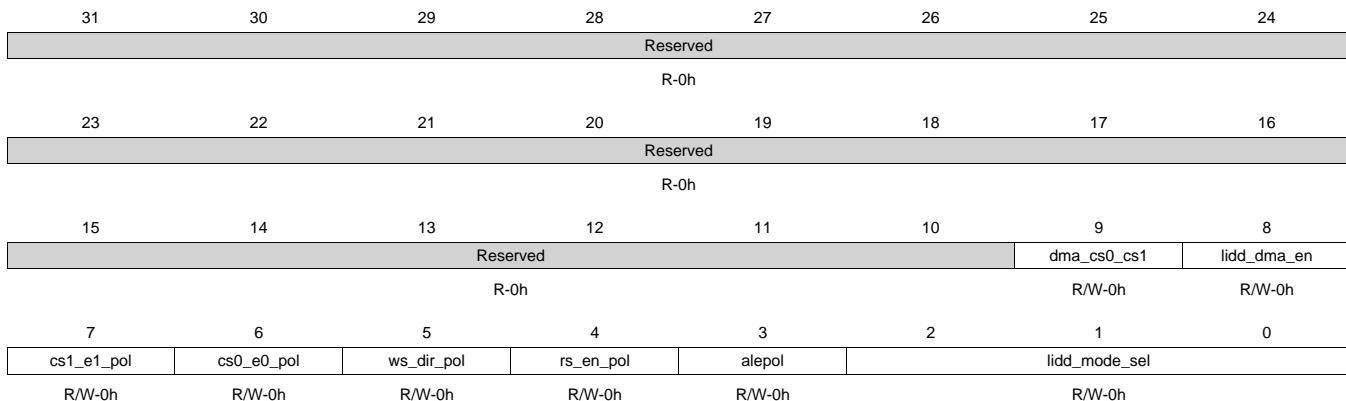
**Table 13-15. CTRL Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-16 | Reserved           | R    | 0h    |  |
| 15-8  | clkdiv             | R/W  | 0h    | Clock divisor<br>Raster mode: Values of 2 through 255 are permitted and resulting pixel clock is lcd_clk/2 through lcd_clk/255<br>LIDD mode: Values of 0 through 255 are permitted with resulting MCLK of lcd_clk/1 through lcd_clk/255 where both 0 and 1 result in lcd_clk/1 |
| 7-2   | Reserved           | R/W  | 0h    |  |
| 1     | auto_uflow_restart | R/W  | 0h    | 0 = On an underflow, the software has to restart the module 1 = On an underflow, the hardware will restart on the next frame   |
| 0     | modesel            | R/W  | 0h    | LCD Mode select<br>0 = LCD Controller in LIDD Mode 1 = LCD Controller in Raster Mode   |

### 13.5.3 LIDD\_CTRL Register (offset = Ch) [reset = 0h]

LIDD\_CTRL is shown in [Figure 13-21](#) and described in [Table 13-16](#).

**Figure 13-21. LIDD\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

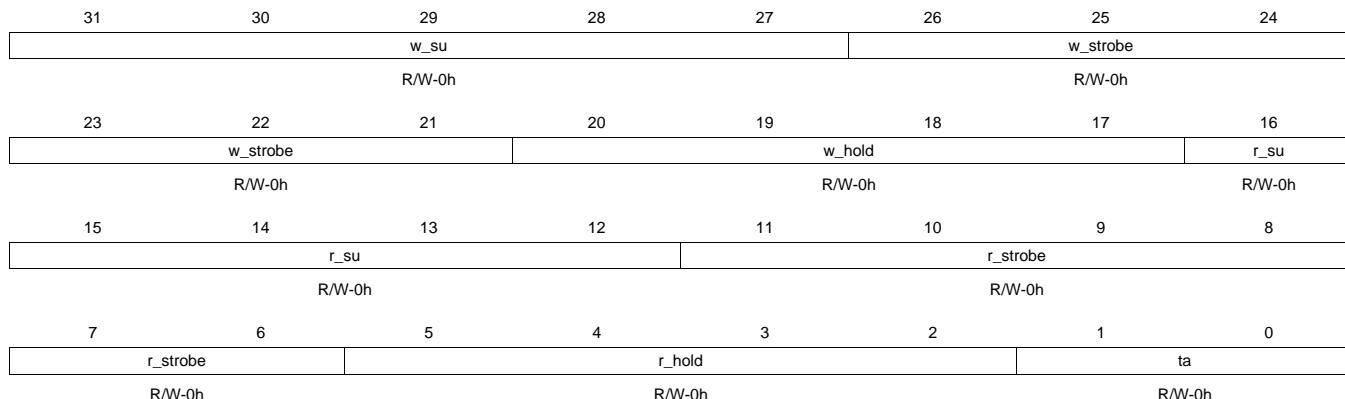
**Table 13-16. LIDD\_CTRL Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-10 | Reserved      | R    | 0h    |   |
| 9     | dma_cs0_cs1   | R/W  | 0h    | CS0/CS1 Select for LIDD DMA writes<br>0 = DMA writes to LIDD CS0 1 = DMA writes for LIDD CS1  |
| 8     | lidd_dma_en   | R/W  | 0h    | LIDD DMA Enable 0 = Deactivate DMA control of LIDD interface<br>DMA control is released upon completion of transfer of the current frame of data (LIDD Frame Done) after this bit is cleared<br>The MPU has direct read/write access to the panel in this mode<br>1 = Activate DMA to drive LIDD interface to support streaming data to smart panels<br>The MPU cannot access the panel directly in this mode |
| 7     | cs1_e1_pol    | R/W  | 0h    | Chip Select 1/Enable 1 (Secondary) Polarity Control 0 = Do Not Invert Chip Select 1/Enable 1<br>Chip Select 1 is active low by default<br>Enable 1 is active high by default<br>1 = Invert Chip Select 1/Enable 1   |
| 6     | cs0_e0_pol    | R/W  | 0h    | Chip Select 0/Enable 0 (Secondary) Polarity Control<br>0 = Do Not Invert Chip Select 0/Enable 0<br>Chip Select 0 is active low by default<br>Enable 0 is active high by default<br>1 = Invert Chip Select 0/Enable 0  |
| 5     | ws_dir_pol    | R/W  | 0h    | Write Strobe/Direction Polarity Control 0 = Do Not Invert Write Strobe/Direction<br>Write Strobe/Direction is active low/write low by default<br>1 = Invert Write Strobe/Direction  |
| 4     | rs_en_pol     | R/W  | 0h    | Read Strobe/Direction Polarity Control<br>0 = Do Not Invert Read Strobe/Direction<br>Read Strobe/Direction is active low/write low by default<br>1 = Invert Read Strobe/Direction   |
| 3     | alepol        | R/W  | 0h    | Address Latch Enable (ALE) Polarity Control 0 = Do Not Invert ALE<br>ALE is active low by default<br>1 = Invert   |
| 2-0   | lidd_mode_sel | R/W  | 0h    | LIDD Mode Select<br>Selects type of LCD display interface for the LIDD to drive: 000b = Sync MPU68 001b = Async MPU68 010b = Sync MPU80 011b = Async MPU80 100b = Hitachi (Async) 101b = N/A 110b = N/A 111b = N/A  |

### 13.5.4 LIDD\_CS0\_CONF Register (offset = 10h) [reset = 0h]

LIDD\_CS0\_CONF is shown in [Figure 13-22](#) and described in [Table 13-17](#).

**Figure 13-22. LIDD\_CS0\_CONF Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

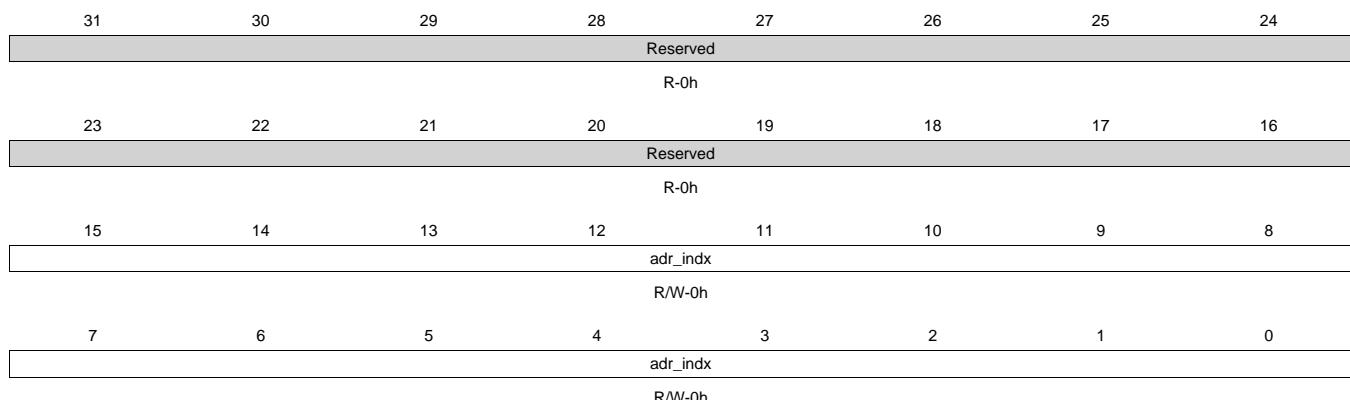
**Table 13-17. LIDD\_CS0\_CONF Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-27 | w_su     | R/W  | 0h    | Write Strobe Set-Up cycles<br>When performing a write access, this field defines the number of memclk cycles that Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 0 have to be ready before the Write Strobe is asserted<br>The minimum value is 0x0 |
| 26-21 | w_strobe | R/W  | 0h    | Write Strobe Duration cycles<br>Field value defines the number of memclk cycles for which the Write Strobe is held active when performing a write access<br>The minimum value is 0x1  |
| 20-17 | w_hold   | R/W  | 0h    | Write Strobe Hold cycles<br>Field value defines the number of memclk cycles for which Data Bus/Pas Output Enable, ALE, the Direction bit, and Chip Select 0 are held after the Write Strobe is de-asserted when performing write access<br>The minimum value is 0x1   |
| 16-12 | r_su     | R/W  | 0h    | Read Strobe Set-Up cycles<br>When performing a read access, this field defines the number of memclk cycles that Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 0 have to be ready before the Read Strobe is asserted                                |
| 11-6  | r_strobe | R/W  | 0h    | Read Strobe Duration cycles<br>Field value defines the number of memclk cycles for which the Read Strobe is held active when performing a read access<br>The minimum value is 0x1   |
| 5-2   | r_hold   | R/W  | 0h    | Read Strobe Hold cycles<br>Field value defines the number of memclk cycles for which Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 0 are held after the Read Strobe is deasserted when performing a read access<br>The minimum value is 0x1        |
| 1-0   | ta       | R/W  | 0h    | Field value defines the number of memclk (ta+1) cycles between the end of one CS0 device access and the start of another CS0 device access unless the two accesses are both Reads<br>In this case, this delay is not incurred<br>The minimum value is 0x0             |

### 13.5.5 LIDD\_CS0\_ADDR Register (offset = 14h) [reset = 0h]

LIDD\_CS0\_ADDR is shown in [Figure 13-23](#) and described in [Table 13-18](#).

**Figure 13-23. LIDD\_CS0\_ADDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

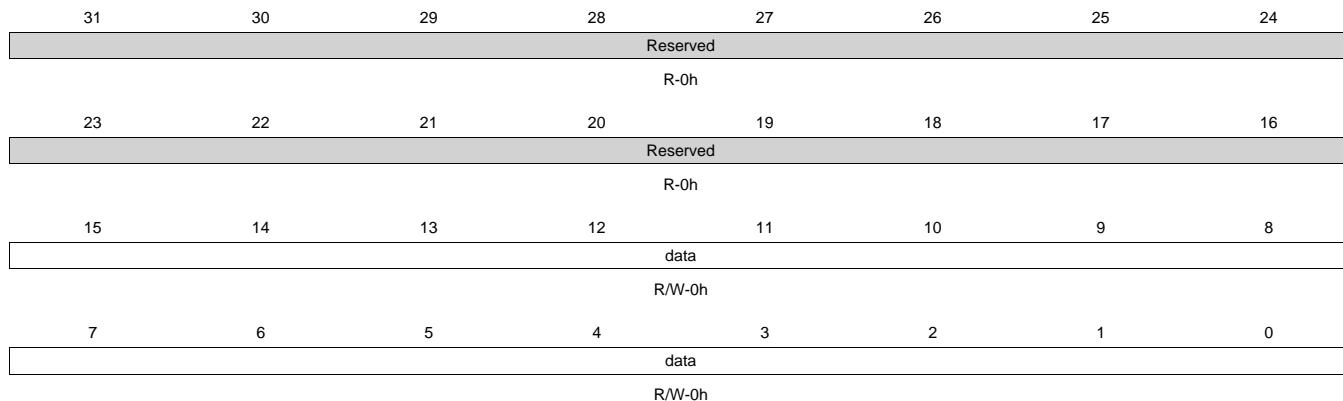
**Table 13-18. LIDD\_CS0\_ADDR Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-16 | Reserved | R    | 0h    |   |
| 15-0  | adr_idx  | R/W  | 0h    | The LCD Controller supports a shared Address/Data output bus<br>A write to this register would initiate a bus write transaction<br>A read from this register would initiate a bus read transaction<br>CPU reads and writes to this register are not permitted if the LIDD<br>module is in DMA mode (cfg_lidd_dma_en = 1)<br>If the LIDD is being used as a generic bus interface, writing to this<br>register can store adr_idx to an external transparent latch holding a<br>16-bit address<br>If the LIDD is being used to interface with a character based LCD<br>panel in configuration mode, reading and writing to this register can<br>be used to access the command instruction area of the panel |

### 13.5.6 LIDD\_CS0\_DATA Register (offset = 18h) [reset = 0h]

LIDD\_CS0\_DATA is shown in [Figure 13-24](#) and described in [Table 13-19](#).

**Figure 13-24. LIDD\_CS0\_DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

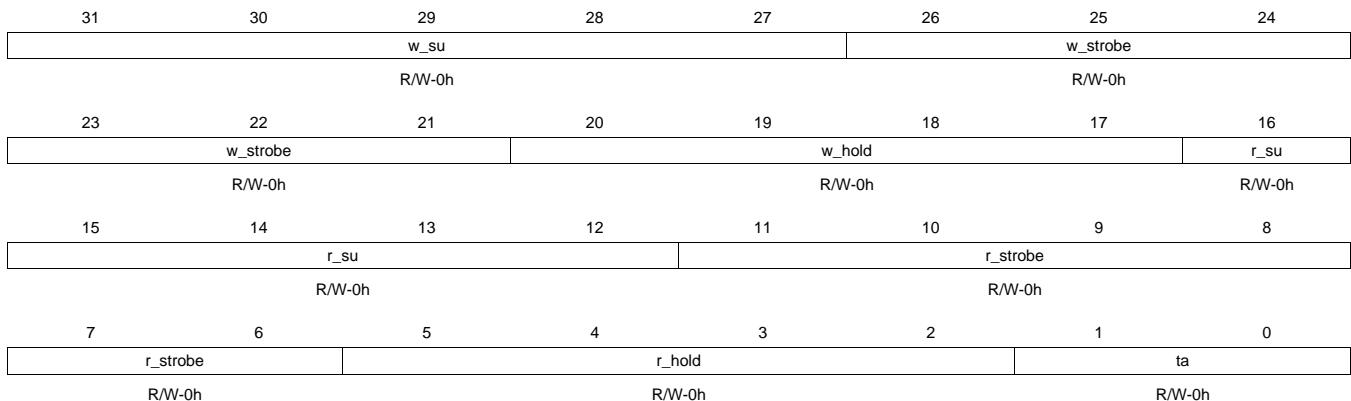
**Table 13-19. LIDD\_CS0\_DATA Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15-0  | data     | R/W  | 0h    | <p>The LCD Controller supports a shared Address/Data output bus</p> <p>A write to this register would initiate a bus write transaction</p> <p>A read from this register would initiate a bus read transaction</p> <p>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (cfg_lidd_dma_en = 1)</p> <p>If the LIDD is being used as a generic bus interface, writing to this register can store adr_indx to an external transparent latch holding a 16-bit address</p> <p>If the LIDD is being used to interface with a character based LCD panel in configuration mode, reading and writing to this register can be used to access the command instruction area of the panel</p> |

### 13.5.7 LIDD\_CS1\_CONF Register (offset = 1Ch) [reset = 0h]

LIDD\_CS1\_CONF is shown in [Figure 13-25](#) and described in [Table 13-20](#).

**Figure 13-25. LIDD\_CS1\_CONF Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

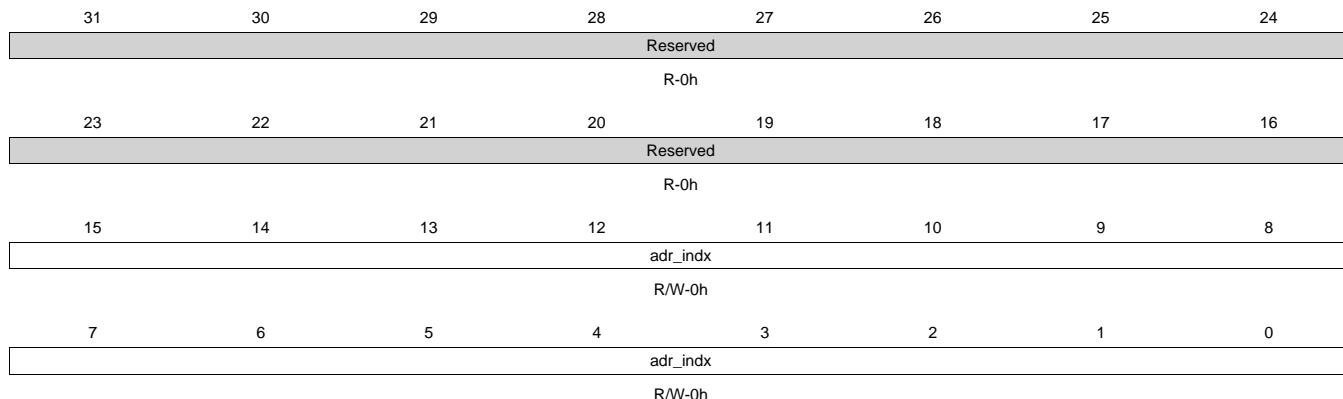
**Table 13-20. LIDD\_CS1\_CONF Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-27 | w_su     | R/W  | 0h    | Write Strobe Set-Up cycles<br>When performing a write access, this field defines the number of memclk cycles that Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 1 have to be ready before the Write Strobe is asserted<br>The minimum value is 0x0 |
| 26-21 | w_strobe | R/W  | 0h    | Write Strobe Duration cycles<br>Field value defines the number of memclk cycles for which the Write Strobe is held active when performing a write access<br>The minimum value is 0x1  |
| 20-17 | w_hold   | R/W  | 0h    | Write Strobe Hold cycles<br>Field value defines the number of memclk cycles for which Data Bus/Pad Output Enable, ALE, the Direction bit, and Chip Select 1 are held after the Write Strobe is deasserted when performing a write access<br>The minimum value is 0x1  |
| 16-12 | r_su     | R/W  | 0h    | Read Strobe Set-Up cycles<br>When performing a read access, this field defines the number of memclk cycles that Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 1 have to be ready before the Read Strobe is asserted<br>The minimum value is 0x0    |
| 11-6  | r_strobe | R/W  | 0h    | Read Strobe Duration cycles<br>Field value defines the number of memclk cycles for which the Read Strobe is held active when performing a read access<br>The minimum value is 0x1   |
| 5-2   | r_hold   | R/W  | 0h    | Read Strobe Hold cycles<br>Field value defines the number of memclk cycles for which Data Bus/Pad Output Enable, , the Direction bit, and Chip Select 1 are held after the Read Strobe is deasserted when performing a read access<br>The minimum value is 0x1        |
| 1-0   | ta       | R/W  | 0h    | Field value defines the number of memclk (ta+1) cycles between the end of one CS1 device access and the start of another CS1 device access unless the two accesses are both Reads<br>In this case, this delay is not incurred<br>The minimum value is 0x0             |

### 13.5.8 LIDD\_CS1\_ADDR Register (offset = 20h) [reset = 0h]

LIDD\_CS1\_ADDR is shown in [Figure 13-26](#) and described in [Table 13-21](#).

**Figure 13-26. LIDD\_CS1\_ADDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

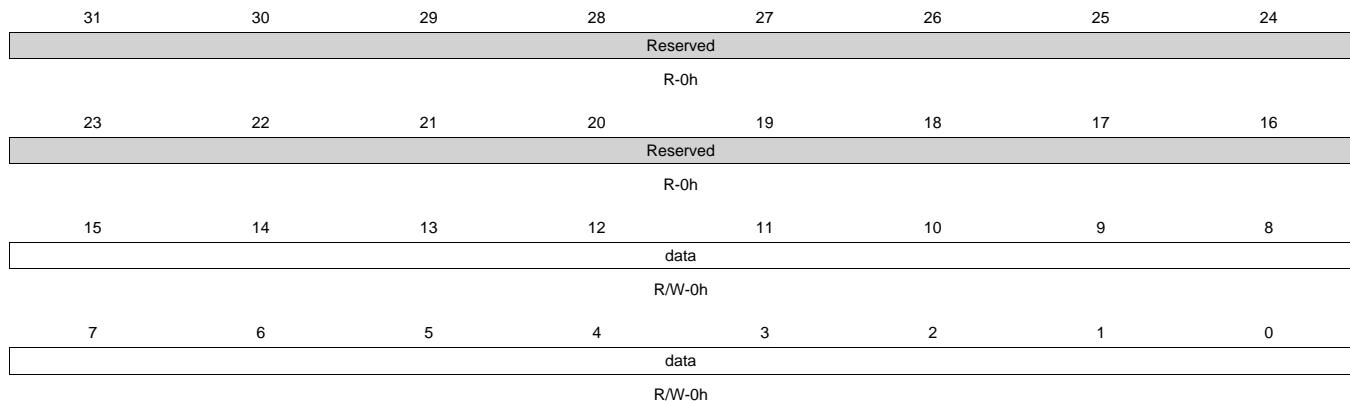
**Table 13-21. LIDD\_CS1\_ADDR Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-16 | Reserved | R    | 0h    |   |
| 15-0  | adr_idx  | R/W  | 0h    | The LCD Controller supports a shared Address/Data output bus<br>A write to this register would initiate a bus write transaction<br>A read from this register would initiate a bus read transaction<br>CPU reads and writes to this register are not permitted if the LIDD<br>module is in DMA mode (cfg_lidd_dma_en = 1)<br>If the LIDD is being used as a generic bus interface, writing to this<br>register can store adr_idx to an external transparent latch holding a<br>16-bit address<br>If the LIDD is being used to interface with a character based LCD<br>panel in configuration mode, reading and writing to this register can<br>be used to access the command instruction area of the panel |

### 13.5.9 LIDD\_CS1\_DATA Register (offset = 24h) [reset = 0h]

LIDD\_CS1\_DATA is shown in [Figure 13-27](#) and described in [Table 13-22](#).

**Figure 13-27. LIDD\_CS1\_DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-22. LIDD\_CS1\_DATA Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-16 | Reserved | R    | 0h    |   |
| 15-0  | data     | R/W  | 0h    | The LCD Controller supports a shared Address/Data output bus<br>A write to this register would initiate a bus write transaction<br>A read from this register would initiate a bus read transaction<br>CPU reads and writes to this register are not permitted if the LIDD<br>module is in DMA mode (cfg_lidd_dma_en = 1)<br>If the LIDD is being used as a generic bus interface, writing to this<br>register can store adr_idx to an external transparent latch holding a<br>16-bit address<br>If the LIDD is being used to interface with a character based LCD<br>panel in configuration mode, reading and writing to this register can<br>be used to access the command instruction area of the panel |

### 13.5.10 RASTER\_CTRL Register (offset = 28h) [reset = 0h]

RASTER\_CTRL is shown in Figure 13-28 and described in Table 13-23.

**Figure 13-28. RASTER\_CTRL Register**

|        |         |          |    |               |        |        |         |
|--------|---------|----------|----|---------------|--------|--------|---------|
| 31     | 30      | 29       | 28 | 27            | 26     | 25     | 24      |
|        |         | Reserved |    | tft24unpacked | tft24  | stn565 |         |
|        |         | R/W-0h   |    | R/W-0h        | R/W-0h | R/W-0h | R/W-0h  |
| 23     | 22      | 21       | 20 | 19            | 18     | 17     | 16      |
| tftmap | nibmode | palmode  |    |               | reqdly |        |         |
| R/W-0h | R/W-0h  | R/W-0h   |    |               | R/W-0h |        |         |
| 15     | 14      | 13       | 12 | 11            | 10     | 9      | 8       |
|        |         | reqdly   |    | Reserved      |        | nono8b | rdorder |
|        |         | R/W-0h   |    | R/W-0h        | R/W-0h | R/W-0h | R/W-0h  |
| 7      | 6       | 5        | 4  | 3             | 2      | 1      | 0       |
| lcdtft |         | Reserved |    |               |        | lcdbw  | lcden   |
| R/W-0h |         | R/W-0h   |    |               |        | R/W-0h | R/W-0h  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-23. RASTER\_CTRL Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-27 | Reserved      | R/W  | 0h    |   |
| 26    | tft24unpacked | R/W  | 0h    | 24 bit Mode Packing Only used when cfg_tft24=1 and cfg_lcdtft=1<br>0: 24-bit pixels are packed into 32 bit boundaries, which means 4 pixels are saved in every three words<br>Word 0: pix1[7:0], pix0[23:0]<br>Word 1: pix2[15:0], pix1[23:8]<br>Word 2: pix3[23:0], pix2[23:16]<br>1: 24-bit pixels are stored unpacked in with the uppermost byte unused<br>Word 0: Unused[7:0], pix0[23:0]<br>Word 1: Unused[7:0], pix1[23:0]<br>Word 2: Unused[7:0], pix2[23:0]<br>Word3: Unused[7:0], pix3[23:0]   |
| 25    | tft24         | R/W  | 0h    | 24 bit mode 0 = off 1 = on (24-bit data in active mode) The format of the framebuffer data depends on cfg_tft24unpacked   |
| 24    | stn565        | R/W  | 0h    | Passive Matrix Mode only (cfg_lcdtft='0') and 16 bpp raw data framebuffers (bpp = ?00?<br>If the bpp field in the framebuffer palette header is ?00?(12/16/24 bpp source), then the DDR contains raw data and the palette lookup is bypassed<br>Only for this case, this bit selects whether the framebuffer format is 16 bpp 565 or 12 bpp<br>The Grayscaler can only take 12 bits per pixel<br>The framebuffer data is 16 bits per pixel 565 when stn565 is set to '1' and only the 4 most significant bits of each color component are sent to the Grayscaler input<br>0: Framebuffer is 12 bpp packed in bits [11:0] 1: Framebuffer is 16 bpp 565 |
| 23    | tftmap        | R/W  | 0h    | TFT Mode Alternate Signal Mapping for Palettized framebuffer<br>Must be '0' for all 12/16/24 bpp raw data formats<br>Can only be '1' for 1/2/4/8 bpp Palette Lookup data<br>Valid only in Active Matrix mode when cfg_lcdtft='1'<br>0 = 4 bits per component output data for 1, 2, 4, and 8 bpp modes will be right aligned on lcd_pins (11:0) 1 = 4 bits per component output data for 1, 2, 4, and 8 bpp will be converted to 5,6,5, format and use pins (15:0) R3 R2 R1 R0 R3 G3 G2 G1 G0 G3 G2 B3 B2 B1 B0 B3   |
| 22    | nibmode       | R/W  | 0h    | Nibble Mode This bit is used to determine palette indexing and is used in conjunction with cfg_rdorder<br>0: Nibble mode is disabled 1: Nibble mode is enabled  |

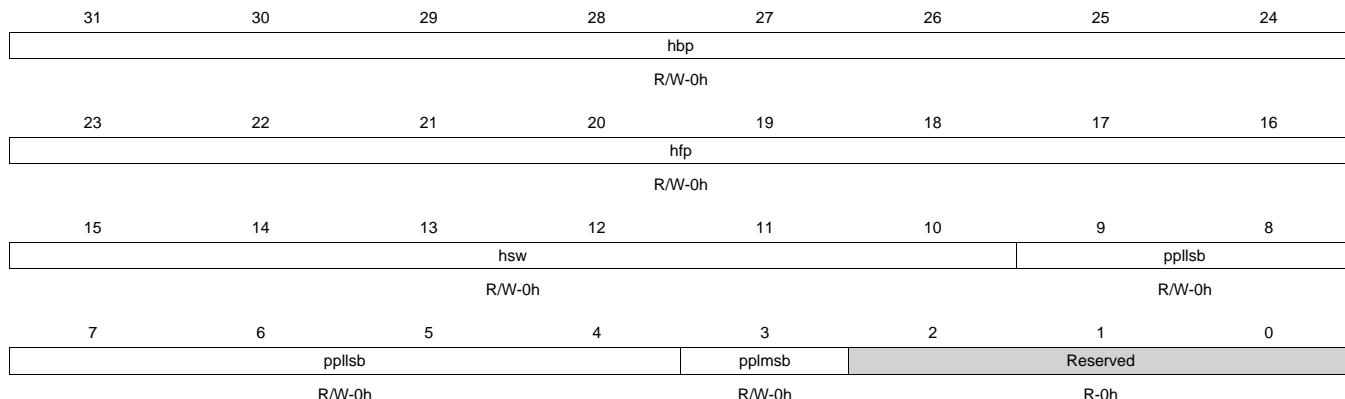
**Table 13-23. RASTER\_CTRL Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 21-20 | pemode   | R/W  | 0h    | Palette Loading Mode 00: Palette and data loading, reset value 01: Palette loading only 10: Data loading only For Raw Data (12/16/24 bpp) framebuffers, no palette lookup is employed Thus, these framebuffers should use the data-only loading mode   |
| 19-12 | reqdly   | R/W  | 0h    | Palette Loading Delay When loading the Palette from DDR, palette data is burst into the internal Palette SRAM from the Async FIFO 1-, 2-, and 4-bit per pixel framebuffer encodings use a fixed 16-word entry palette residing above the video data The 8 bit per pixel framebuffer encoding uses a 256-word entry palette residing above the video data Likewise, 12, 16, and 24 bit per pixel framebuffer encodings also define a 256-word entry palette even though these encodings will not do a full bit-depth palette lookup However, the 256-word palette entry must still be read from DDR as a framebuffer is fetched Bursting in 256 words in sequential lcd_clk cycles may cause the Async FIFO to underflow depending on the SOC DDR burst bandwidth This 8-bit reqdly parameter pauses reading of the Palette data from the Async FIFO between each burst of 16 words The delay is in terms of lcd_clk (system clock) cycles Value (0-255) used to specify the number of system clock cycles that should be paused between bursts of 16 word reads from the Async FIFO while loading the Palette SRAM Programming reqdly=00h disables this pause when loading the palette table |
| 11-10 | Reserved | R/W  | 0h    |  |
| 9     | nono8b   | R/W  | 0h    | Mono 8 bit 0: lcd_pixel_o[3:0] is used to output four pixel values to the panel each pixel clock transition 1: lcd_pixel_o[7:0] is used to output eight pixel values to the panel each pixel clock transition This bit is ignored in all other modes   |
| 8     | rdorder  | R/W  | 0h    | Raster Data Order Select 0: The frame buffer parsing for Palette Data lookup is from Bit 0 to bit 31 of the input word from the DMA output 1: The frame buffer parsing for Palette Data lookup is from Bit 31 to Bit 0 of the input word from the DMA output This bit has no effect on raw data framebuffers (12/16/24 bpp) This bit is used to determine palette indexing and is used in conjunction with cfg_nibmode   |
| 7     | lcdfft   | R/W  | 0h    | LCD 0: Passive or display operation enabled, dither logic is enabled 1: Active or display operation enabled, external palette and DAC required, dither logic bypassed, pin timing changes to support continuous pixel clock, output enable, vsync, and hsync   |
| 6-2   | Reserved | R/W  | 0h    |  |
| 1     | lcdbw    | R/W  | 0h    | Only Applies for Passive Matrix Panels LCD Monochrome 0: Color operation enabled 1: Monochrome operation enabled   |
| 0     | lcden    | R/W  | 0h    | LCD Controller Enable 0: LCD controller disabled 1: LCD controller enabled   |

### 13.5.11 RASTER\_TIMING\_0 Register (offset = 2Ch) [reset = 0h]

RASTER\_TIMING\_0 is shown in [Figure 13-29](#) and described in [Table 13-24](#).

**Figure 13-29. RASTER\_TIMING\_0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

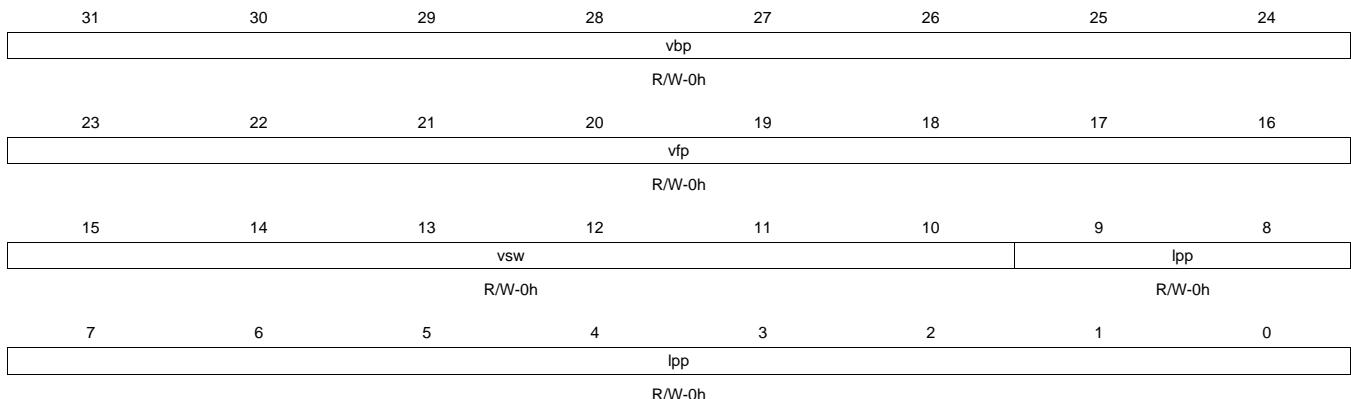
**Table 13-24. RASTER\_TIMING\_0 Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | hbp      | R/W  | 0h    | Horizontal Back Porch Lowbits Bits 7:0 of the horizontal back porch field<br>Encoded value (from 1-1024) used to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (programmed value plus 1)<br>Note that pixel clock is held in its inactive state during the beginning of the line wait period in passive display mode, and is permitted to transition in active display mode |
| 23-16 | hfp      | R/W  | 0h    | Horizontal Front Porch Lowbits Encoded value (from 1-1024) used to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (programmed value plus 1)<br>Note that pixel clock is held in its inactive state during the end of line wait period in passive display mode, and is permitted to transition in active display mode   |
| 15-10 | hsw      | R/W  | 0h    | Horizontal Sync Pulse Width Lowbits Bits 5:0 of the horizontal sync pulse width field<br>Encoded value (from 1-1024) used to specify the number of pixel clock periods to pulse the line clock at the end of each line (programmed value plus 1)<br>Note that pixel clock is held in its inactive state during the generation of line clock in passive display mode, and is permitted to transition in active display mode   |
| 9-4   | ppllsb   | R/W  | 0h    | Pixels-per-line LSB[9:4] Encoded LSB value (from 1-1024) used to specify the number of pixels contained within each line on the LCD display (programmed to value minus one)<br>PPL = 11'b{pplmsb, ppllsb, 4'b1111} + 1 ex: pplmsb=1'b1, pppllsb=6'0001000 PPL = 11'b{1, 000100, 1111}+ 1 = 1104d (decimal) pixels per line<br>In other words, PPL = 16*({pplmsb, ppllsb}+1)  |
| 3     | pplmsb   | R/W  | 0h    | Pixels-per-line MSB[10] Needed in order to support up to 2048 ppl  |
| 2-0   | Reserved | R    | 0h    |  |

### 13.5.12 RASTER\_TIMING\_1 Register (offset = 30h) [reset = 0h]

RASTER\_TIMING\_1 is shown in [Figure 13-30](#) and described in [Table 13-25](#).

**Figure 13-30. RASTER\_TIMING\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-25. RASTER\_TIMING\_1 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 31-24 | vbp   | R/W  | 0h    | Vertical Back Porch Value (from 0-255) use to specify the number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display<br>Note that line clock transitions during the insertion of the extra line clock periods  |
| 23-16 | vfp   | R/W  | 0h    | Vertical Front Porch Value (from 0-255) used to specify the number of line clock periods to add to the end of each frame<br>Note that the line clock transitions during the insertion of the extra line clock periods   |
| 15-10 | vsw   | R/W  | 0h    | Vertical Sync Width Pulse In active mode (lcdft=1), encoded value (from 1-64) used to specify the number of line clock periods to set the lcd_fp pin active at the end of each frame after the (vfp) period elapses<br>The number of clock cycles is programmed value plus one<br>The frame clock is used as the VSYNC signal in active mode<br>In passive mode (lcdft=0), encoded value (from 1-64) used to specify the number of extra line clock periods to insert after the vertical front porch (vfp) period has elapsed<br>Note that the width of lcd_fp is not affected by vsw in passive mode and line clock transitions during the insertion of the extra line clock periods (programmed value plus one) |
| 9-0   | lpp   | R/W  | 0h    | Lines Per Panel Encoded value (programmed value range of {0:2047} represents an actual range of {1:2048}) used to specify the number of lines per panel<br>It represents the total number of lines on the LCD (programmed value plus one)<br>Bit 10 of this field is in RASTER_TIMING_2   |

### 13.5.13 RASTER\_TIMING\_2 Register (offset = 34h) [reset = 0h]

RASTER\_TIMING\_2 is shown in [Figure 13-31](#) and described in [Table 13-26](#).

**Figure 13-31. RASTER\_TIMING\_2 Register**

| 31       | 30     | 29           | 28     | 27       | 26           | 25           | 24       |
|----------|--------|--------------|--------|----------|--------------|--------------|----------|
| Reserved |        | hsw_highbits |        |          | lpp_b10      | phsvs_on_off | phsvs_rf |
| R-0h     |        | R/W-0h       |        |          | R/W-0h       | R/W-0h       | R/W-0h   |
| 23       | 22     | 21           | 20     | 19       | 18           | 17           | 16       |
| ieo      | ipc    | ihs          | ivs    |          | acbi         |              |          |
| R/W-0h   | R/W-0h | R/W-0h       | R/W-0h |          | R/W-0h       |              |          |
| 15       | 14     | 13           | 12     | 11       | 10           | 9            | 8        |
|          |        |              | acb    |          | R/W-0h       |              |          |
| 7        | 6      | 5            | 4      | 3        | 2            | 1            | 0        |
| Reserved |        | hbp_highbits |        | Reserved | hfp_highbits |              |          |
| R-0h     |        | R/W-0h       |        | R-0h     | R/W-0h       |              |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-26. RASTER\_TIMING\_2 Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31    | Reserved     | R    | 0h    |   |
| 30-27 | hsw_highbits | R/W  | 0h    | Bits 9:6 of the horizontal sync width field   |
| 26    | lpp_b10      | R/W  | 0h    | Lines Per Panel Bit 10 Bit 10 of the lpp field in RASTER_TIMING_1   |
| 25    | phsvs_on_off | R/W  | 0h    | Hsync/Vsync Pixel Clock Control On/Off 0 = lcd_lp and lcd_fp are driven on opposite edges of pixel clock than the lcd_pixel_o 1 = lcd_lp and lcd_fp are driven according to bit 24 Note that this bit MUST be set to '0' for Passive Matrix displays The edge timing is fixed   |
| 24    | phsvs_rf     | R/W  | 0h    | Program HSYNC/VSYNC Rise or Fall 0: lcd_lp and lcd_fp are driven on the rising edge of pixel clock (bit 25 must be set to 1) 1: lcd_lp and lcd_fp are driven on the falling edge of pixel clock (bit 25 must be set to 1)   |
| 23    | ieo          | R/W  | 0h    | Invert Output Enable 0 = lcd_ac pin is active high in active display mode 1 = lcd_ac pin is active low in active display mode Active display mode: data driven out of the LCD's data lines on programmed pixel clock edge where AC-bias is active Note that ieo is ignored in passive display mode  |
| 22    | ipc          | R/W  | 0h    | Invert Pixel Clock 0 = Data is driven on the LCD's data lines on the rising edge of lcd_cp 1 = Data is driven on the LCD's data lines in the falling edge of lcd_cp For Active Matrix output (cfg_lcdftf='1'), the Output Pixel Clock is a free running clock in that it transitions in horizontal blanking (including horizontal front porch, horizontal back porch) areas and all vertical blanking times For Passive Matrix output (cfg_lcdftf='0'), the Output Pixel Clock on occurs when an output data value is written It is in a return-to-zero state when cfg_ipc='0' and a return-to-one state when cfg_ipc='1' |
| 21    | ihs          | R/W  | 0h    | Invert Hsync 0: lcd_lp pin is active high and inactive low 1: lcd_lp pin is active low and inactive high Active and passive mode: horizontal sync pulse/line clock active between lines, after the end of line wait period  |
| 20    | ivs          | R/W  | 0h    | Invert Vsync 0 = lcd_fp pin is active high and inactive low 1 = lcd_fp pin is active low and inactive high Active mode: vertical sync pulse active between frames, after end of frame wait period Passive mode: frame clock active during first line of each frame  |

**Table 13-26. RASTER\_TIMING\_2 Register Field Descriptions (continued)**

| Bit   | Field        | Type | Reset | Description  |
|-------|--------------|------|-------|--|
| 19-16 | acbi         | R/W  | 0h    | AC Bias Pins Transitions per Interrupt Value (from 0 to 15) used to specify the number of AC Bias pin transitions to count before setting the line count status (lcs) bit, signaling an interrupt request<br>Counter frozen when lcd is set, and is restarted when lcs is cleared by software<br>This function is disabled when acbi = b'0000' |
| 15-8  | acb          | R/W  | 0h    | AC Bias Pin Frequency Value (from 0-255) used to specify the number of line clocks to count before transitioning the AC Bias pin<br>This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display<br>acb = Number of line clocks/toggle of the lcd_ac pin                          |
| 7-6   | Reserved     | R    | 0h    |  |
| 5-4   | hbp_highbits | R/W  | 0h    | Bits 9:8 of the horizontal back porch field  |
| 3-2   | Reserved     | R    | 0h    |  |
| 1-0   | hfp_highbits | R/W  | 0h    | Bits 9:8 of the horizontal front porch field   |

### 13.5.14 RASTER\_SUBPANEL Register (offset = 38h) [reset = 0h]

RASTER\_SUBPANEL is shown in [Figure 13-32](#) and described in [Table 13-27](#).

**Figure 13-32. RASTER\_SUBPANEL Register**

| 31     | 30       | 29     | 28     | 27       | 26 | 25     | 24 |
|--------|----------|--------|--------|----------|----|--------|----|
| spen   | Reserved | hols   |        | Reserved |    | lppt   |    |
| R/W-0h | R-0h     | R/W-0h |        | R/W-0h   |    | R/W-0h |    |
| 23     | 22       | 21     | 20     | 19       | 18 | 17     | 16 |
|        |          |        |        | lppt     |    |        |    |
|        |          |        |        | R/W-0h   |    |        |    |
| 15     | 14       | 13     | 12     | 11       | 10 | 9      | 8  |
|        |          |        | dpdlsb |          |    |        |    |
|        |          |        |        | R/W-0h   |    |        |    |
| 7      | 6        | 5      | 4      | 3        | 2  | 1      | 0  |
|        |          |        |        | dpdlsb   |    |        |    |
|        |          |        |        | R/W-0h   |    |        |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

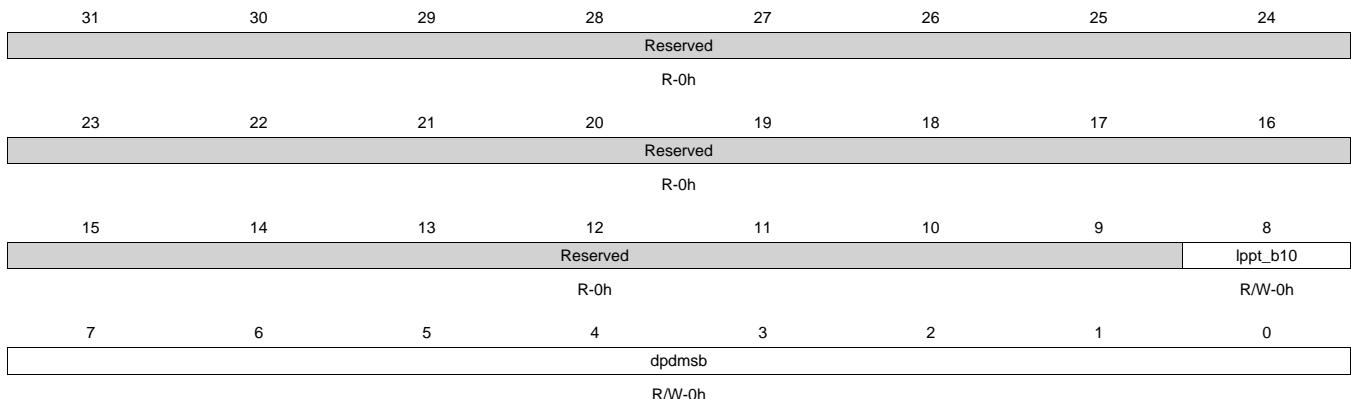
**Table 13-27. RASTER\_SUBPANEL Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31    | spen     | R/W  | 0h    | Sub Panel Enable 0 = function disabled 1 = sub-panel function mode enabled   |
| 30    | Reserved | R    | 0h    |  |
| 29    | hols     | R/W  | 0h    | High or Low Signal This field indicates the position of the sub-panel based on the LPPT value<br>0 = Default Pixel Data lines are at the top of the screen and the active video lines are at the bottom of the screen<br>1 = Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen  |
| 28-26 | Reserved | R/W  | 0h    |  |
| 25-16 | lppt     | R/W  | 0h    | Line Per Panel Threshold Encoded value (programmed value range of {0:2047} represents an actual range of {1:2048}) used to specify the number of lines on the bottom part of the panel<br>Bit10 of this field is in RASTER_SUBPANEL2<br>Hols determines whether Default Pixel Data is on the top (hols='0') or on the bottom (hols='1')<br>Lppt defines the number of lines on the bottom part of the output |
| 15-0  | dpdlsb   | R/W  | 0h    | Default Pixel Data LSB[15:0] DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reach or after passing LPPT   |

### 13.5.15 RASTER\_SUBPANEL2 Register (offset = 3Ch) [reset = 0h]

RASTER\_SUBPANEL2 is shown in [Figure 13-33](#) and described in [Table 13-28](#).

**Figure 13-33. RASTER\_SUBPANEL2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-28. RASTER\_SUBPANEL2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-9 | Reserved | R    | 0h    |  |
| 8    | lppt_b10 | R/W  | 0h    | Lines Per Panel Threshold Bit 10 This register is Bit 10 of the lppt field in RASTER_SUBPANEL  |
| 7-0  | dpdmsb   | R/W  | 0h    | Default Pixel Data MSB [23:16] DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reached or after passing the LPPT |

### 13.5.16 LCDDMA\_CTRL Register (offset = 40h) [reset = 0h]

LCDDMA\_CTRL is shown in Figure 13-34 and described in Table 13-29.

**Figure 13-34. LCDDMA\_CTRL Register**

|          |    |            |    |                 |    |          |            |
|----------|----|------------|----|-----------------|----|----------|------------|
| 31       | 30 | 29         | 28 | 27              | 26 | 25       | 24         |
| Reserved |    |            |    |                 |    |          |            |
| R/W-0h   |    |            |    |                 |    |          |            |
| 23       | 22 | 21         | 20 | 19              | 18 | 17       | 16         |
| Reserved |    |            |    | dma_master_prio |    |          |            |
| R/W-0h   |    |            |    |                 |    |          |            |
| 15       | 14 | 13         | 12 | 11              | 10 | 9        | 8          |
| Reserved |    |            |    | th_fifo_ready   |    |          |            |
| R/W-0h   |    |            |    |                 |    |          |            |
| 7        | 6  | 5          | 4  | 3               | 2  | 1        | 0          |
| Reserved |    | burst_size |    | byte_swap       |    | Reserved | big endian |
| R/W-0h   |    | R/W-0h     |    | R/W-0h          |    | R-0h     | R/W-0h     |
| R/W-0h   |    | R/W-0h     |    | R/W-0h          |    | R/W-0h   |            |

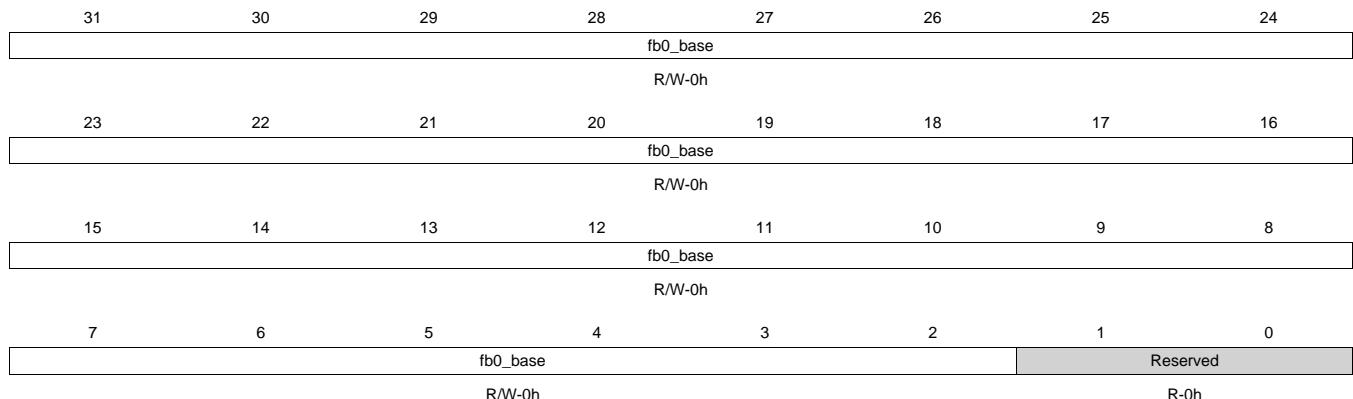
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-29. LCDDMA\_CTRL Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-19 | Reserved        | R/W  | 0h    |   |
| 18-16 | dma_master_prio | R/W  | 0h    | Priority for the L3 OCP Master Bus  |
| 15-11 | Reserved        | R/W  | 0h    |   |
| 10-8  | th_fifo_ready   | R/W  | 0h    | DMA FIFO threshold<br>The DMA FIFO becomes ready when the number of words specified by this register from the frame buffer have been loaded<br>000 = 8 001 = 16 010 = 32 011 = 64 100 = 128 101 = 256 110 = 512<br>111 = reserved   |
| 7     | Reserved        | R/W  | 0h    |   |
| 6-4   | burst_size      | R/W  | 0h    | Burst Size setting for DMA transfers (all DMA transfers are 32 bits wide): 000 = burst size of 1 001 = burst size of 2 010 = burst size of 4 011 = burst size of 8 100 = burst size of 16 101 = N/A 110 = N/A 111 = N/A burst_size cannot be changed once the DMA is enabled in LIDD or Raster modes<br>In this case, the DMA must be disabled and the Done Interrupt must occur before the value in this register can be changed |
| 3     | byte_swap       | R/W  | 0h    | This bit controls the bytelane ordering of the data on the output of the DMA module<br>It works in conjunction with the big-endian bit<br>See the big-endian description for configuration guidelines   |
| 2     | Reserved        | R    | 0h    |   |
| 1     | big_endian      | R/W  | 0h    | Big Endian Enable<br>Use this bit when the processor is operating in Big Endian mode AND writes to the frame buffer(s) are less than 32 bits wide<br>Only in this scenario do we need to change the byte alignment for data coming into the FIFO from the frame buffer(s)<br>0 = Big Endian data reordering disabled 1 = Big Endian data reordering enabled   |
| 0     | frame_mode      | R/W  | 0h    | Frame Mode 0 = one frame buffer (FB0 only) used 1 = two frame buffers used DMA ping-pongs between FB0 and FB1 in this mode  |

**13.5.17 LCDDMA\_FBO\_BASE Register (offset = 44h) [reset = 0h]**

LCDDMA\_FBO\_BASE is shown in [Figure 13-35](#) and described in [Table 13-30](#).

**Figure 13-35. LCDDMA\_FBO\_BASE Register**


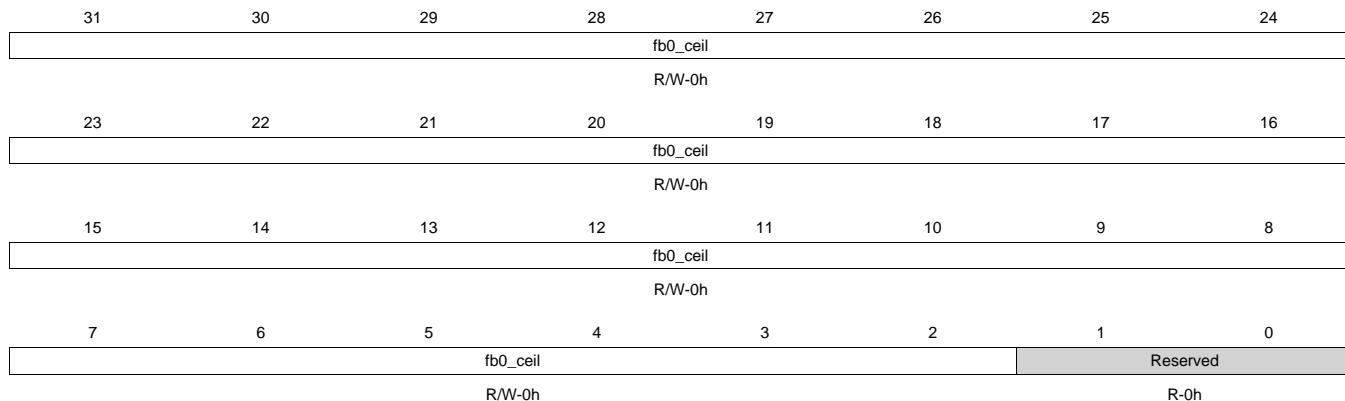
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-30. LCDDMA\_FBO\_BASE Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                         |
|------|----------|------|-------|-------------------------------------|
| 31-2 | fb0_base | R/W  | 0h    | Frame Buffer 0 Base Address pointer |
| 1-0  | Reserved | R    | 0h    |                                     |

**13.5.18 LCDDMA\_FBO\_CEILING Register (offset = 48h) [reset = 0h]**

LCDDMA\_FBO\_CEILING is shown in [Figure 13-36](#) and described in [Table 13-31](#).

**Figure 13-36. LCDDMA\_FBO\_CEILING Register**


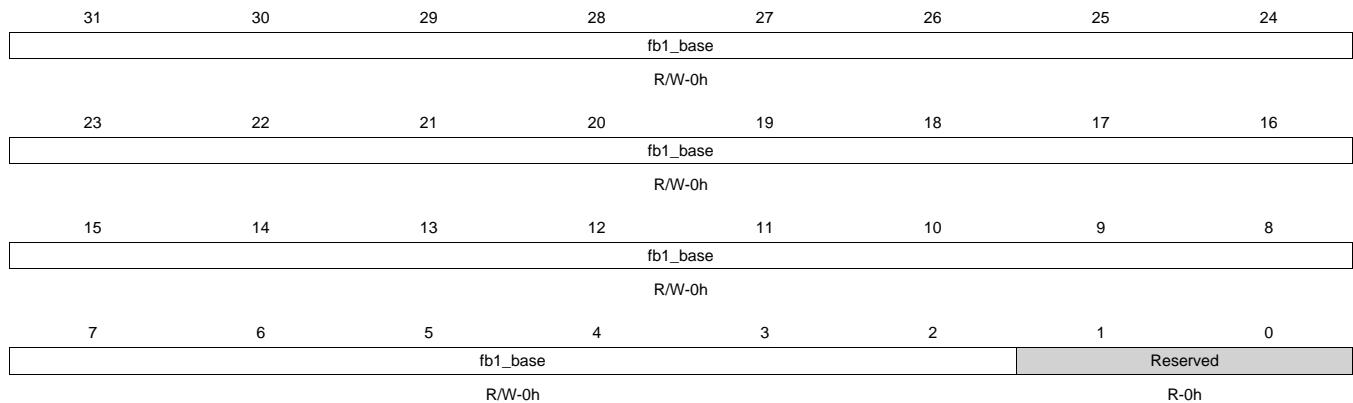
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-31. LCDDMA\_FBO\_CEILING Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                            |
|------|----------|------|-------|--|
| 31-2 | fb0.ceil | R/W  | 0h    | Frame Buffer 0 Ceiling Address pointer |
| 1-0  | Reserved | R    | 0h    |  |

**13.5.19 LCDDMA\_FB1\_BASE Register (offset = 4Ch) [reset = 0h]**

LCDDMA\_FB1\_BASE is shown in [Figure 13-37](#) and described in [Table 13-32](#).

**Figure 13-37. LCDDMA\_FB1\_BASE Register**


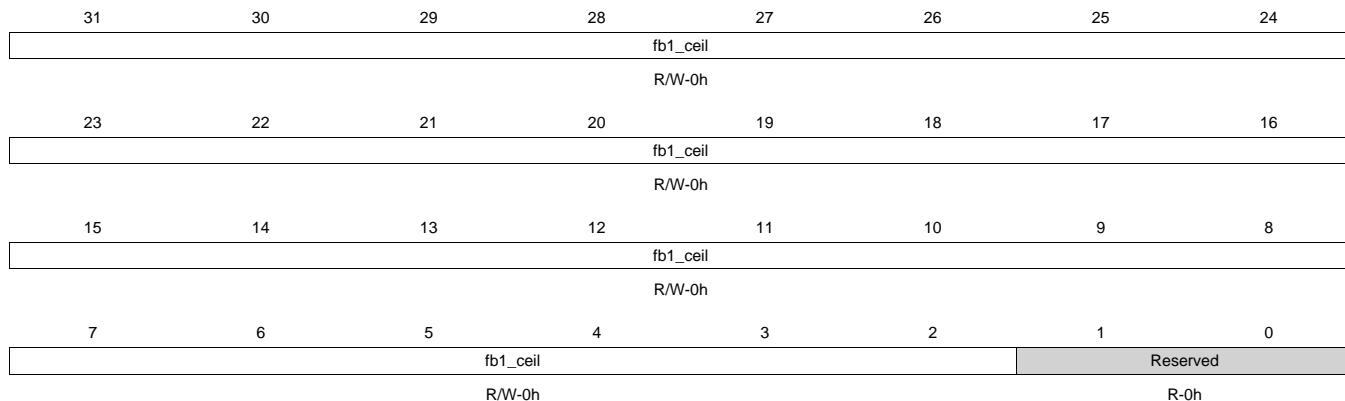
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-32. LCDDMA\_FB1\_BASE Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                         |
|------|----------|------|-------|-------------------------------------|
| 31-2 | fb1_base | R/W  | 0h    | Frame Buffer 1 Base Address pointer |
| 1-0  | Reserved | R    | 0h    |                                     |

**13.5.20 LCDDMA\_FB1\_CEILING Register (offset = 50h) [reset = 0h]**

LCDDMA\_FB1\_CEILING is shown in [Figure 13-38](#) and described in [Table 13-33](#).

**Figure 13-38. LCDDMA\_FB1\_CEILING Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

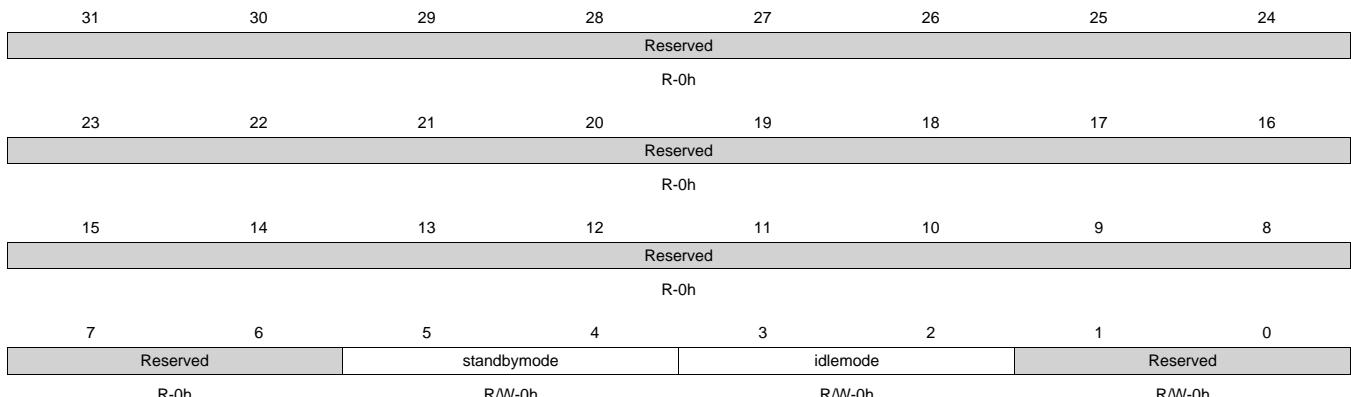
**Table 13-33. LCDDMA\_FB1\_CEILING Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                            |
|------|----------|------|-------|--|
| 31-2 | fb1.ceil | R/W  | 0h    | Frame Buffer 1 Ceiling Address pointer |
| 1-0  | Reserved | R    | 0h    |  |

### 13.5.21 SYSCONFIG Register (offset = 54h) [reset = 0h]

SYSCONFIG is shown in [Figure 13-39](#) and described in [Table 13-34](#).

**Figure 13-39. SYSCONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-34. SYSCONFIG Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description   |
|------|-------------|------|-------|---|
| 31-6 | Reserved    | R    | 0h    |   |
| 5-4  | standbymode | R/W  | 0h    | Configuration of the local initiator state management mode<br>By definition, initiator may generate read/write transaction as long as it is out of STANDBY state<br>0: Force-standby mode: local initiator is unconditionally placed in standby state<br>Backup mode, for debug only<br>1: No-standby mode: local initiator is unconditionally placed out of standby state<br>Backup mode, for debug only<br>2: Smart-standby mode: local initiator standby status depends on local conditions, i.e.<br>the module's functional requirement from the initiator<br>IP module shall not generate (initiator-related) wakeup events<br>3: Reserved   |
| 3-2  | idlemode    | R/W  | 0h    | Configuration of the local target state management mode<br>By definition, target can handle read/write transaction as long as it is out of IDLE state<br>0: Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e.<br>regardless of the IP module's internal requirements<br>Backup mode, for debug only<br>1: No-idle mode: local target never enters idle state<br>Backup mode, for debug only<br>2: Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements<br>IP module shall not generate (IRQ- or DMA-request-related) wakeup events<br>3: Reserved |
| 1-0  | Reserved    | R/W  | 0h    |   |

### 13.5.22 IRQSTATUS\_RAW Register (offset = 58h) [reset = 0h]

IRQSTATUS\_RAW is shown in Figure 13-40 and described in Table 13-35.

**Figure 13-40. IRQSTATUS\_RAW Register**

|          |            |             |          |             |              |                               |              |
|----------|------------|-------------|----------|-------------|--------------|-------------------------------|--------------|
| 31       | 30         | 29          | 28       | 27          | 26           | 25                            | 24           |
| Reserved |            |             |          |             |              |                               |              |
| R/W-0h   |            |             |          |             |              |                               |              |
| 23       | 22         | 21          | 20       | 19          | 18           | 17                            | 16           |
| Reserved |            |             |          |             |              |                               |              |
| R/W-0h   |            |             |          |             |              |                               |              |
| 15       | 14         | 13          | 12       | 11          | 10           | 9                             | 8            |
| Reserved |            |             |          |             |              | eof1_raw_set                  | eof0_raw_set |
| R/W-0h   |            |             |          |             |              |                               |              |
| 7        | 6          | 5           | 4        | 3           | 2            | 1                             | 0            |
| Reserved | pl_raw_set | fuf_raw_set | Reserved | acb_raw_set | sync_raw_set | recurrent_raster_done_raw_set | done_raw_set |
| R/W-0h   | R/W-0h     | R/W-0h      | R-0h     | R/W-0h      | R/W-0h       | R/W-0h                        | R/W-0h       |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-35. IRQSTATUS\_RAW Register Field Descriptions**

| Bit   | Field                         | Type | Reset | Description   |
|-------|-------------------------------|------|-------|---|
| 31-10 | Reserved                      | R/W  | 0h    |   |
| 9     | eof1_raw_set                  | R/W  | 0h    | DMA End-of-Frame 1 Raw Interrupt Status and Set Read indicates raw status 0 = Inactive 1 = Active Writing 1 will set status Writing 0 has no effect   |
| 8     | eof0_raw_set                  | R/W  | 0h    | DMA End-of-Frame 0 Raw Interrupt Status and Set Read indicates raw status 0 = inactive 1 = active Writing 1 will set status Writing 0 has no effect   |
| 7     | Reserved                      | R/W  | 0h    |   |
| 6     | pl_raw_set                    | R/W  | 0h    | DMA Palette Loaded Raw Interrupt Status and Set Read indicates raw status 0 = inactive 1 = active Writing 1 will set status Writing 0 has no effect   |
| 5     | fuf_raw_set                   | R/W  | 0h    | DMA FIFO Underflow Raw Interrupt Status and Set LCD dithering logic not supplying data to FIFO at a sufficient rate, FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO<br>Read indicates raw status 0 = Inactive 1 = Active Writing 1 will set status Writing 0 has no effect  |
| 4     | Reserved                      | R    | 0h    |   |
| 3     | acb_raw_set                   | R/W  | 0h    | For Passive Matrix Panels Only AC Bias Count Raw Interrupt Status and Set AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the acbi control bit-field<br>The counter is reloaded with the value in acbi but it is disabled until the user clears ABC<br>Read indicates raw status 0 = inactive 1 = active Writing 1 will set status Writing 0 has no effect |
| 2     | sync_raw_set                  | R/W  | 0h    | Frame Synchronization Lost Raw Interrupt Status and Set Read indicates raw status 0 = inactive 1 = active Writing 1 will set status Writing 0 has no effect   |
| 1     | recurrent_raster_done_raw_set | R/W  | 0h    | Raster Mode Frame Done Interrupt<br>Read indicates raw status<br>0 = Inactive<br>1 = Active<br>Writing 1 will set status<br>Writing 0 has no effect   |

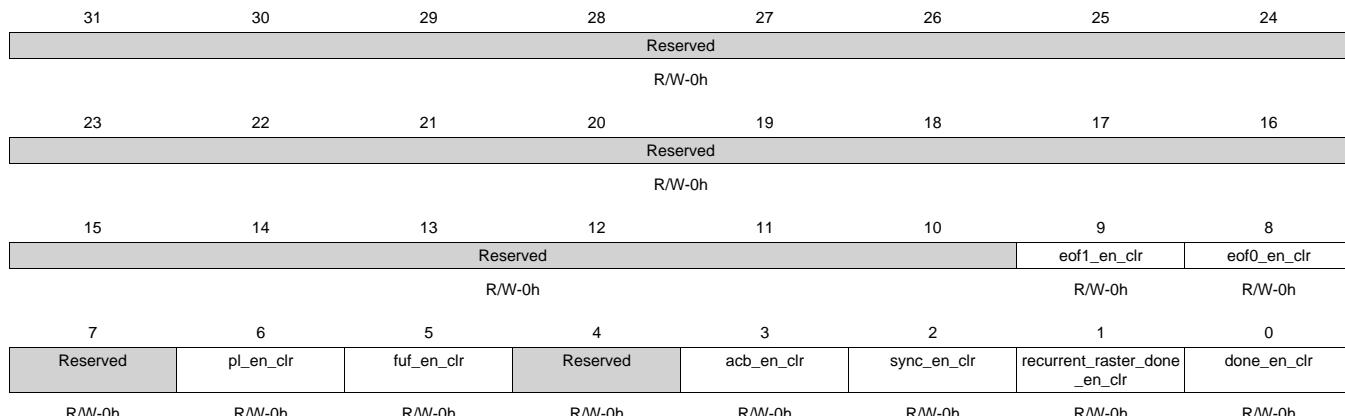
**Table 13-35. IRQSTATUS\_RAW Register Field Descriptions (continued)**

| Bit | Field        | Type | Reset | Description   |
|-----|--------------|------|-------|---|
| 0   | done_raw_set | R/W  | 0h    | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Raw Interrupt Status and Set Read indicates raw status 0 = inactive 1 = active Writing 1 will set status Writing 0 has no effect |

### 13.5.23 IRQSTATUS Register (offset = 5Ch) [reset = 0h]

IRQSTATUS is shown in [Figure 13-41](#) and described in [Table 13-36](#).

**Figure 13-41. IRQSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-36. IRQSTATUS Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-10 | Reserved    | R/W  | 0h    |  |
| 9     | eof1_en_clr | R/W  | 0h    | DMA End-of-Frame 1 Enabled Interrupt and Clear Read indicates enabled (masked) status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect   |
| 8     | eof0_en_clr | R/W  | 0h    | DMA End-of-Frame 0 Enabled Interrupt and Clear Read indicates enabled (masked) status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect   |
| 7     | Reserved    | R/W  | 0h    |  |
| 6     | pl_en_clr   | R/W  | 0h    | DMA Palette Loaded Enabled Interrupt and Clear Read indicates enabled (masked) status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect   |
| 5     | fuf_en_clr  | R/W  | 0h    | DMA FIFO Underflow Enabled Interrupt and Clear LCD dithering logic not supplying data to FIFO at a sufficient rate, FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO<br>Read indicates enabled (masked) status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect  |
| 4     | Reserved    | R/W  | 0h    |  |
| 3     | acb_en_clr  | R/W  | 0h    | For Passive Matrix Panels Only AC Bias Count Enabled Interrupt and Clear AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the acbi control bit-field<br>The counter is reloaded with the value in acbi but it is disabled until the user clears ABC<br>Read indicates enabled (masked) status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect |
| 2     | sync_en_clr | R/W  | 0h    | Frame Synchronization Lost Enabled Interrupt and Clear Read indicates enabled status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect  |

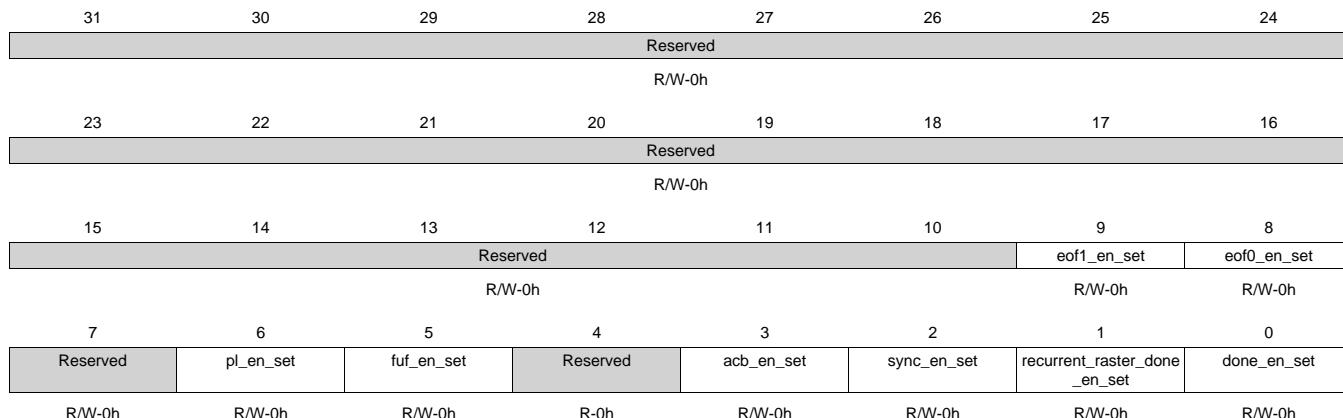
**Table 13-36. IRQSTATUS Register Field Descriptions (continued)**

| Bit | Field                        | Type | Reset | Description   |
|-----|------------------------------|------|-------|---|
| 1   | recurrent_raster_done_en_clr | R/W  | 0h    | Raster Frame Done Interrupt<br>Read indicates enabled status<br>0 = Inactive<br>1 = Active<br>Writing 1 will clear interrupt enable<br>Writing 0 has no effect  |
| 0   | done_en_clr                  | R/W  | 0h    | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Enabled Interrupt and Clear Read indicates enabled status 0 = inactive 1 = active Writing 1 will clear interrupt enable<br>Writing 0 has no effect |

### 13.5.24 IRQENABLE\_SET Register (offset = 60h) [reset = 0h]

IRQENABLE\_SET is shown in [Figure 13-42](#) and described in [Table 13-37](#).

**Figure 13-42. IRQENABLE\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-37. IRQENABLE\_SET Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-10 | Reserved    | R/W  | 0h    |   |
| 9     | eof1_en_set | R/W  | 0h    | DMA End-of-Frame 1 Interrupt Enable Set Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect   |
| 8     | eof0_en_set | R/W  | 0h    | DMA End-of-Frame 0 Interrupt Enable Set Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect   |
| 7     | Reserved    | R/W  | 0h    |   |
| 6     | pl_en_set   | R/W  | 0h    | DMA Palette Loaded Interrupt Enable Set Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect   |
| 5     | fuf_en_set  | R/W  | 0h    | DMA FIFO Underflow Interrupt Enable Set LCD dithering logic not supplying data to FIFO at a sufficient rate, FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO<br>Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect  |
| 4     | Reserved    | R    | 0h    |   |
| 3     | acb_en_set  | R/W  | 0h    | For Passive Matrix Panels Only AC Bias Count Interrupt Enable Set AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the acbi control bit-field<br>The counter is reloaded with the value in acbi but it is disabled until the user clears ABC<br>Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect |
| 2     | sync_en_set | R/W  | 0h    | Frame Synchronization Lost Interrupt Enable Set Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable Writing 0 has no effect   |

**Table 13-37. IRQENABLE\_SET Register Field Descriptions (continued)**

| Bit | Field                        | Type | Reset | Description   |
|-----|------------------------------|------|-------|---|
| 1   | recurrent_raster_done_en_set | R/W  | 0h    | Raster Done Interrupt Enable Set<br>Read indicates enabled (mask) status<br>0 = Disabled<br>1 = Enabled<br>Writing 1 will set interrupt enable<br>Writing 0 has no effect                                 |
| 0   | done_en_set                  | R/W  | 0h    | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Set Read indicates enabled (mask) status 0 = disabled 1 = enabled Writing 1 will set interrupt enable |

### 13.5.25 IRQENABLE\_CLEAR Register (offset = 64h) [reset = 0h]

IRQENABLE\_CLEAR is shown in [Figure 13-43](#) and described in [Table 13-38](#).

**Figure 13-43. IRQENABLE\_CLEAR Register**

|          |           |            |          |            |             |                              |             |
|----------|-----------|------------|----------|------------|-------------|------------------------------|-------------|
| 31       | 30        | 29         | 28       | 27         | 26          | 25                           | 24          |
| Reserved |           |            |          |            |             |                              |             |
| R/W-0h   |           |            |          |            |             |                              |             |
| 23       | 22        | 21         | 20       | 19         | 18          | 17                           | 16          |
| Reserved |           |            |          |            |             |                              |             |
| R/W-0h   |           |            |          |            |             |                              |             |
| 15       | 14        | 13         | 12       | 11         | 10          | 9                            | 8           |
| Reserved |           |            |          |            |             | eof1_en_clr                  | eof0_en_clr |
| R/W-0h   |           |            |          |            |             |                              |             |
| 7        | 6         | 5          | 4        | 3          | 2           | 1                            | 0           |
| Reserved | pl_en_clr | fuf_en_clr | Reserved | acb_en_clr | sync_en_clr | recurrent_raster_done_en_clr | done_en_clr |
| R/W-0h   | R/W-0h    | R/W-0h     | R/W-0h   | R/W-0h     | R/W-0h      | R/W-0h                       | R/W-0h      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-38. IRQENABLE\_CLEAR Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-10 | Reserved    | R/W  | 0h    |   |
| 9     | eof1_en_clr | R/W  | 0h    | DMA End-of-Frame 1 Interrupt Enable Clear Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect   |
| 8     | eof0_en_clr | R/W  | 0h    | DMA End-of-Frame 0 Interrupt Enable Clear Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect   |
| 7     | Reserved    | R/W  | 0h    |   |
| 6     | pl_en_clr   | R/W  | 0h    | DMA Palette Loaded Interrupt Enable Clear Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect   |
| 5     | fuf_en_clr  | R/W  | 0h    | DMA FIFO Underflow Interrupt Enable Clear LCD dithering logic not supplying data to FIFO at a sufficient rate, FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO<br>Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect  |
| 4     | Reserved    | R/W  | 0h    |   |
| 3     | acb_en_clr  | R/W  | 0h    | For Passive Matrix Panels Only AC Bias Count Interrupt Enable Clear AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the acbi control bit-field<br>The counter is reloaded with the value in acbi but it is disabled until the user clears ABC<br>Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect |
| 2     | sync_en_clr | R/W  | 0h    | Frame Synchronization Lost Interrupt Enable Clear Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable Writing 0 has no effect   |

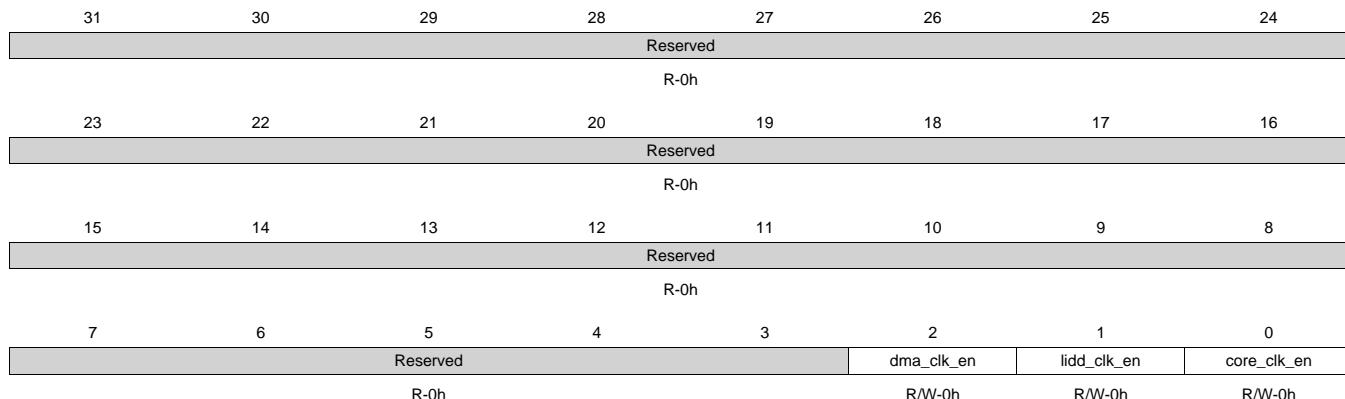
**Table 13-38. IRQENABLE\_CLEAR Register Field Descriptions (continued)**

| Bit | Field                        | Type | Reset | Description   |
|-----|------------------------------|------|-------|---|
| 1   | recurrent_raster_done_en_clr | R/W  | 0h    | Raster Done Interrupt Enable Clear<br>Read indicates enabled status<br>0 = Disabled<br>1 = Enabled<br>Writing 1 will clear interrupt enable<br>Writing 0 has no effect  |
| 0   | done_en_clr                  | R/W  | 0h    | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Clear<br>Read indicates enabled status<br>0 = disabled 1 = enabled Writing 1 will clear interrupt enable<br>Writing 0 has no effect |

### **13.5.26 CLKC\_ENABLE Register (offset = 6Ch) [reset = 0h]**

CLKC\_ENABLE is shown in Figure 13-44 and described in Table 13-39.

**Figure 13-44. CLKC\_ENABLE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

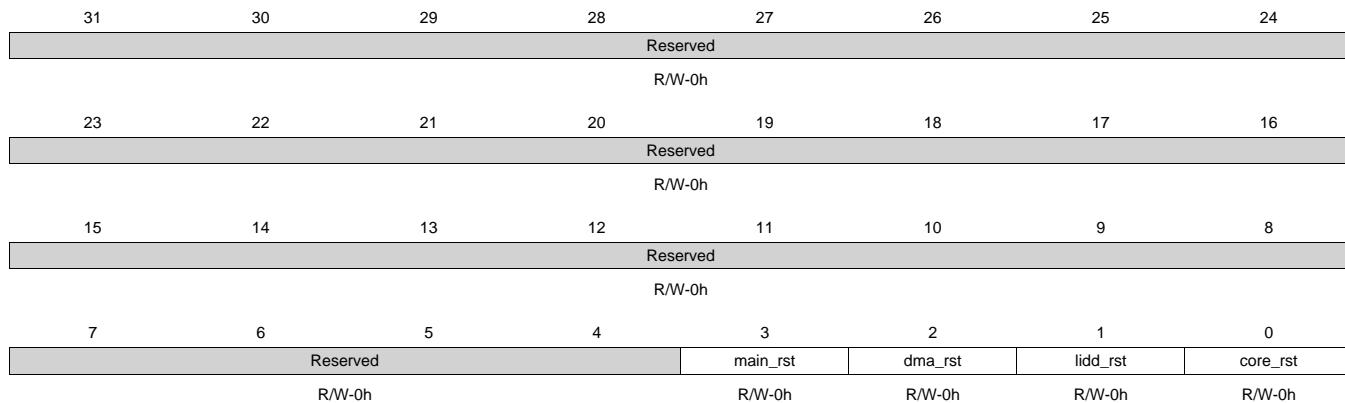
**Table 13-39. CLKC\_ENABLE Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description   |
|------|-------------|------|-------|---|
| 31-3 | Reserved    | R    | 0h    |   |
| 2    | dma_clk_en  | R/W  | 0h    | Software Clock Enable for the DMA submodule<br>The DMA submodule runs on the L3 Clock domain  |
| 1    | lidd_clk_en | R/W  | 0h    | Software Clock Enable for the LIDD submodule (character displays)<br>The LIDD submodule runs on the System Clock (lcd_clk) domain                             |
| 0    | core_clk_en | R/W  | 0h    | Software Clock Enable for the Core, which encompasses the Raster Active Matrix and Passive Matrix logic<br>The Core runs on the System Clock (lcd_clk) domain |

### 13.5.27 CLKC\_RESET Register (offset = 70h) [reset = 0h]

CLKC\_RESET is shown in Figure 13-45 and described in Table 13-40.

**Figure 13-45. CLKC\_RESET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 13-40. CLKC\_RESET Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-4 | Reserved | R/W  | 0h    |  |
| 3    | main_rst | R/W  | 0h    | Software Reset for the entire LCD module<br>1 = Reset Enable 0 = Reset Disable   |
| 2    | dma_rst  | R/W  | 0h    | Software Reset for the DMA submodule<br>1 = Reset Enable 0 = Reset Disable   |
| 1    | lidd_rst | R/W  | 0h    | Software Reset for the LIDD submodule (character displays)<br>1 = Reset Enable 0 = Reset Disable                                       |
| 0    | core_rst | R/W  | 0h    | Software Reset for the Core, which encompasses the Raster Active Matrix and Passive Matrix logic<br>1 = Reset Enable 0 = Reset Disable |

## Ethernet Subsystem

This chapter describes the ethernet subsystem of the device.

| Topic                                   | Page |
|---|------|
| 14.1 Introduction .....                 | 1156 |
| 14.2 Integration .....                  | 1158 |
| 14.3 Functional Description .....       | 1168 |
| 14.4 Software Operation .....           | 1228 |
| 14.5 Ethernet Subsystem Registers ..... | 1233 |

## 14.1 Introduction

Described in the following sections is the 3-port switch (3PSW) Ethernet subsystem. The 3-port switch gigabit ethernet subsystem provides ethernet packet communication and can be configured as an ethernet switch. It provides the gigabit media independent interface (GMII), reduced gigabit media independent interface (RGMII), reduced media independent interface (RMII), the management data input output (MDIO) for physical layer device (PHY) management.

### 14.1.1 Features

The general features of the ethernet switch subsystem are:

- Two 10/100/1000 Ethernet ports with GMII, RMII and RGMII interfaces
- Wire rate switching (802.1d)
- Non Blocking switch fabric
- Flexible logical FIFO based packet buffer structure
- Four priority level QOS support (802.1p)
- CPPI 3.1 compliant DMA controllers
- Support for IEEE 1588v2 Clock Synchronization (2008 Annex D and Annex F)
  - Timing FIFO and time stamping logic inside the SS
- Device Level Ring (DLR) Support
- Address Lookup Engine
  - 1024 addresses plus VLANs
  - Wire rate lookup
  - VLAN support
  - Host controlled time-based aging
  - Spanning tree support
  - L2 address lock and L2 filtering support
  - MAC authentication (802.1x)
  - Receive or destination based Multicast and Broadcast limits
  - MAC address blocking
  - Source port locking
  - OUI host accept/deny feature
- Flow Control Support (802.3x)
- EtherStats and 802.3Stats RMON statistics gathering (shared)
- Support for external packet dropping engine
- CPGMAC\_SL transmit to CPGMAC\_SL receive Loopback mode (digital loopback) supported
- CPGMAC\_SL receive to CPGMAC\_SL transmit Loobback mode (FIFO loopback) supported
- Maximum frame size 2016 bytes (2020 with VLAN)
- 8k (2048 x 32) internal CPPI buffer descriptor memory
- MDIO module for PHY Management
- Programmable interrupt control with selected interrupt pacing
- Emulation Support.
- Programmable transmit Inter-Packet Gap (IPG)
- Reset isolation

### **14.1.2 Unsupported Features**

There are 18 level interrupts from the CPGMAC module and 2 (used) level interrupts from the MDIO module. The CPSW\_3GSS includes an interrupt combiner/pacer to combine these interrupts together to produce 4 interrupt outputs (per processing core). This device does not split processing among multiple cores but allows servicing of the Core0 interrupts by the Cortex A8 or the PRU-ICSS.

The unsupported CPGMAC features in the device are shown in the following table.

**Table 14-1. Unsupported CPGMAC Features**

| Feature                       | Reason  |
|-------------------------------|---|
| Multi-core split processing   | Core 1 and Core 2 interrupts not connected.   |
| GMII                          | Only 4 Rx/Tx data pins are pinned out for each port. The device supports MII (on GMII interface), RGMII, and RMII interfaces only |
| Phy link status               | The MLINK inputs are not pinned out. Phy link status outputs can be connected to device GPIOs.                                    |
| Internal Delay mode for RGMII | RGMII Internal Delay mode is not supported.   |

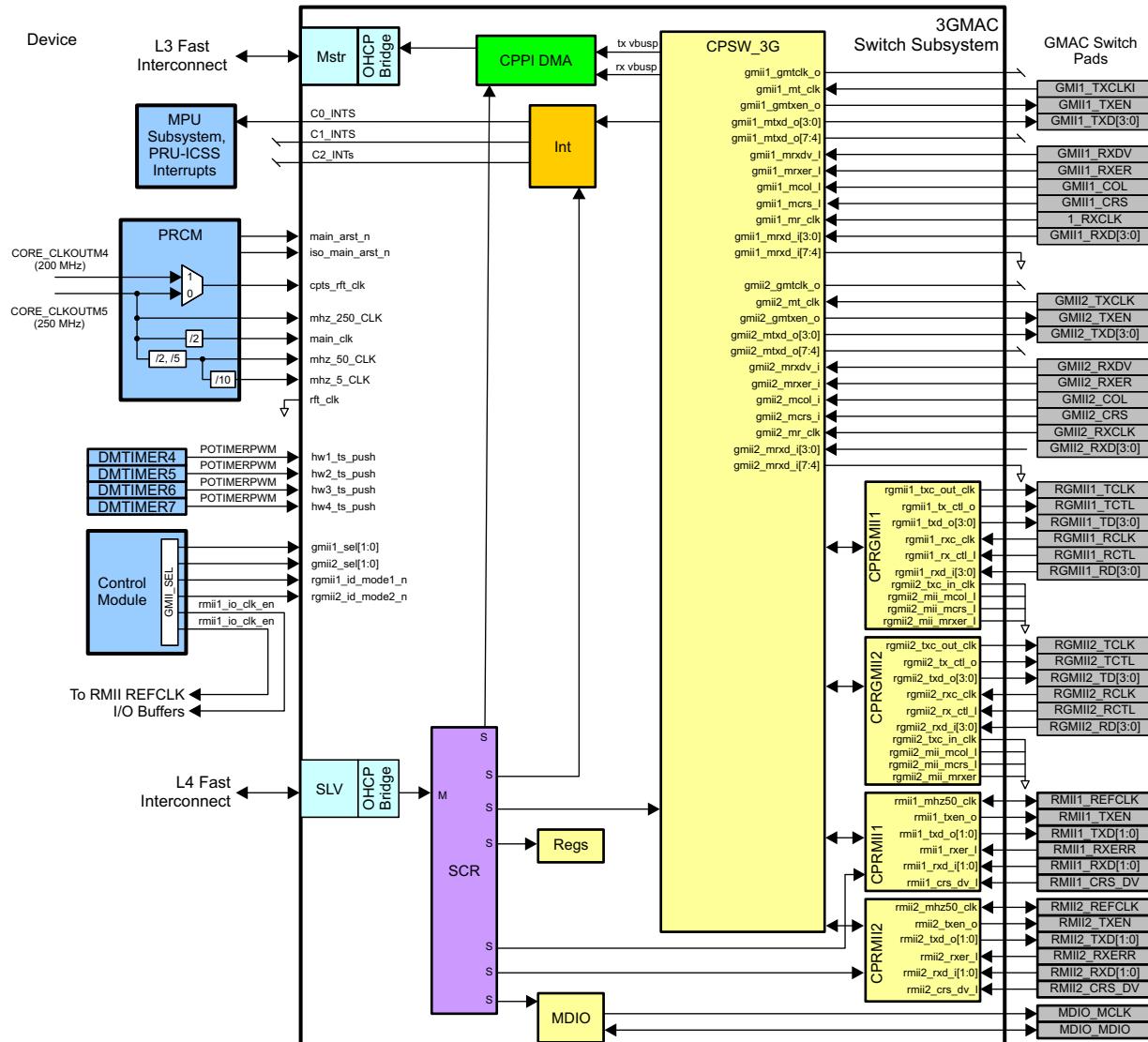
## 14.2 Integration

This device includes a single instantiation of the three-port Gigabit Ethernet Switch Subsystem (CPSW\_3GSS\_RG). The switch provides 2 external ethernet ports (ports 1 and 2) and an internal CPPI interface port (port 0) with IEEE 1588v2 and 802.1ae support. The subsystem consists of:

- One instance of the 3-port Gigabit switch CPSW-3G, which contains:
  - 2 CPGMAC\_SL 10/100/1000 ethernet port modules with GMII interface
- Two RGMII interface modules
- Two RMII interface modules
- One MDIO interface module
- One Interrupt Controller module
- Local CPPI memory of size 8K Bytes

The integration of the Ethernet Switch is shown in [Figure 14-1](#)

**Figure 14-1. Ethernet Switch Integration**



### 14.2.1 Ethernet Switch Connectivity Attributes

The general connectivity attributes for the Ethernet Switch module are shown in [Table 14-2](#).

**Table 14-2. Ethernet Switch Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain   |
| Clock Domain        | PD_PER_CPSW_125MHZ_GCLK (Main)<br>PD_PER_CPSW_250MHZ_GCLK (MHZ_250_CLK)<br>PD_PER_CPSW_50MHZ_GCLK (MHZ_50_CLK)<br>PD_PER_CPSW_5MHZ_GCLK (MHZ_5_CLK)<br>PD_PER_CPSW_CPTS_RFT_CLK (CPTS_RFT_CLK)  |
| Reset Signals       | CPSW_MAIN_ARST_N<br>CPSW_ISO_MAIN_ARST_N  |
| Idle/Wakeup Signals | Smart Idle  |
| Interrupt Requests  | 4 Interrupts<br>RX_THRESH (3PGSWRXTHR0) – Receive Threshold interrupt (nonpaced)<br>RX (3PGSWRXINT0) – Receive interrupt (paced)<br>TX (3PGSWTXINT0) – Transmit interrupt (paced)<br>Misc (3PGSWMISC0) – Other interrupts<br>All Ethernet Switch interrupts go to MPU Subsystem and PRU-ICSS.<br>The Subsystem contains 3 sets of interrupts C0, C1, and C2 to allow for split core processing of packets. On this device, only the C0 version of the interrupts is used. |
| DMA Requests        | None  |
| Physical Address    | L4 Fast slave port<br>L3 Fast initiator port  |

### 14.2.2 Ethernet Switch Clock and Reset Management

The ethernet switch controller operates in its own clock domain and its initiator and target interfaces are connected to the L3/L4 through asynchronous bridges. The OCP interfaces are driven by the MAIN clock input. Additional reference clock inputs are provided for operating the various ethernet ports at different rates.

**Table 14-3. Ethernet Switch Clock Signals**

| Clock Signal  | Max Freq | Reference / Source             | Comments                              |
|---|----------|--------------------------------|---------------------------------------|
| rft_clk<br>Gigabit GMII Tx Reference clock          | 125 MHz  | Tied low                       | not supported                         |
| main_clk<br>Logic/Interface clock                   | 125 MHz  | CORE_CLKOUTM5 / 2              | pd_per_cpsw_125mhz_gclk from PRCM     |
| mhz250_clk<br>Gigabit RGMII Reference clock         | 250 MHz  | CORE_CLKOUTM5                  | pd_per_cpsw_250mhz_gclk from PRCM     |
| mhz50_clk<br>RMII and 100mbps RGMII Reference clock | 50 MHz   | CORE_CLKOUTM5 / 5              | pd_per_cpsw_50mhz_gclk from PRCM      |
| mhz5_clk<br>10 mbpsRGMII Reference clock            | 5 MHz    | CORE_CLKOUTM5 / 50             | pd_per_cpsw_5mhz_gclk from PRCM       |
| cpts_rft_clk<br>IEEE 1588v2 clock                   | 250 MHz  | CORE_CLKOUTM4<br>CORE_CLKOUTM5 | pd_per_cpsw_cpts_rft_clk from PRCM    |
| gmii1_mr_clk<br>GMII Port 1 Receive clock           | 25 MHz   | External Pin                   | gmii1_rxclk_in from GMII1_RCLK pad    |
| gmii2_mr_clk<br>GMII Port 2 Receive clock           | 25 MHz   | External Pin                   | gmii2_rxclk_in from GMII2_RCLK pad    |
| gmii1_mt_clk<br>GMII Port 1 Transmit clock          | 25 MHz   | External Pin                   | gmii1_txclk_in from GMII1_TCLK pad    |
| gmii2_mt_clk<br>GMII Port 2 Transmit clock          | 25 MHz   | External Pin                   | gmii2_txclk_in from GMII2_TCLK pad    |
| rgmii1_rxc_clk<br>RGMII Port 1 Receive clock        | 250 MHz  | External Pin                   | rgmii1_rclk_in from RGMII1_RCLK pad   |
| rgmii2_rxc_clk<br>RGMII Port 2 Receive clock        | 250 MHz  | External Pin                   | rgmii2_rclk_in from RGMII2_RCLK pad   |
| rmii1_mhz_50_clk<br>RMII Port 1 Reference clock     | 50 MHz   | External Pin                   | rmii1_refclk_in from RMII1_REFCLK pad |
| rmii2_mhz_50_clk<br>RMII Port 2 Reference clock     | 50 MHz   | External Pin                   | rmii2_refclk_in from RMII2_REFCLK pad |

### 14.2.3 Ethernet Switch Pin List

The external signals for the Ethernet Switch module are shown in the following table.

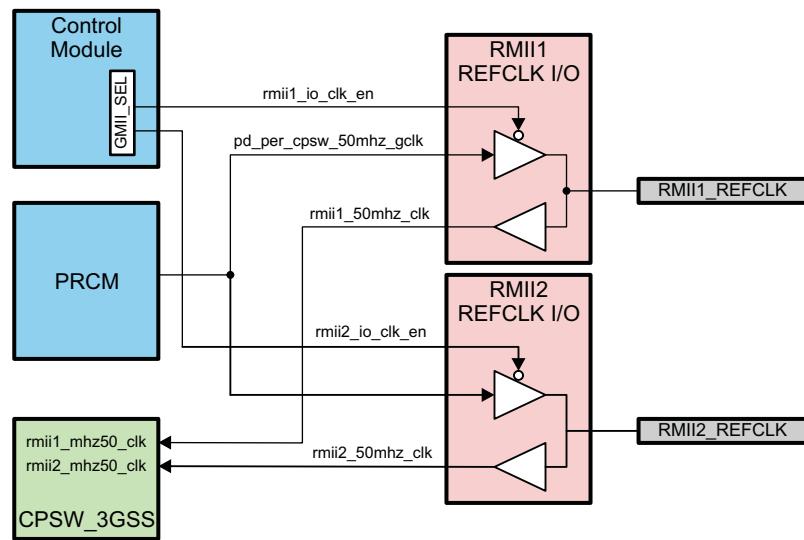
**Table 14-4. Ethernet Switch Pin List**

| Pin            | Type* | Description                     |
|----------------|-------|---------------------------------|
| GMIIx_RXCLK    | I     | GMII/MII Receive clock          |
| GMIIx_RXD[7:0] | I     | GMII/MII Receive data           |
| GMIIx_RXDV     | I     | GMII/MII Receive data valid     |
| GMIIx_RXER     | I     | GMII/MII Receive error          |
| GMIIx_COL      | I     | GMII/MII Collision detect       |
| GMIIx_CRS      | I     | GMII/MII Carrier sense          |
| GMIIx_TXCLK    | I     | GMII/MII Transmit clock         |
| GMII_TXD[7:0]  | O     | GMII/MII Transmit data          |
| GMIIx_TXEN     | O     | GMII/MII Transmit enable        |
| RGMIIx_RCLK    | I     | RGMII Receive clock             |
| RGMIIx_RCTL    | I     | RGMII Receive control           |
| RGMIIx_RD[3:0] | I     | RGMII Receive data              |
| RGMIIx_TCLK    | O     | RGMII Transmit clock            |
| RGMIIx_TCTL    | O     | RGMII Transmit control          |
| RGMIIx_TD[3:0] | O     | RGMII Transmit data             |
| RMIIx_RXD[1:0] | I     | RMII Receiver data              |
| RMIIx_RXER     | I     | RMII Receiver error             |
| RMIIx_CRS_DV   | I     | RMII Carrier sense / Data valid |
| RMIIx_TXEN     | O     | RMII Transmit enable            |
| RMIIx_REFCLK   | I/O   | RMII Reference clock            |
| RMIIx_RXD[1:0] | O     | RMII Transmit data              |
| MDIO_MCLK      | O     | MDIO Serial clock               |
| MDIO_MDIO      | I/O   | MDIO Serial data                |

### 14.2.4 Ethernet Switch RMII Clocking Details

The RMII interface reference clock pin can operate as either an input or output. When used as an input, the clock is driven by the I/O pad. When used as an output, the pad and the RMII clock input are driven by the internal 50-MHz reference clock from the PRCM. The operation is controlled by the GMII\_SEL[RMIIx\_IO\_CLK\_EN] fields in the Control Module, as shown in [Figure 14-2](#).

**Figure 14-2. Ethernet Switch RMII Clock Detail**



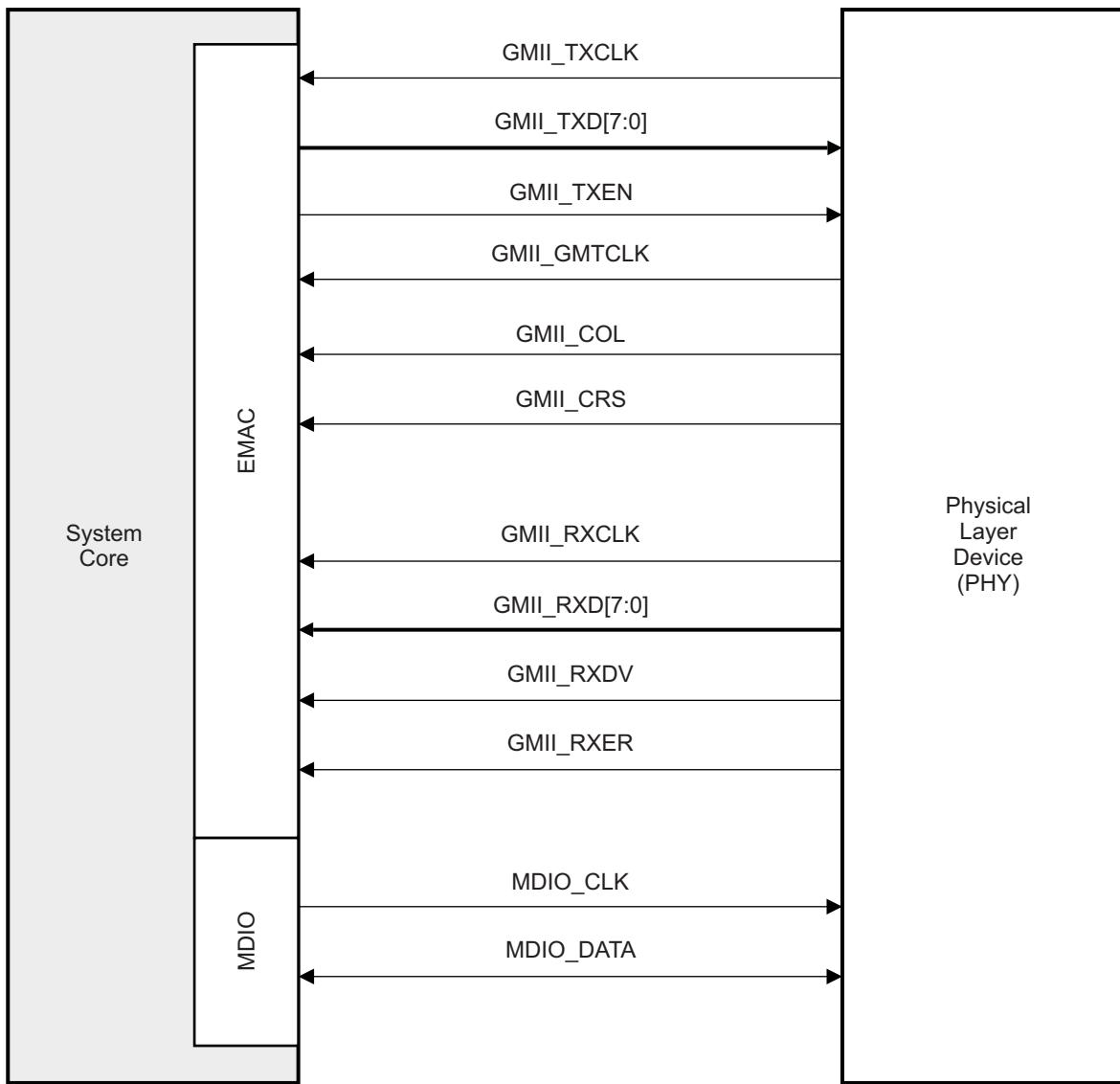
#### 14.2.5 GMII Interface Signal Connections and Descriptions

GMII Interface can operate in GIG/MII Modes.

In GIG(1000Mbps) Mode 3PSW operates only in Full duplex Mode.

In MII Mode(100/10 Mbps) 3PSW operates in Full duplex and Half Duplex.

The pin connections of the GMII Interface are shown in [Figure 14-3](#).

**Figure 14-3. MII Interface Connections**


The detailed description of the signals in GIG/MII Mode are explained in the following sections.

#### 14.2.5.1 GMII Interface Signal Descriptions in GIG (1000Mbps) Mode

**Table 14-5. GMII Interface Signal Descriptions in GIG (1000Mbps) Mode**

| Signal | Type | Description  |
|--------|------|--|
| GMTCLK | O    | The transmit clock is a continuous clock that provides the timing reference for transmit operations. The clock is generated by the CPSW and is 125 MHz at 1000 Mbps operation.                               |
| MtCLK  | I    | The transmit clock is a continuous free running clock and is generated by the PHY.   |
| Mtxd   | O    | The transmit data pins are a collection of 8 data signals comprising 8 bits of data. MTXDO is the least-significant bit (LSB). The signals are synchronized by GMTCLK and valid only when MTXEN is asserted. |
| Mtxen  | O    | The transmit enable signal indicates that the MTXD pins are generating 8bit data for use by the PHY. It is driven synchronously by GMTCLK.   |

**Table 14-5. GMII Interface Signal Descriptions in GIG (1000Mbps) Mode (continued)**

| Signal | Type | Description   |
|--------|------|---|
| MCOL   | I    | In full-duplex operation, the MCOL pin is used for hardware transmit flow control. Asserting the MCOL pin will stop packet transmissions; packets in the process of being transmitted when MCOL is asserted will complete transmission. The MCOL pin should be held low if hardware transmit flow control is not used.                    |
| MCRS   | I    | In full-duplex operation, the MCRS pin should be held low.  |
| MRCLK  | I    | The receive clock is a continuous clock that provides the timing reference for receive operations. The MRXD, MRXDV, and MRXER signals are tied to this clock. The clock is generated by the PHY and is 125 MHz at 1000Mbps of operation.  |
| MRXD   | I    | The receive data pins are a collection of 8 data signals comprising 8 bits of data. MRXD0 is the least-significant bit (LSB). The signals are synchronized by MRCLK and valid only when MRXDV is asserted.  |
| MRXDV  | I    | The receive data valid signal indicates that the MRXD pins are generating byte data for use by the 3PSW. It is driven synchronously to MRCLK.   |
| MDCLK  | O    | Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.  |
| MDIO   | I/O  | MDIO DATA(MDIO_D). MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication, PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations. |

#### 14.2.5.2

**Table 14-6. GMII Interface Signal Descriptions in MII (100/10Mbps) Mode**

| Signal | Type | Description   |
|--------|------|---|
| GMTCLK | O    | The clock is generated by the CPSW and is running at 125 MHz  |
| MtCLK  | I    | The transmit clock is a continuous clock that provides the timing reference for transmit operations. The MTXD and MTXEN signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.   |
| Mtxd   | O    | MTXD[7-4] pins of MTXD data are not used. The transmit data pins are a collection of 4 data signals MTXD[3-0] comprising 4 bits of data. MTXD0 is the least-significant bit (LSB). The signals are synchronized by MTCLK and valid only when MTXEN is asserted.   |
| Mtxen  | O    | The transmit enable signal indicates that the MTXD pins are generating 4bit data for use by the PHY. It is driven synchronously by MTCLK.   |
| MCOL   | I    | In half-duplex operation, the MCOL pin is asserted by the PHY when it detects a collision on the network. It remains asserted while the collision condition persists. This signal is not necessarily synchronous to MTCLK nor MRCLK.  |
|        | I    | In full-duplex operation, the MCOL pin is used for hardware transmit flow control. Asserting the MCOL pin will stop packet transmissions; packets in the process of being transmitted when MCOL is asserted will complete transmission. The MCOL pin should be held low if hardware transmit flow control is not used |
| MCRS   | I    | In half-duplex operation, the MCRS pin is asserted by the PHY when the network is not idle in either transmit or receive. The pin is deasserted when both transmit and receive are idle. This signal is not necessarily synchronous to MTCLK nor MRCLK.   |
|        | I    | In full-duplex operation, the MCRS pin should be held low.  |
| MRCLK  | I    | The receive clock is a continuous clock that provides the timing reference for receive operations. The MRXD, MRXDV, and MRXER signals are tied to this clock. The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation.   |
| MRXD   | I    | MRXD[7-4] bits are unused in the MII mode. The receive data pins are a collection of 4 data signals comprising 4 bits of data. MRXD0 is the least-significant bit (LSB). The signals are synchronized by MRCLK and valid only when MRXDV is asserted.   |

**Table 14-6. GMII Interface Signal Descriptions in MII (100/10Mbps) Mode (continued)**

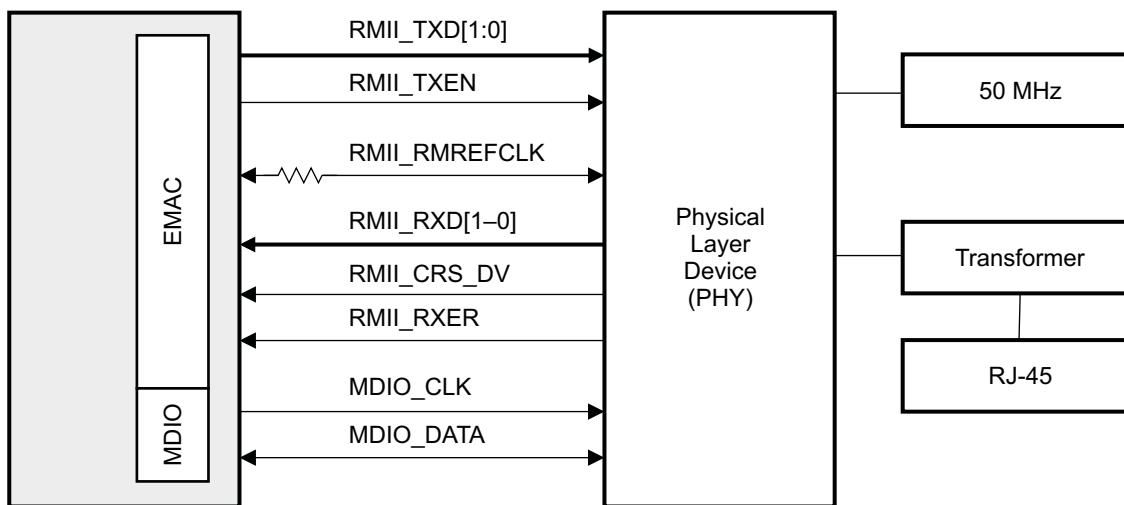
| Signal | Type | Description  |
|--------|------|--|
| MRXDV  | I    | The receive data valid signal indicates that the MRXD pins are generating nibble data for use by the 3PSW. It is driven synchronously to MRCLK.  |
| MDCLK  | O    | Management data clock (MDIO_CLK). The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.   |
| MDIO   | I/O  | MDIO DATA(MDIO_D). MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication,PHY address, register address, and data bit cycles. The MDIO_D pin acts as an output for all but the data bit cycles at which time it is an input for read operations. |

#### 14.2.6 RMII Signal Connections and Descriptions

Figure 14-4 shows a device with integrated 3PSW and MDIO interfaced via a RMII connection in a typical system.

The individual CPSW and MDIO signals for the RMII interface are summarized in Table 14-7.

For more information, see either the IEEE 802.3 standard or ISO/IEC 8802-3:2000(E).

**Figure 14-4. RMII Interface Connections**

**Table 14-7. RMII Interface Signal Descriptions**

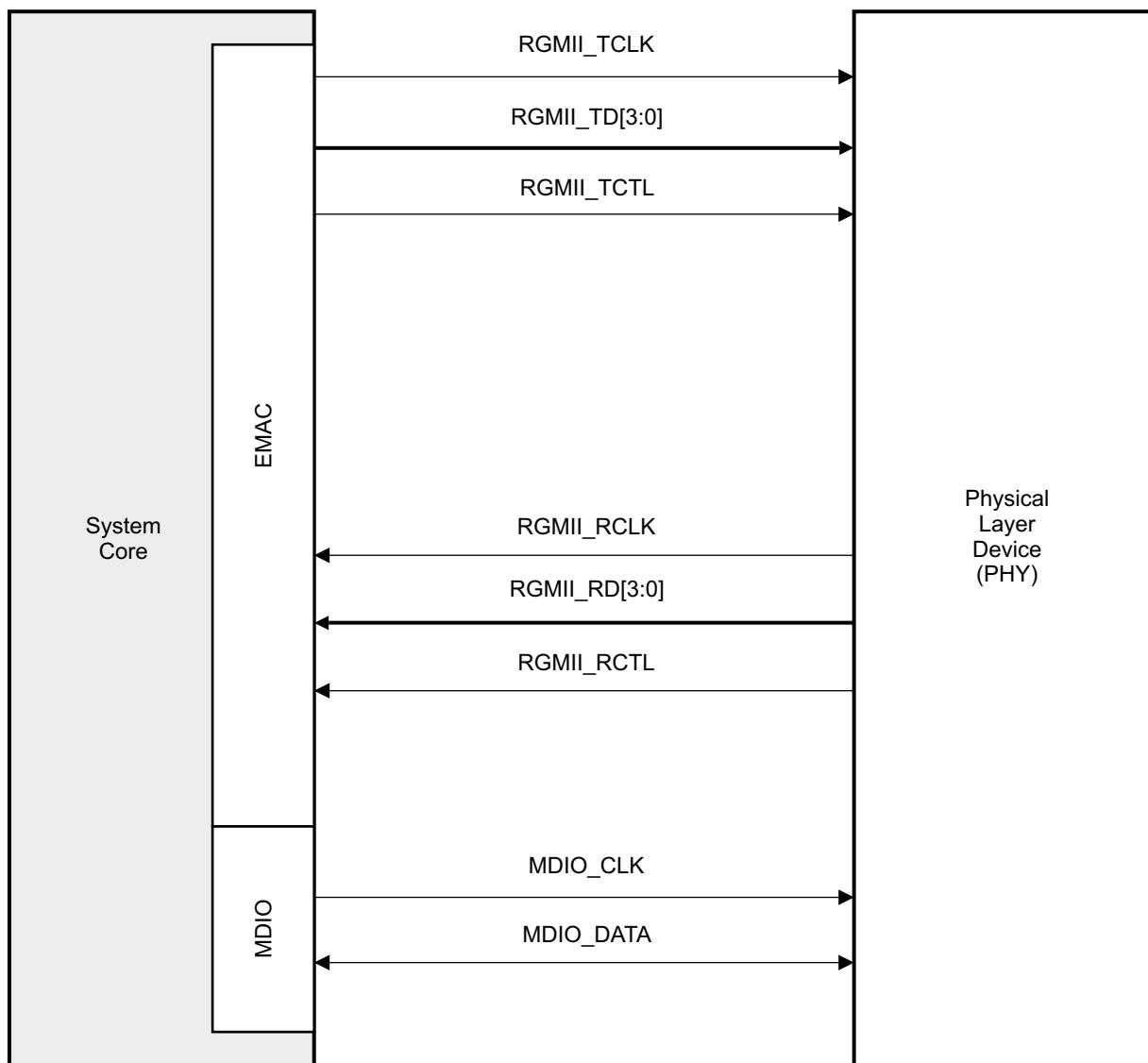
| Signal        | Type | Description  |
|---------------|------|--|
| RMII_TXD[1-0] | O    | Transmit data. The transmit data pins are a collection of 2 bits of data. RMII_TXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_REFCLK and valid only when RMII_TXEN is asserted.                             |
| RMII_TXEN     | O    | Transmit enable. The transmit enable signal indicates that the RMII_TXD pins are generating data for use by the PHY. RMII_TXEN is synchronous to RMII_MHZ_50_CLK.  |
| RMII_REFCLK   | I/O  | RMII reference clock.  |
| RMII_RXD[1-0] | I    | Receive data. The receive data pins are a collection of 2 bits of data. RMII_RXD0 is the least-significant bit (LSB). The signals are synchronized by RMII_REFCLK and valid only when RMII_CRS_DV is asserted and RMII_RXER is deasserted. |
| RMII_CRS_DV   | I    | Carrier sense/receive data valid. Multiplexed signal between carrier sense and receive data valid.   |

**Table 14-7. RMII Interface Signal Descriptions (continued)**

| Signal    | Type | Description  |
|-----------|------|--|
| RMII_RXER | I    | Receive error. The receive error signal is asserted to indicate that an error was detected in the received frame.  |
| MDIO_CLK  | O    | Management data clock. The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.  |
| MDIO_DATA | I/O  | MDIO DATA. MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication,PHY address, register address, and data bit cycles. The MDIO_DATA signal acts as an output for all but the data bit cycles at which time it is an input for read operations. |

#### 14.2.7 RGMII Signal Connections and Descriptions

Figure 14-5 shows a device with integrated CPSW and MDIO interfaced via a RGMII connection in a typical system.

**Figure 14-5. RGMII Interface Connections**

**Table 14-8. RGMII Interface Signal Descriptions**

| <b>Signal</b> | <b>Type</b> | <b>Description</b>  |
|---------------|-------------|---|
| RGMII_TD[3-0] | O           | The transmit data pins are a collection of 4 bits of data. RGMII_RD0 is the least-significant bit (LSB).<br>The signals are valid only when RGMII_TCTL is asserted.   |
| RGMII_TCTL    | O           | Transmit Control/enable .The transmit enable signal indicates that the RGMII_TD pins are generating data for use by the PHY.  |
| RGMII_TCLK    | O           | The transmit reference clock will be 125Mhz, 25Mhz, or 2.5Mhz depending on speed of operation.  |
| RGMII_RD[3-0] | I           | The receive data pins are a collection of 4 bits of data. RGMII_RD is the least-significant bit (LSB).<br>The signals are valid only when RGMII_RCTL is asserted.   |
| RGMII_RCTL    | I           | The receive data valid/control signal indicates that the RGMII_RD pins are nibble data for use by the 3PSW.   |
| RGMII_RCLK    | I           | The receive clock is a continuous clock that provides the timing reference for receive operations.The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation,125 MHz at 1000Mbps of operation.  |
| MDIO_CLK      | O           | Management data clock. The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO pin.   |
| MDIO_DATA     | I/O         | MDIO DATA. MDIO data pin drives PHY management data into and out of the PHY by way of an access frame consisting of start of frame, read/write indication,PHY address, register address, and data bit cycles. The MDIO_DATA pin acts as an output for all but the data bit cycles at which time it is an input for read operations. |

## 14.3 Functional Description

The 3 port switch (3PSW) Ethernet Subsystem peripheral are compliant to the IEEE Std 802.3 Specification. The 3PSW Ethernet Subsystem contains two RGMII/RMII interfaces, one CPPI 3.0 interface, Interrupt Controller, MDIO and CPSW\_3G which contains two GMII interfaces as shown in [Figure 14-6](#).

The subsystem modules are discussed in detail in the following sections.

### 14.3.1 CPSW\_3G Subsystem

#### 14.3.1.1 Interrupt Pacing

The receive and transmit pulse interrupts can be paced. The receive threshold and miscellaneous interrupts are not paced. The Interrupt pacing feature limits the number of interrupts that occur during a given period of time. For heavily loaded systems in which interrupts can occur at a very high rate (e.g. 148,800 packets per second for Ethernet), the performance benefit is significant due to minimizing the overhead associated with servicing each interrupt. Interrupt pacing increases the CPU cache hit ratio by minimizing the number of times that large interrupt service routines are moved to and from the CPU instruction cache.

Each CPU receive and transmit pulse interrupt contains an interrupt pacing sub-block (six total). Each sub-block is disabled by default allowing the selected interrupt inputs to pass through unaffected. The interrupt pacing module counts the number of interrupts that occur over a 1ms interval of time. At the end of each 1ms interval, the current number of interrupts is compared with a target number of interrupts (specified by the associated maximum number of interrupts register).

Based on the results of the comparison, the length of time during which interrupts are blocked is dynamically adjusted. The 1ms interval is derived from a 4us pulse that is created from a prescale counter whose value is set in the int\_prescale value in the Int\_Control register. The int\_prescale value should be written with the number of VBUSP\_CLK periods in 4us. The pacing timer determines the interval during which interrupts are blocked and decrements every 4us. It is reloaded each time a zero count is reached. The value loaded into the pacing timer is calculated by hardware every 1ms according to the following algorithm:

```
if (intr_count > 2*intr_max)
pace_timer = 255;
else if (intr_count > 1.5*intr_max)
pace_timer = last_pace_timer*2 + 1;
else if (intr_count > 1.0*intr_max)
pace_timer = last_pace_timer + 1;
else if (intr_count > 0.5*intr_max)
pace_timer = last_pace_timer - 1;
else if (intr_count != 0)
pace_timer = last_pace_timer/2;
else
pace_timer = 0;
```

If the rate of interrupt inputs is much less than the target interrupt rate specified in the associated maximum interrupts register, then the interrupt is not blocked. If the interrupt rate is greater than the target rate, the interrupt will be “paced” at the rate specified in the interrupt maximum register. The interrupt maximum register should be written with a value between 2 and 63 inclusive indicating the target number of interrupts per milli-second.

#### 14.3.1.2 Reset Isolation

Reset isolation for the Ethernet switch on Device is that the switch function of the ethernet IP remains active even in case of all device resets except for POR pin reset and ICEPICK COLD reset. Packet traffic to/from the 3PSW host will be flushed/dropped, but the ethernet switch will remain operational for all traffic between external devices on the switch even though the device is under-going a device reset. Pin mux configuration for ethernet related IO and reference clocks needed by the Ethernet switch IP to be active is controlled by a protected control module bit. If reset isolation is enabled, then only a POR pin or ICEPICK COLD reset event should fully reset the Ethernet switch IP including the actual switch and also the reference clocks and pin mux control specifically associated with the Ethernet IP.

#### 14.3.1.2.1 Modes of Operation

The device has two modes of operation concerning the reset of the 3PSW Ethernet switch.

The mode is controlled by the ISO\_CONTROL bit in **RESET\_ISO** register of the **device control module**. This bit should default to '0'. Writes to the ISO\_CONTROL bit must be supervisor mode writes.

Mode 1: ISO\_CONTROL=0 (reset isolation disabled)

- This mode is selected when ISO\_CONTROL bit of control module is = 0. This should be the default state of the bit after control module reset.
- Upon any device level resets, the entire CPSW\_3GSS\_R IP, L3/L4, control module (including all pin mux control and the ISO\_CONTROL bit) is immediately reset.

Mode 2: ISO\_CONTROL=1 (reset isolation enabled)

- This mode is selected when ISO\_CONTROL bit of control module is = 1.
- Upon any device reset source other than POR pin or ICEPICK cold (so this includes SW global cold, any watchdog reset, warm RESETn pin, ICEPICK warm, SW global warm), the following should be true:
  - The CPSW\_3GSS\_R is put into 'isolate' mode and non-switch related portions of the IP are reset.
  - The 50-MHz and 125-MHz reference clocks to the 3PSW Ethernet Subsystem remains active throughout the entire reset condition.
  - The control for pin multiplexing for all of the signals should maintain their current configuration throughout the entire reset condition.
  - The reset isolated logic inside 3PSW Ethernet Subsystem IP which maintains the switch functionality
  -
- Upon any cold reset sources, the entire 3PSW Ethernet Subsystem, control module (including all pin mux control and the ISO\_CONTROL bit itself) is reset.

### 14.3.1.3 Interrupts

The 3 Port Switch Ethernet Subsystem generates four interrupt events.

#### 14.3.1.3.1 Receive Packet Completion Pulse Interrupt (RX\_PULSE)

The RX\_PULSE interrupt is a paced pulse interrupt selected from the 3PSW RX\_PEND [7:0] interrupts. The receive DMA controller has eight channels with each channel having a corresponding (RX\_PEND[7:0]).

The following steps will enable the receive packet completion interrupt.

- Enable the required channel interrupts of the DMA engine by setting 1 to the appropriate bit in the RX\_INTMASK\_SET register.
- The receive completion interrupt(s) to be routed to RX\_PULSE is selected by setting one or more bits in the receive interrupt enable register Cn\_RX\_EN. The masked interrupt status can be read in the Receive Interrupt Masked Interrupt Status (Cn\_RX\_STAT) register.

When the 3PSW completes a packet reception, the subsystem issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon interrupt reception, the CPU processes one or more packets from the buffer chain and then acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer (RXn\_CP) in the receive DMA state RAM.

Upon reception of an interrupt, software should perform the following:

- Read the RX\_STAT register to determine which channel(s) caused the interrupt.
- Process received packets for the interrupting channel(s).
- Write the 3PSW completion pointer(s) (RXn\_CP). The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the subsystem (address of last buffer descriptor used by the subsystem). If the two values are not equal (which means that the 3PSW has received more packets than the CPU has processed), the receive packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the system has received), the pending interrupt is de-asserted. The value that the 3PSW is expecting is found by reading the receive channel*n* completion pointer register (RXn\_CP).
- Write the value 1h to the CPDMA\_EOI\_VECTOR register.

To disable the interrupt:

- The eight channel interrupts may be individually disabled by writing to 1 the appropriate bit in the RX\_INTMASK\_CLEAR
- The receive completion pulse interrupt could be disabled by clearing to 0 all the bits of the RX\_EN.

The software could still poll for the RX\_INTSTAT\_RAW and RX\_INTSTAT\_MASKED registers if the corresponding interrupts are enabled.

#### 14.3.1.3.2 Transmit Packet Completion Pulse Interrupt (TX\_PULSE)

The TX\_PULSE interrupt is a paced pulse interrupt selected from the 3PSW TX\_PEND [7:0] interrupts. The transmit DMA controller has eight channels with each channel having a corresponding (TX\_PEND[7:0]).

To enable the transmit packet completion interrupt:

- Enable the required channel interrupts of the DMA engine by setting 1 to the appropriate bit in the TX\_INTMASK\_SET register.
- The transmit completion interrupt(s) to be routed to TX\_PULSE is selected by setting one or more bits in the transmit interrupt enable register Cn\_TX\_EN .The masked interrupt status can be read in the Transmit Interrupt Masked Interrupt Status (Cn\_TX\_STAT) register.

When the 3PSW completes the transmission of a packet, the 3PSW subsystem issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's transmit completion pointer located in the state RAM block. The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written.

Upon receiving an interrupt, software should perform the following:

- Read the TX\_STAT register to determine which channel(s) caused the interrupt
- Process received packets for the interrupting channel(s).
- Write the 3PSW completion pointer(s) (TX<sub>n</sub>\_CP). The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the 3PSW (address of last buffer descriptor used by the 3PSW). If the two values are not equal (which means that the 3PSW has transmitted more packets than the CPU has processed), the transmit packet completion interrupt signal remains asserted. If the two values are equal (which means that the host has processed all packets that the subsystem has transferred), the pending interrupt is cleared. The value that the 3PSW is expecting is found by reading the transmit channel<sub>n</sub> completion pointer register (TX<sub>n</sub>\_CP).
- Write the 2h to the CPDMA\_EOI\_VECTOR register.

To disable the interrupt:

- The eight channel interrupts may be individually disabled by writing to 1 the appropriate bit in the TX\_INTMASK\_CLEAR.
- The receive completion pulse interrupt could be disabled by clearing to 0 all the bits of the TX\_EN. The software could still poll for the TX\_INTSTAT\_RAW and TX\_INTSTAT\_MASKED registers if the corresponding interrupts are enabled.

#### **14.3.1.3.3 Receive Threshold Pulse Interrupt (RX\_THRESH\_PULSE)**

The RX\_THRESH\_PULSE interrupt is an immediate (non-paced) pulse interrupt selected from the CPSW\_3G RX\_THRESH\_PEND[7:0] interrupts. The receive DMA controller has eight channels with each channel having a corresponding threshold pulse interrupt (RX\_THRESH\_PEND [7:0]).

To enable the receive threshold pulse Interrupt:

- Enable the required channel interrupts of the DMA engine by setting 1 to the appropriate bit in the RX\_INTMASK\_SET register.
- The receive threshold interrupt(s) to be routed to RX\_THRESH\_PULSE is selected by setting one or more bits in the receive threshold interrupt enable register RX\_THRESH\_EN. The masked interrupt status can be read in the Receive Threshold Masked Interrupt Status (C<sub>n</sub>\_RX\_THRESH\_STAT) register.

The RX\_THRESH\_PULSE is asserted when enabled when the channel's associated free buffer count RX<sub>n</sub>\_FREEBUFFER is less than or equal to the corresponding RX<sub>n</sub>\_PENDTHRESH register.

Upon receiving an interrupt, software should perform the following:

- Read the C<sub>n</sub>\_RX\_THRESH\_STAT bit address location to determine which channel(s) caused the interrupt.
- Process the received packets in order to add more buffers to any channel that is below the threshold value.
- Write the CPSW\_3G completion pointer(s).
- Write the value 0h to the CPDMA\_EOI\_VECTOR register.

The threshold pulse interrupt is an immediate interrupt intended to indicate that software should immediately process packets to preclude an overrun condition from occurring for the particular channels.

To disable the interrupt:

- The eight channel receive threshold interrupts may be individually disabled by writing to 1 the appropriate bit in the RX\_INTMASK\_CLEAR register.
- The receive threshold pulse interrupt could be disabled by clearing to Zero the corresponding bits of the RX\_THRESH\_EN. The software could still poll for the RX\_INTSTAT\_RAW and INTSTAT\_MASKED registers if the corresponding interrupts are enabled.

#### 14.3.1.3.4 Miscellaneous Pulse Interrupt (MISC\_PULSE)

The MISC\_PULSE interrupt is an immediate, non-paced, pulse interrupt selected from the miscellaneous interrupts (EVNT\_PEND, STAT\_PEND, HOST\_PEND, MDIO\_LINKINT, MDIO\_USERINT). The miscellaneous interrupt(s) is selected by setting one or more bits in the  $C_n\text{-MISC\_EN}[4:0]$  register. The masked interrupt status can be read in the  $C_n\text{-MISC\_STAT}[4:0]$  register. Upon reception of an interrupt, software should perform the following:

- Read the Misc\_Stat[4:0] register to determine which caused the interrupt.
- Process the interrupt.
- Write the appropriate value (0x3) to the CPDMA\_EOI\_VECTOR register
- MDIO\_LINTINT[1] and MDIO\_USERINT[1] are unused.

##### 14.3.1.3.4.1 EVNT\_PEND (CPTS\_PEND) Interrupt

See [Section 14.3.7, Common Platform Time Sync \(CPTS\)](#), for information on the time sync event interrupt.

##### 14.3.1.3.4.2 Statistics Interrupt

The statistics level interrupt (STAT\_PEND) will be asserted if enabled when any statistics value is greater than or equal to 0x80000000. The statistics interrupt is cleared by writing to decrement all statistics values greater than 0x80000000 (such that their new values are less than 0x80000000). The statistics interrupt is enabled by setting to one the appropriate bit in the INTMASK\_SET register in the CPDMA submodule.

The statistics interrupt is disabled by writing one to the appropriate bit in the INTMASK\_CLEAR register. The raw and masked statistics interrupt status may be read by reading the TX\_IntStat\_Raw and TX\_IntStat\_Masked registers, respectively

##### 14.3.1.3.4.3 Host Error Interrupt

The host error interrupt (HOST\_PEND) will be asserted if enabled when a host error is detected during transmit or receive CPDMA transactions. The host error interrupt is intended for software debug, and is cleared by a warm reset or a system reset. The raw and masked statistics interrupt status can be read by reading the TX\_INTSTAT\_RAW and TXINTSTAT\_MASKED registers, respectively.

The following list shows the transmit host error conditions:

- SOP error
- OWNERSHIP bit not set in SOP buffer
- next buffer descriptor pointer without EOP set to 0
- buffer pointer set to 0
- buffer length set to 0
- packet length error

The receive host error conditions are show in the following list:

- Ownership bit not set in input buffer.
- Zero buffer pointer.
- Zero buffer Length on non-SOP descriptor.
- SOP buffer length not greater than offset.

The host error interrupt is disabled by clearing to 0 the appropriate bit in the CPDMA\_INTMASK\_CLEAR register.

#### 14.3.1.3.4.4 MDIO Interrupts

MDIO\_LINKINT is set if there is a change in the link state of the PHY corresponding to the address in the PHYADDRMON field of the MDIouserPHYSEL $n$  register and the corresponding LINKINTENB bit is set. The MDIO\_LINKINT event is also captured in the MDIOLINKINTMASKED register. When the GO bit in the MDIouserACCESS $n$  registers transitions from 1 to 0, indicating the completion of a user access, and the corresponding USERINTMASKSET bit in the MDIouserINTMASKSET register is set, the MDIO\_USERINT signal is asserted 1. The MDIO\_USERINT event is also captured in the MDIouserINTMASKED register.

To enable the miscellaneous pulse interrupt:

The miscellaneous interrupt(s) is selected by setting one or more bits in the miscellaneous interrupt enable register (MISC\_EN).

- The Statistics interrupt is enabled by setting to 1 the STAT\_INT\_MASK bit in the DMA\_INTMASK\_SET register.
- The HOST\_PEND is enabled by setting to 1 the HOST\_ERR\_INT\_MASK in the DMA\_INTMASK\_SET register.

Upon receiving of an interrupt, software should perform the following:

- Read the Cn\_MISC\_STAT register to determine the source of the interrupt.
- Process the interrupt.
- Write the value 3h to the CPDMA\_EOI\_VECTOR register.

#### 14.3.1.4 Embedded Memories

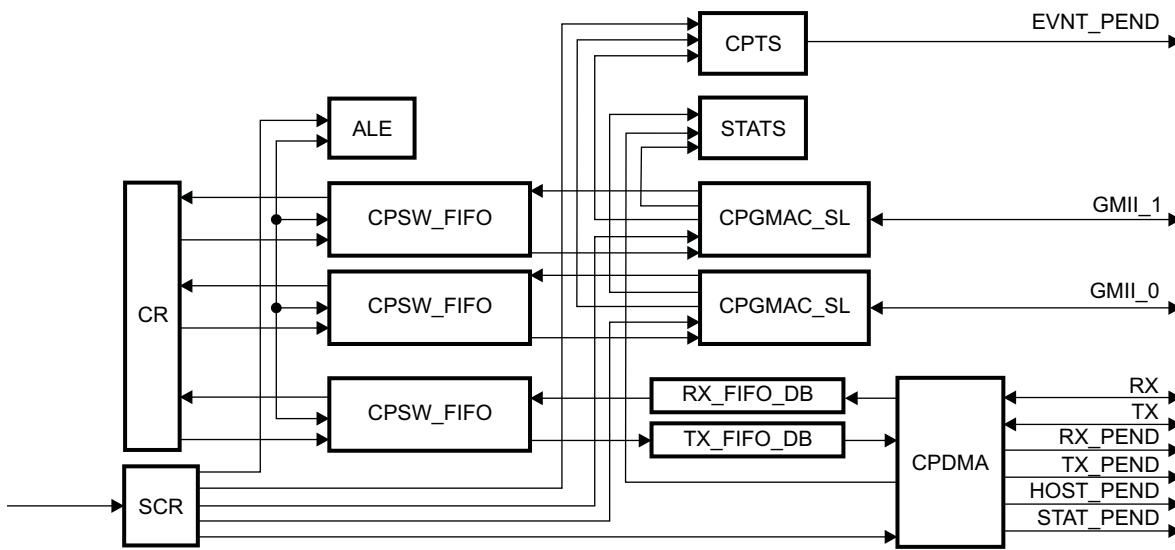
| Memory Type Description             | Number of Instances |
|-------------------------------------|---------------------|
| Single port 2560 by 64 RAM          | 3 (Packet FIFOs)    |
| Single port 64-word by 1152-bit RAM | 1 (ALE)             |
| Single port 2048-word by 32-bit RAM | 1 (CPPI)            |

#### 14.3.2 CPSW\_3G

The CPSW\_3G GMII interfaces are compliant to the IEEE Std 802.3 Specification.

The CPSW\_3G contains two CPGMAC\_SL interfaces (ports 1 and 2), one CPPI 3.0 interface Host Port (port 0), Common Platform Time Sync (CPTS), ALE Engine and CPDMA.

The top level block diagram of CPSW\_3G is shown below:

**Figure 14-6. CPSW\_3G Block Diagram**


#### 14.3.2.1 Media Independent Interface (GMII)

The CPSW\_3G contains two CPGMAC\_SL submodules. Each CPGMAC\_SL has a single GMII interface. The CPGMAC\_SL submodules are ports 1 and 2. For more details, see [Section 14.3.3, Ethernet Mac Sliver \(CPGMAC\\_SL\)](#).

#### 14.3.2.2 IEEE 1588v2 Clock Synchronization Support

The CPSW\_3G supports IEEE 1588v2 clock synchronization. Ethernet GMII Transmit (egress) and receive (ingress) time sync operation are also supported.

##### 14.3.2.2.1 IEEE 1588v2 Receive Packet Operation

There are two CPSW\_3G receive time sync interfaces for each ethernet port. The first is the TS\_RX\_MII interface and the second is the TS\_RX\_DEC interface. Both interfaces are generated in the switch and are input to the CPTS module. There are register bits in the CPSW\_3G that control time sync operations in addition to the registers in the CPTS module. The pX\_ts\_rx\_en bit in the switch Px\_Control register must be set for receive time sync operation to be enabled (TS\_RX\_MII).

The TS\_RX\_MII interface issues a record signal (pX\_ts\_rx\_mii\_rec) along with a handle (pX\_ts\_rx\_mii\_hdl[3:0]) to the CPTS controller for each packet that is received. The record signal is a single clock pulse indicating that a receive packet has been detected at the associated port MII interface.

The handle value is incremented with each packet and rolls over to zero after 15. There are 16 possible handle values so there can be a maximum of 16 packets “in flight” from the TS\_RX\_MII to the TS\_RX\_DEC block (through the CPGMAC\_SL) at any given time. A handle value is reused (not incremented) for any received packet that is shorter than about 31 octets (including preamble). Handle reuse on short packets prevents any possible overrun condition if multiple fragments are consecutively received.

The TS\_RX\_MII logic is in the receive wireside clock domain. There is no decode logic in the TS\_RX\_MII to determine if the packet is a time sync event packet or not. Each received packet generates a record signal and new handle. The handle is sent to the CPTS controller with the record pulse and the handle is also sent to the TS\_RX\_DEC block along with the packet. The packet decode is performed in the TS\_RX\_DEC block. The decode function is separated from the record function because in some systems the incoming packet can be encrypted. The decode function would be after packet decryption in those systems.

The TS\_RX\_DEC function decodes each received packet and determines if the packet meets the time sync event packet criteria. If the packet is determined to be a time sync event packet, then the time sync event packet is signaled to the CPTS controller via the TS\_RX\_DEC interface (pX\_ts\_rx\_dec\_evnt, pX\_ts\_rx\_dec\_hdl[3:0], pX\_ts\_rx\_dec\_msg\_type, pX\_ts\_rx\_dec\_seq\_id). The event signal is a single clock pulse indicating that the packet matched the time sync event packet criteria and that the associated packet handle, message type, and sequence ID are valid. No indication is given for received packets that do not meet the time sync event criteria. The 16-bit sequence ID is found in the time sync event packet at the sequence ID offset into the PTP message header (pX\_ts\_seq\_id\_offset). A packet is determined to be a receive event packet under the following conditions:

#### **14.3.2.2.1.1 Annex F**

1. Receive time sync is enabled (pX\_ts\_rx\_en is set in the switch Px\_Control register).

2. One of the following sequences is true:

Where the first packet ltype matches:

- ts\_ltype1 and pX\_ts\_ltype1\_en is set
- ts\_ltype2 and pX\_ts\_ltype2\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
- vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
- vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet ltype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set

3. The PTP message begins in the byte after the LTYPE.

4. The packet message type is enabled in the pX\_ts\_msg\_type\_en field in the Px\_TS\_SEQ\_MTYPE register.

5. The packet was received without error (not long/short/mac\_ctl/crc/code/align).

6. The ALE determined that the packet is to be sent only to the host (port 0).

#### **14.3.2.2.1.2 Annex D**

1. Receive time sync is enabled (pX\_ts\_rx\_en is set in the switch Px\_Control register).

2. One of the following sequences is true:

Where the first packet ltype matches:

- 0x0800 and pX\_ts\_annex\_d\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches 0x0800 and pX\_ts\_annex\_d\_en is set
- vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches 0x0800 and pX\_ts\_annex\_d\_en is set
- vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet ltype matches 0x0800 and pX\_ts\_annex\_d\_en is set

3. Byte 14 (the byte after the LTYPE) contains 0x45 (IP\_VERSION).

**Note:** The byte numbering assumes that there are no vlans. The byte number is intended to show the relative order of the bytes.

4. Byte 22 contains 0x00 if the pX\_ts\_ttl\_nonzero bit in the switch Px\_Control register is zero or byte 22 contains any value if pX\_ts\_ttl\_nonzero is set. Byte 22 is the time to live field.

5. Byte 23 contains 0x11 (UDP Fixed).
6. Byte 30 contains decimal 224 (0xe0).
7. Byte 31 contains 0x00.
8. Byte 32 contains 0x01.
9. Byte 33 contains one of the following:
  - Decimal 129 and the pX\_ts\_129 bit in the switch Px\_Control register is set
  - Decimal 130 and the pX\_ts\_130 bit in the switch Px\_Control register is set
  - Decimal 131 and the pX\_ts\_131 bit in the switch Px\_Control register is set
  - Decimal 129 and the pX\_ts\_132 bit in the switch Px\_Control register is set
10. Bytes 36 and 37 contain one of the following:
  - Decimal 0x01 and 0x3f respectively and the pX\_ts\_319 bit in the switch Px\_Control register is set.
  - Decimal 0x01 and 0x40 respectively and the pX\_ts\_320 bit in the switch Px\_Control register is set.
11. The PTP message begins in byte 42.
12. The packet message type is enabled in the pX\_ts\_msg\_type\_en field in Px\_Control.
13. The packet was received without error (not long/short/mac\_ctl/crc/code/align).
14. The ALE determined that the packet is to be sent only to the host (port 0).

#### **14.3.2.2.2 IEEE 1588v2 Transmit Packet Operation**

There are two CPSW\_3G transmit time sync interfaces for each ethernet port. The first is the TS\_TX\_DEC interface and the second is the TS\_TX\_MII interface. Both interfaces are generated in the switch and are input to the CPTS module. The pX\_ts\_tx\_en bit in the Px\_Control register must be set for transmit time sync operation to be enabled.

The TS\_TX\_DEC function decodes each packet to be transmitted and determines if the packet meets the time sync event packet criteria. If the packet is determined to be a time sync event packet, then the time sync event is signaled to the CPTS controller via the TS\_TX\_DEC interface (pX\_ts\_tx\_dec\_evnt, pX\_ts\_tx\_dec\_hdl[3:0], pX\_ts\_tx\_dec\_msg\_type, pX\_ts\_tx\_dec\_seq\_id). The event signal is a single clock pulse indicating that the packet matched the time sync event packet criteria and that the associated packet handle, message type, and sequence ID are valid.

The 16-bit sequence ID is found in the time sync event packet at the sequence ID offset into the message header (pX\_ts\_seq\_id\_offset). No indication is given for transmit packets that do not meet the time sync event criteria. The time sync event packet handle is also passed along with the packet to the TS\_TX\_MII with an indication that the packet is a time sync event packet. Unlike receive, only transmit event packets increment the handle value. The decode function is separated from the record function because some systems may encrypt the packet. The encryption is after the decode function on transmit (egress). A packet is determined to be a transmit event packet if the following is met:

##### **14.3.2.2.2.1 Annex F**

1. Transmit time sync is enabled (pX\_ts\_tx\_en is set in the switch Px\_Control register).
2. One of the following sequences is true:
  - The first packet ltype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
  - The first packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
  - The first packet ltype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
  - The first packet ltype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
  - The first packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
  - The first packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set

- Itype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
- The first packet Itype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet Itype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet Itype matches ts\_ltype1 and pX\_ts\_ltype1\_en is set
  - The first packet Itype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet Itype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet Itype matches ts\_ltype2 and pX\_ts\_ltype2\_en is set
3. The packet message type is enabled in pX\_ts\_msg\_type\_en.
  4. The packet was received by the host (port 0).

#### 14.3.2.2.2 Annex D

1. Transmit time sync is enabled (pX\_ts\_tx\_en is set in the switch Px\_Control register).
2. One of the following sequences is true:
  - The first packet Itype matches 0x0800 and pX\_ts\_annex\_d\_en is set
  - The first packet Itype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet Itype matches 0x0800 and pX\_ts\_annex\_d\_en is set
  - The first packet Itype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet Itype matches 0x0800 and pX\_ts\_annex\_d\_en is set
  - The first packet Itype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet Itype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet Itype matches 0x0800 and pX\_ts\_annex\_d\_en is set
3. Byte 14 (the byte after the LTYPE) contains 0x45 (IP\_VERSION).
 

**Note:** The byte numbering assumes that there are no vlans. The byte number is intended to show the relative order of the bytes. If VLAN(s) are present, then the byte numbers push down.
4. Byte 22 contains 0x00 if the pX\_ts\_ttl\_nonzero bit in the switch Px\_Control register is zero or byte 22 contains any value if pX\_ts\_ttl\_nonzero is set. Byte 22 is the time to live field.
5. Byte 23 contains 0x11 (UDP Fixed).
6. Byte 30 contains decimal 224 (0xe0)
7. Byte 31 contains 0x00
8. Byte 32 contains 0x01
9. Byte 33 contains one of the following:
  - Decimal 129 and the pX\_ts\_129 bit in the switch Px\_Control register is set
  - Decimal 130 and the pX\_ts\_130 bit in the switch Px\_Control register is set
  - Decimal 131 and the pX\_ts\_131 bit in the switch Px\_Control register is set
  - Decimal 132 and the pX\_ts\_132 bit in the switch Px\_Control register is set
10. Bytes 36 and 37 contain either of the following:
  - Decimal 1 (hex 0x01) and decimal 63 (hex 0x3f) respectively and and the pX\_ts\_319 bit in the switch Px\_Control register is set
  - Decimal 1 (hex 0x01) and decimal 64 (hex 0x40) respectively and and the pX\_ts\_320 bit in the switch Px\_Control register is set
11. The PTP message begins in byte 42 (this is offset 0).
12. The packet message type is enabled in pX\_ts\_msg\_type\_en.
13. The packet was received by the host (port 0).

The TS\_TX\_MII interface issues a single clock record signal (pX\_ts\_tx\_mii\_rec) at the beginning of each transmitted packet. If the packet is a time sync event packet then a single clock event signal (pX\_ts\_tx\_mii\_evnt) along with a handle (pX\_ts\_rx\_mii\_hdl[2:0]) will be issued before the next record signal for the next packet. The event signal will not be issued for packets that did not meet the time sync event criteria in the TS\_TX\_DEC function. If consecutive record indications occur without an interleaving event indication, then the packet associated with the first record was not a time sync event packet.

The record signal is a single clock pulse indicating that a transmit packet egress has been detected at the associated port MII interface. The handle value is incremented with each time sync event packet and rolls over to zero after 7. There are 8 possible handle values so there can be a maximum of eight time sync event packets “in flight” from the TS\_TX\_DEC to the TS\_TX\_MII block at any given time. The handle value increments only on time sync event packets. The TS\_TX\_MII logic is in the transmit wireside clock domain.

#### 14.3.2.3 Device Level Ring (DLR) Support

Device Level Ring (DLR) support is enabled by setting the dlr\_en bit in the CPSW\_Control register. When enabled, incoming received DLR packets are detected and sent to queue 3 (highest priority) of the egress port(s). If the host port is the egress port for a DLR packet then the packet is sent on the CPDMA Rx channel selected by the p0\_dlr\_cpdma\_ch field in the P0\_Control register. The supervisor node MAC address feature is supported with the dlr\_unicast bit in the unicast address table entry.

When set, the dlr\_unicast bit causes a packet with the matching destination address to be flooded to the vlan\_member\_list minus the receive port and minus the host port (the port\_number field in the unicast address table entry is a don't care). Matching dlr\_unicast packets are flooded regardless of whether the packet is a DLR packet or not. The en\_p0\_uni\_flood bit in the ALE\_Control register has no effect on DLR unicast packets. Packets are determined to be DLR packets, as shown:

1. DLR is enabled (dlr\_en is set in the switch CPSW\_Control register).
2. One of the following sequences is true:
  - The first packet ltype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches dlr\_ltype.
  - The first packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the second packet ltype matches dlr\_ltype.
  - The first packet ltype matches vlan\_ltype1 and pX\_vlan\_ltype1\_en is set and the second packet ltype matches vlan\_ltype2 and pX\_vlan\_ltype2\_en is set and the third packet ltype matches dlr\_ltype.

#### 14.3.2.4 CPDMA RX and TX Interfaces

The CPDMA submodule is a CPPI 3.0 compliant packet DMA transfer controller. The CPPI 3.0 interface is port 0.

After reset, initialization, and configuration the host may initiate transmit operations. Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the STATERAM block. The transmit DMA controller then fetches the first packet in the packet chain from memory in accordance with CPPI 3.0 protocol. The DMA controller writes the packet into the external transmit FIFO in 64-byte bursts (maximum).

Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration. The receive DMA controller writes the receive packet data to external memory in accordance with CPPI 3.0 protocol. See the CPPI Buffer Descriptors section for detailed description of Buffer Descriptors

#### 14.3.2.4.1 CPPI Buffer Descriptors

The buffer descriptor is a central part of the 3PSW Ethernet Subsystem and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data.

Host Software sends and receives network frames via the CPPI 3.0 compliant host interface. The host interface includes module registers and host memory data structures. The host memory data structures are buffer descriptors and data buffers. Buffer descriptors are data structures that contain information about a single data buffer. Buffer descriptors may be linked together to describe frames or queues of frames for transmission of data and free buffer queues available for received data.

Note: The 8k bytes of Ethernet Subsystem CPPI RAM begin at address 0x4a102000 and end at 0x4a103FFF from the 3PSW perspective. The buffer descriptors programmed to access the CPPI RAM memory should use the address range from 0x4a102000.

##### 14.3.2.4.1.1 TX Buffer Descriptors

A TX buffer descriptor is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

**Figure 14-7. Tx Buffer Descriptor Format**

|                         |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-------------------------|-------|---------------|-----|-----------------------|-------------|----------|----------------|---------------|----------|------------|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Word 0                  |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31                      | 0     |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Next Descriptor Pointer |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Word 1                  |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31                      | 0     |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Buffer Pointer          |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Word 2                  |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31                      | 0     |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 16 15                   |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Buffer Offset           |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Word 3                  |       |               |     |                       |             |          |                |               |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 31                      | 30    | 29            | 28  | 27                    | 26          | 25       |                |               | 21       | 20         | 19 | 18 | 17 | 16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SOP                     | EOP   | Owner<br>ship | EOQ | Teardown_Co<br>mplete | Pass<br>CRC | Reserved | To_Po<br>rt_En |               | Reserved | To_Port_En |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15                      | 11 10 |               |     |                       |             |          |                |               | 0        |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Reserved                |       |               |     |                       |             |          |                | packet_length |          |            |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

##### 14.3.2.4.1.1.1 CPPI Tx Data Word – 0

###### Next Descriptor Pointer

The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue. This pointer is used to create a linked list of buffer descriptors. If the value of this pointer is zero, then the current buffer is the last buffer in the queue. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC. The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL.

If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly appended descriptor. The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read. In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time. The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC.

##### 14.3.2.4.1.1.2 CPPI Tx Data Word – 1

###### Buffer Pointer

The byte aligned memory address of the buffer associated with the buffer descriptor. The host sets the **buffer\_pointer**. The software application must set this value prior to adding the descriptor to the active transmit list. This pointer is not altered by the EMAC.

#### 14.3.2.4.1.1.3 CPPI Tx Data Word – 2

##### **Buffer\_Offset**

Buffer Offset – Indicates how many unused bytes are at the start of the buffer. A value of 0x0000 indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer. A value of 0x000F (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer. The host sets the buffer\_offset value (which may be zero to the buffer length minus 1). Valid only on sop.

##### **Buffer\_Length**

Buffer Length – Indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. The host sets the buffer\_length. The buffer\_length must be greater than zero.

#### 14.3.2.4.1.1.4 CPPI Tx Data Word – 3

##### **Start of Packet (SOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

0 - Not start of packet buffer

1 - Start of packet buffer

##### **End of Packet (EOP) Flag**

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag. This bit is set by the software application and is not altered by the EMAC.

0 - Not end of packet buffer.

1 - End of packet buffer.

##### **Ownership**

When set this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC. This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue. For a single fragment packet, the SOP, EOP, and OWNER flags are all set. The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet. Note that this flag is valid on SOP descriptors only.

0 - The packet is owned by the host

1 - The packet is owned by the port

##### **EOQ**

When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted. This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set), and there are no more descriptors in the transmit list (next descriptor pointer is NULL).

The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted. This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC. Note that this flag is valid on EOP descriptors only.

- 0 - The Tx queue has more packets to transfer.
- 1 - The Descriptor buffer is the last buffer in the last packet in the queue.

#### **teardown\_Complete**

This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission. Note that this flag is valid on SOP descriptors only. Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not processed by the EMAC.

- 0 - The port has not completed the teardown process.
- 1 - The port has completed the commanded teardown process.

#### **pass\_crc**

This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue. Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC. When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC. The buffer length and packet length fields do not include the CRC bytes. When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data. The buffer length and packet length fields include the CRC bytes, as they are part of the valid packet data. Note that this flag is valid on SOP descriptors only.

- 0 – The CRC is not included with the packet data and packet length.
- 1 – The CRC is included with the packet data and packet length.

#### **to\_port**

To Port – Port number to send the directed packet to. This field is set by the host. This field is valid on SOP. Directed packets go to the directed port, but an ALE lookup is performed to determine untagged egress in VLAN\_AWARE mode.

- 1 – Send the packet to port 1 if to\_port\_en is asserted.
- 2 – Send the packet to port 2 if to\_port\_en is asserted.

#### **To\_port\_enable**

To Port Enable – Indicates when set that the packet is a directed packet to be sent to the to\_port field port number. This field is set by the host. The packet is sent to one port only (index not mask). This bit is valid on SOP.

- 0 – not a directed packet
- 1 – directed packet

#### **Packet Length**

Specifies the number of bytes in the entire packet. Offset bytes are not included. The sum of the buffer\_length fields should equal the packet\_length. Valid only on SOP. The packet length must be greater than zero. The packet data will be truncated to the packet length if the packet length is shorter than the sum of the packet buffer descriptor buffer lengths. A host error occurs if the packet length is greater than the sum of the packet buffer descriptor buffer lengths

#### **14.3.2.4.1.2 RX Buffer Descriptors**

An RX buffer descriptor is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

**Figure 14-8. Rx Buffer Descriptor Format**

|                         |          |               |               |                           |                    |               |       |             |             |         |               |           |    |    |    |
|-------------------------|----------|---------------|---------------|---------------------------|--------------------|---------------|-------|-------------|-------------|---------|---------------|-----------|----|----|----|
| Word 0                  | 31       | 0             |               |                           |                    |               |       |             |             |         |               |           |    |    |    |
| Next Descriptor Pointer |          |               |               |                           |                    |               |       |             |             |         |               |           |    |    |    |
| Word 1                  | 31       | 0             |               |                           |                    |               |       |             |             |         |               |           |    |    |    |
| Buffer Pointer          |          |               |               |                           |                    |               |       |             |             |         |               |           |    |    |    |
| Word 2                  | 31       | 27 26         | 16 15         | 11 10                     | 0                  |               |       |             |             |         |               |           |    |    |    |
|                         | Reserved | Buffer Offset |               |                           | Reserved           | Buffer Length |       |             |             |         |               |           |    |    |    |
| Word 3                  | 31       | 30            | 29            | 28                        | 27                 | 26            | 25    | 24          | 23          | 22      | 21            | 20        | 19 | 18 | 16 |
| SOP                     | EOP      | Owner<br>ship | EOQ           | Teard<br>own_Co<br>mplete | Passe<br>d_CR<br>C | Long          | Short | MAC_<br>Ctl | Overru<br>n | PKT_Err | Rx_Vlan_Encap | From_Port |    |    |    |
| 15                      |          |               | 11            |                           | 10                 |               |       |             |             |         |               |           |    |    | 0  |
| Reserved                |          |               | packet_length |                           |                    |               |       |             |             |         |               |           |    |    |    |

**14.3.2.4.1.2.1 CPPI Rx Data Word – 0****next\_descriptor\_pointer**

The 32-bit word aligned memory address of the next buffer descriptor in the RX queue. This is the mechanism used to reference the next buffer descriptor from the current buffer descriptor. If the value of this pointer is zero then the current buffer is the last buffer in the queue. The host sets the **next\_descriptor\_pointer**.

**14.3.2.4.1.2.2 CPPI Rx Data Word – 1****buffer\_pointer**

The byte aligned memory address of the buffer associated with the buffer descriptor. The host sets the **buffer\_pointer**.

**14.3.2.4.1.2.3 CPPI Rx Data Word – 2****Buffer\_Offset**

Buffer Offset – Indicates how many unused bytes are at the start of the buffer. A value of 0x0000 indicates that there are no unused bytes at the start of the buffer and that valid data begins on the first byte of the buffer. A value of 0x000F (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer. The port writes the **buffer\_offset** with the value from the **rx\_buffer\_offset** register value. The host initializes the **buffer\_offset** to zero for free buffers. The **buffer\_length** must be greater than the **rx\_buffer\_offset** value. The buffer offset is valid only on **sop**.

**Buffer\_Length**

Buffer Length – Indicates how many valid data bytes are in the buffer. Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field. The host initializes the **buffer\_length**, but the port may overwrite the host initiated value with the actual buffer length value on SOP and/or EOP buffer descriptors. SOP buffer length values will be overwritten if the packet size is less than the size of the buffer or if the offset is nonzero. EOP buffer length values will be overwritten if the entire buffer is not filled up with data. The **buffer\_length** must be greater than zero.

#### 14.3.2.4.1.2.4 CPPI Rx Data Word – 3

##### Start of Packet (SOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet. In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on SOP descriptors.

##### End of Packet (EOP) Flag

When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet. In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set. Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set. This flag is initially cleared by the software application before adding the descriptor to the receive queue. This bit is set by the EMAC on EOP descriptors.

##### Ownership (OWNER) Flag

When set, this flag indicates that the descriptor is currently owned by the EMAC. This flag is set by the software application before adding the descriptor to the receive descriptor queue. This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet. The flag is updated by the EMAC on SOP descriptor only. So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC. (Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set.)

##### End of Queue (EOQ) Flag

When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted. This flag is initially cleared by the software application prior to adding the descriptor to the receive queue. This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received (also sets the EOP flag), and there are no more descriptors in the receive list (next descriptor pointer is NULL). The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted. This is useful when the application appends additional free buffer descriptors to an active receive queue. Note that this flag is valid on EOP descriptors only.

##### Teardown Complete (TDOWNCMPLT) Flag

This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data. This would happen under device driver reset or shutdown conditions. The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs. No additional queue processing is performed.

##### Pass CRC (PASSCRC) Flag

This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC. This flag should be cleared by the software application before submitting the descriptor to the receive queue.

##### Long (Jabber) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RX\_CEF\_EN bit was set in the MacControl. Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.

##### Short (Fragment) Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RX\_CSF\_EN bit was set in the MacControl.

##### Control Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RX\_CMF\_EN bit was set in the MacControl.

##### Overrun Flag

This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.

#### **Pkt\_error Flag**

Packet Contained Error on Ingress –

00 – no error

01 – CRC error on ingress

10 – Code error on ingress

11 – Align error on ingress

#### **rx\_vlan\_encap**

VLAN Encapsulated Packet – Indicates when set that the packet data contains a 32-bit VLAN header word that is included in the packet byte count. This field is set by the port to be the value of the CPSW control register rx\_vlan\_encap bit

#### **from\_port**

From Port – Indicates the port number that the packet was received on (ingress to the switch).

#### **Packet Length**

Specifies the number of bytes in the entire packet. The packet length is reduced to 12-bits. Offset bytes are not included. The sum of the buffer\_length fields should equal the packet\_length. Valid only on SOP.

### **14.3.2.4.2 Receive DMA Interface**

The Receive DMA is an eight channel CPPI 3.0 compliant interface. Each channel has a single queue for frame reception.

#### **14.3.2.4.2.1 Receive DMA Host Configuration**

To configure the Rx DMA for operation the host must perform the following:

- Initialize the receive addresses.
- Initialize the Rx\_HDP Registers to zero.
- Enable the desired receive interrupts in the IntMask register.
- Write the rx\_buffer\_offset register value.
- Setup the receive channel(s) buffer descriptors in host memory as required by CPPI 3.0.
- Enable the RX DMA controller by setting the rx\_en bit in the Rx\_Control register.

#### **14.3.2.4.2.2 Receive Channel Teardown**

The host commands a receive channel teardown by writing the channel number to the Rx\_Teardown register. When a teardown command is issued to an enabled receive channel the following will occur:

- Any current frame in reception will complete normally.
- The teardown\_complete bit will be set in the next buffer descriptor in the chain if there is one.
- The channel head descriptor pointer will be cleared to zero
- A receive interrupt for the channel will be issued to the host.
- The host should acknowledge a teardown interrupt with a 0xffffffffc acknowledge value.

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete buffer descriptor bit. The port does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with a 0xffffffffc acknowledge value (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location to determine if the interrupt was due to a commanded teardown. The read value will be 0xffffffffc if the interrupt was due to a teardown command.

#### **14.3.2.4.3 Transmit DMA Interface**

The Transmit DMA is an eight channel CPPI 3.0 compliant interface. Priority between the eight queues may be either fixed or round robin as selected by tx\_ptype in the DMAControl register. If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority. Round robin priority proceeds from channel 0 to channel 7. Packet Data transfers occur on the TX\_VBUSP interface in 64-byte maximum burst transfers

##### **14.3.2.4.3.1 Transmit DMA Host Configuration**

To configure the TX DMA for operation the host must do the following:

- Initialize the Tx\_HDP registers to a zero value.
- Enable the desired transmit interrupts in the IntMask register.
- Setup the transmit channel(s) buffer descriptors in host memory as defined in CPPI 3.0.
- Configure and enable the transmit operation as desired in the TxControl register.
- Write the appropriate Tx\_HDP registers with the appropriate values to start transmit operations.

##### **14.3.2.4.3.2 Transmit Channel Teardown**

The host commands a transmit channel teardown by writing the channel number to the Tx\_Teardown register. When a teardown command is issued to an enabled transmit channel the following will occur:

- Any frame currently in transmission will complete normally
- The teardown complete bit will be set in the next sop buffer descriptor (if there is one).
- The channel head descriptor pointer will be set to zero.
- An interrupt will be issued to inform the host of the channel teardown.
- The host should acknowledge a teardown interrupt with a 0xffffffffc acknowledge value

Channel teardown may be commanded on any channel at any time. The host is informed of the teardown completion by the set teardown complete buffer descriptor bit. The port does not clear any channel enables due to a teardown command. A teardown command to an inactive channel issues an interrupt that software should acknowledge with a 0xffffffffc acknowledge value (note that there is no buffer descriptor in this case). Software may read the interrupt acknowledge location to determine if the interrupt was due to a commanded teardown. The read value will be 0xffffffffc if the interrupt was due to a teardown command.

#### **14.3.2.4.4 Transmit Rate Limiting**

Transmit operations can be configured to rate limit the transmit data for each transmit priority. Rate limiting is enabled for a channel when the tx\_rlim[7:0] bit associated with that channel is set in the DMAControl register. Rate limited channels must be the highest priority channels. For example, if two rate limited channels are required then tx\_rlim[7:0] should be set to 11000000 with the msb corresponding to channel 7.

When any channels are configured to be rate-limiting, the priority type must be fixed for transmit. Round-robin priority type is not allowed when rate-limiting. Each of the eight transmit priorities has an associated register to control the rate at which the priority is allowed to send data (Tx\_Pri(7..0)\_Rate) when the channel is rate-limiting. Each priority has a send count (pri(7..0)\_send\_cnt[13:0]) and an idle count (pri(7..0)\_idle\_cnt[13:0]). The transfer rate includes the inter-packet gap (12 bytes) and the preamble (8 bytes). The rate in Mbits/second that each priority is allowed to send is controlled by the below equation.

$$\text{Priority Transfer rate in Mbit/s} = ((\text{priN}_\text{idle\_cnt}/(\text{priN}_\text{idle\_cnt} + \text{priN}_\text{send\_cnt})) * \text{frequency}) * 32$$

Where the frequency is the CLK frequency.

#### 14.3.2.4.5 Command IDLE

The cmd\_idle bit in the DMA\_Control register allows CPDMA operation to be suspended. When the idle state is commanded, the CPDMA will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the tx cell fifo will be transmitted. For receive, frames that are detected by the CPDMA after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Commanded idle is similar in operation to emulation control and clock stop.

#### 14.3.2.5 VLAN Aware Mode

The CPSW\_3G is in VLAN aware mode when the CPSW Control register vlan\_aware bit is set. In VLAN aware mode ports 0 receive packets (out of the CPSW\_3G) may or may not be VLAN encapsulated depending on the CPSW Control register rx\_vlan\_encap bit. The header packet VLAN is generated as described in [Section 14.3.3, Ethernet Mac Sliver \(CPGMAC\\_SL\)](#). Port 0 receive packet data is never modified. VLAN is not removed regardless of the force untagged egress bit for Port 0. VLAN encapsulated receive packets have a 32-bit VLAN header encapsulation word added to the packet data. VLAN encapsulated packets are specified by a set rx\_vlan\_encap bit in the packet buffer descriptor.

Port 0 transmit packets are never VLAN encapsulated (encapsulation is not allowed).

In VLAN aware mode, transmitted packet data is changed depending on the packet type (pkt\_type), packet priority (pkt\_pri), and VLAN information as shown in the below tables:

**Figure 14-9. VLAN Header Encapsulation Word**

| 31               | 29 | 28          | 27 |             |   |   |   |          |   |   |   |   |   | 16 |  |
|------------------|----|-------------|----|-------------|---|---|---|----------|---|---|---|---|---|----|--|
| HDR_PKT_Priority |    | HDR_PKT_CFI |    | HDR_PKT_Vid |   |   |   |          |   |   |   |   |   |    |  |
| 15               |    |             | 10 | 9           | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |    |  |
| Reserved         |    |             |    | PKT_Type    |   |   |   | Reserved |   |   |   |   |   |    |  |

**Table 14-9. VLAN Header Encapsulation Word Field Descriptions**

| Field            | Description  |
|------------------|--|
| HDR_PKT_Priority | Header Packet VLAN priority (Highest priority: 7)  |
| HDR_PKT_CFI      | Header Packet VLAN CFI bit.  |
| HDR_PKT_Vid      | Header Packet VLAN ID  |
| PKT_Type         | Packet Type. Indicates whether the packet is VLAN-tagged, priority-tagged, or non-tagged.<br>00: VLAN-tagged packet<br>01: Reserved<br>10: Priority-tagged packet<br>11: Non-tagged packet |

#### 14.3.2.6 VLAN Unaware Mode

The CPSW\_3G is in VLAN unaware mode when the CPSW Control register vlan\_aware bit is cleared. Port 0 receive packets (out of the CPSW\_3G) may or may not be VLAN encapsulated depending on the CPSW Control register rx\_vlan\_encap bit. Port 0 transmit packets are never VLAN encapsulated.

#### 14.3.2.7 Address Lookup Engine (ALE)

The address lookup engine (ALE) processes all received packets to determine which port(s) if any that the packet should be forwarded to. The ALE uses the incoming packet received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded. The ALE outputs the port mask to the switch fabric that indicates the port(s) the packet should be forwarded to. The ALE is enabled when the ale\_enable bit in the ALE\_Control register is set. All packets are dropped when the ale\_enable bit is cleared to zero.

In normal operation, the CPGMAC\_SL modules are configured to issue an abort, instead of an end of packet, at the end of a packet that contains an error (runt, frag, oversize, jabber, crc, alignment, code etc.) or at the end of a mac control packet. However, when the CPGMAC\_SL configuration bit(s) cef, csf, or cmf are set, error frames, short frames or mac control frames have a normal end of packet instead of an abort at the end of the packet. When the ALE receives a packet that contains errors (due to a set header error bit), or a mac control frame and does not receive an abort, the packet will be forwarded only to the host port (port 0). No ALE learning occurs on packets with errors or mac control frames. Learning is based on source address and lookup is based on destination address.

The ALE may be configured to operate in bypass mode by setting the ale\_bypass bit in the ALE\_Control register. When in bypass mode, all CPGMAC\_SL received packets are forwarded only to the host port (port 0). Packets from the two ports can be on separate Rx DMA channels by configuring the CPDMA\_Rx\_Ch\_Map register. In bypass mode, the ALE processes host port transmit packets the same as in normal mode. In general, packets would be directed by the host in bypass mode.

The ALE may be configured to operate in OUI deny mode by setting the enable\_oui\_deny bit in the ALE\_Control register. When in OUI deny mode, a packet with a non-matching OUI source address will be dropped unless the destination address matches a multicast table entry with the super bit set. Broadcast packets will be dropped unless the broadcast address is entered into the table with the super bit set. Unicast packets will be dropped unless the unicast address is in the table with block and secure both set (supervisory unicast packet).

Multicast supervisory packets are designated by the super bit in the table entry. Unicast supervisory packets are indicated when block and secure are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing.

#### **14.3.2.7.1 Address Table Entry**

The ALE table contains 1024 entries. Each table entry represents a free entry, an address, a VLAN, an address/VLAN pair, or an OUI address. Software should ensure that there are not double address entries in the table. The double entry used would be indeterminate. Reserved table bits must be written with zeroes.

Source Address learning occurs for packets with a unicast, multicast or broadcast destination address and a unicast or multicast (including broadcast) source address. Multicast source addresses have the group bit (bit 40) cleared before ALE processing begins, changing the multicast source address to a unicast source address. A multicast address of all ones is the broadcast address which may be added to the table. A learned unicast source address is added to the table with the following control bits:

**Table 14-10. Learned Address Control Bits**

|              |    |
|--------------|----|
| unicast_type | 11 |
| Block        | 0  |
| Secure       | 0  |

If a received packet has a source address that is equal to the destination address then the following occurs:

- The address is learned if the address is not found in the table.
- The address is updated if the address is found.
- The packet is dropped.

#### **14.3.2.7.1.1 Free Table Entry**

**Table 14-11. Free (Unused) Address Table Entry Bit Values**

| 71:62    | 61:60           | 59:0     |
|----------|-----------------|----------|
| Reserved | Entry Type (00) | Reserved |

#### 14.3.2.7.1.2 Multicast Address Table Entry

**Table 14-12. Multicast Address Table Entry Bit Values**

| 71:70    | 68:66     | 65    | 64       | 63:62           | 61:60           | 59:48    | 47:0              |
|----------|-----------|-------|----------|-----------------|-----------------|----------|-------------------|
| Reserved | Port Mask | Super | Reserved | Mcast Fwd State | Entry Type (01) | Reserved | Multicast Address |

##### Table Entry Type

00: Free Entry

01: Address Entry : unicast or multicast determined by dest **address bit 40** .

10: VLAN entry

11: VLAN Address Entry : unicast or multicast determined by **address bit 40**.

##### Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

##### Port Mask(2:0) (PORT\_MASK)

This three bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

##### Multicast Forward State (MCAST\_FWD\_STATE)

Multicast Forward State – Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port\_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00: Forwarding

01: Blocking/Forwarding/Learning

10: Forwarding/Learning

11: Forwarding

The forward state test returns a true value if both the Rx and Tx ports are in the required state.

##### Table Entry Type (ENTRY\_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

##### Packet Address (MULTICAST\_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### 14.3.2.7.1.3 VLAN/Multicast Address Table Entry

**Table 14-13. VLAN/Multicast Address Table Entry Bit Values**

| 71:69    | 68:66     | 65    | 64       | 63:62           | 61:60           | 59:48   | 47:0              |
|----------|-----------|-------|----------|-----------------|-----------------|---------|-------------------|
| Reserved | Port Mask | Super | Reserved | Mcast Fwd State | Entry Type (11) | vlan_id | Multicast Address |

### **Supervisory Packet (SUPER)**

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

### **Port Mask(2:0) (PORT\_MASK)**

This three bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

### **Multicast Forward State (MCAST\_FWD\_STATE)**

Multicast Forward State – Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port\_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00 – Forwarding

01 – Blocking/Forwarding/Learning

10 – Forwarding/Learning

11 – Forwarding

The forward state test returns a true value if both the Rx and Tx ports are in the required state.

### **Table Entry Type (ENTRY\_TYPE)**

Address entry type. Unicast or multicast determined by address bit 40.

11: VLAN address entry. Unicast or multicast determined by address bit 40.

### **VLAN ID (VLAN\_ID)**

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

### **Packet Address (MULTICAST\_ADDRESS)**

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

#### **14.3.2.7.1.4 Unicast Address Table Entry**

**Table 14-14. Unicast Address Table Entry Bit Values**

| <b>71:70</b> | <b>69</b>   | <b>68</b> | <b>67:66</b> | <b>65</b> | <b>64</b> | <b>63:62</b>              | <b>61:60</b>    | <b>59:48</b> | <b>47:0</b>     |
|--------------|-------------|-----------|--------------|-----------|-----------|---------------------------|-----------------|--------------|-----------------|
| Reserved     | DLR Unicast | Reserved  | Port Number  | Block     | Secure    | Unicast Type (00) or (X1) | Entry Type (01) | Reserved     | Unicast Address |

### **DLR Unicast**

DLR Unicast – When set, this bit indicates that the address is a Device Level Ring (DLR) unicast address. Received packets with a matching destination address will be flooded to the vlan\_member\_list (minus the receive port and the host port). The port\_number field is a don't care when this bit is set. Matching packets received on port 1 egress on port 2. Matching packets received on port 2 egress on port 1. Matching packets received on port 0 egress on ports 1 and 2.

### **Port Number (PORT\_NUMBER)**

Port Number – This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).

### **Block (BLOCK)**

Block – The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 – Address is not blocked.

1 – Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

### **Secure (SECURE)**

Secure – This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port\_number.

0 – Received port number is a don't care.

1 – Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

### **Unicast Type (UNICAST\_TYPE)**

Unicast Type – This field indicates the type of unicast address the table entry contains.

00 – Unicast address that is not ageable.

01 – Ageable unicast address that has not been touched.

10 – OUI address - lower 24-bits are don't cares (not ageable).

11 – Ageable unicast address that has been touched.

### **Table Entry Type (ENTRY\_TYPE)**

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

### **Packet Address (UNICAST\_ADDRESS)**

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

#### **14.3.2.7.1.5 OUI Unicast Address Table Entry**

**Table 14-15. OUI Unicast Address Table Entry Bit Values**

| 71:64    | 63:62             | 61:60           | 59:48    | 47:24       | 23:0     |
|----------|-------------------|-----------------|----------|-------------|----------|
| Reserved | Unicast Type (10) | Entry Type (01) | Reserved | Unicast OUI | Reserved |

### **Unicast Type (UNICAST\_TYPE)**

Unicast Type – This field indicates the type of unicast address the table entry contains.

00 – Unicast address that is not ageable.

01 – Ageable unicast address that has not been touched.

10 – OUI address - lower 24-bits are don't cares (not ageable).

11 – Ageable unicast address that has been touched.

### **Table Entry Type (ENTRY\_TYPE)**

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

#### **Packet Address (UNICAST\_OUI)**

For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup.

#### **14.3.2.7.1.6 VLAN/Unicast Address Table Entry**

**Table 14-16. Unicast Address Table Entry Bit Values**

| 71:68    | 67:66       | 65    | 64     | 63:62                     | 61:60           | 59:48   | 47:0            |
|----------|-------------|-------|--------|---------------------------|-----------------|---------|-----------------|
| Reserved | Port Number | Block | Secure | Unicast Type (00) or (X1) | Entry Type (11) | vlan_id | Unicast Address |

#### **Port Number (PORT\_NUMBER)**

Port Number – This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

#### **Block (BLOCK)**

Block – The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 – Address is not blocked.

1 – Drop a packet with a matching source or destination address (secure must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

#### **Secure (SECURE)**

Secure – This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port\_number.

0 – Received port number is a don't care.

1 – Drop the packet if the received port is not the secure port for the source address and do not update the address (block must be zero)

#### **Unicast Type (UNICAST\_TYPE)**

Unicast Type – This field indicates the type of unicast address the table entry contains.

00 – Unicast address that is not ageable.

01 – Ageable unicast address that has not been touched.

10 – OUI address - lower 24-bits are don't cares (not ageable).

11 – Ageable unicast address that has been touched.

#### **Table Entry Type (ENTRY\_TYPE)**

Address entry. Unicast or multicast determined by address bit 40.

11 – VLAN address entry. Unicast or multicast determined by address bit 40.

#### **VLAN ID (VLAN\_ID)**

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

### Packet Address (UNICAST\_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

#### 14.3.2.7.1.7 VLAN Table Entry

**Table 14-17. VLAN Table Entry**

| 71:62    | 61:60           | 59:48   | 47:27    | 26:24                 | 23:19    | 18:16                | 15:11    | 10:8                   | 7:3      | 2:0              |
|----------|-----------------|---------|----------|-----------------------|----------|----------------------|----------|------------------------|----------|------------------|
| Reserved | Entry Type (10) | vlan_id | Reserved | Force Untagged Egress | Reserved | Reg Mcast Flood Mask | Reserved | Unreg Mcast Flood Mask | Reserved | Vlan Member List |

### Table Entry Type (ENTRY\_TYPE)

10: VLAN entry

### VLAN ID (VLAN\_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

### Force Untagged Packet Egress (FORCE\_UNTAGGED\_EGRESS)

This field causes the packet VLAN tag to be removed on egress (except on port 0).

### Registered Multicast Flood Mask (REG\_MCAST\_FLOOD\_MASK)

Mask used for multicast when the multicast address is found

### Unregistered Multicast Flood Mask (UNREG\_MCAST\_FLOOD\_MASK)

Mask used for multicast when the multicast address is not found

### VLAN Member List (VLAN\_MEMBER\_LIST)

This three bit field indicates which port(s) are members of the associated VLAN.

#### 14.3.2.7.2 Packet Forwarding Processes

There are four processes that an incoming received packet may go through to determine packet forwarding. The processes are Ingress Filtering, VLAN\_Aware Lookup, VLAN\_Unaware Lookup, and Egress.

Packet processing begins in the Ingress Filtering process. Each port has an associated packet forwarding state that can be one of four values (Disabled, Blocked, Learning, or Forwarding). The default state for all ports is disabled. The host sets the packet forwarding state for each port. The receive packet processes are described in the following sections.

In the packet ingress process (receive packet process), there is a forward state test for unicast destination addresses and a forward state test for multicast addresses. The multicast forward state test indicates the port states required for the receiving port in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state for the packet to be forwarded for transmission. The mcast\_fwd\_state indicates the required port state for the receiving port as indicated in [Table 14-13](#).

The unicast forward state test indicates the port state required for the receiving port in order to forward the unicast packet. The transmit port must be in the Forwarding state in order to forward the packet. The block and secure bits determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state. The transmit port must be in the Forwarding state regardless. The forward state test used in the ingress process is determined by the destination address packet type (multicast/unicast).

In general, packets received with errors are dropped by the address lookup engine without learning, updating, or touching the address. The error condition and the abort are indicated by the CPGMAC\_SL to the ALE. Packets with errors may be passed to the host (not aborted) by a CPGMAC\_SL port if the port has a set rx\_cmf\_en, rx\_cef\_en, or rx\_csf\_en bit(s).

Error packets that are passed to the host by the CPGMAC\_SL are considered to be bypass packets by the ALE and are sent only to the host. Error packets do not learn, update, or touch addresses regardless of whether they are aborted or sent to the host. Packets with errors received by the host are forwarded as normal.

The following control bits are in the CPGMAC\_SL1/2\_MacControl register.

**rx\_cef\_en:** This CPGMAC\_SL control bit enables frames that are fragments, long, jabber, CRC, code, and alignment errors to be forwarded.

**rx\_csf\_en:** This CPGMAC\_SL bit enables short frames to be forwarded.

**rx\_cmf\_en:** This CPGMAC\_SL control bit enables mac control frames to be forwarded.

#### 14.3.2.7.2.1 Ingress Filtering Process

```

If (Rx port_state is Disabled)
then discard the packet

if (directed packet)
then use directed port number and go to Egress process

if ((ale_bypass or error packet) and (host port is not the receive port))
then use host portmask and go to Egress process

if (((block) and (unicast source address found)) or ((block) and (unicast destination address found)))
then discard the packet

if ((enable_rate_limit) and (rate limit exceeded) and (not rate_limit_tx))
then if (((mcast/bcast destination address found) and (not super)) or (mcast/bcast destination address not found))
      then discard the packet

if ((not forward state test valid) and (destination address found))
then discard the packet to any port not meeting the requirements
  • Unicast destination addresses use the unicast forward state test and multicast destination addresses use the multicast forward state test.

if ((destination address not found) and ((not transmit port forwarding) or (not receive port forwarding)))
then discard the packet to any ports not meeting the above requirements

if (source address found) and (secure) and (receive port number != port_number)
then discard the packet

if ((not super) and (drop_untagged) and ((non-tagged packet) or ((priority tagged) and not(en_vid0_mode)))
then discard the packet

If (VLAN_Unaware)
  force_untagged_egress = "000000"
  reg_mcast_flood_mask = "111111"
  unreg_mcast_flood_mask = "111111"
  vlan_member_list = "111111"
else if (VLAN not found)
  force_untagged_egress = unknown_force_untagged_egress
  reg_mcast_flood_mask = unknown_reg_mcast_flood_mask
  unreg_mcast_flood_mask = unknown_unreg_mcast_flood_mask
  vlan_member_list = unknown_vlan_member_list
else
  force_untagged_egress = found_force_untagged_egress
  reg_mcast_flood_mask = found_reg_mcast_flood_mask
  unreg_mcast_flood_mask = found_unreg_mcast_flood_mask
  vlan_member_list = found_vlan_member_list

if ((not super) and (vid_ingress_check) and (Rx port is not VLAN member))
then discard the packet

```

```

if ((enable_auth_mode) and (source address not found) and not(destination address found and (super)))
then discard the packet

if (destination address equals source address)
then discard the packet

if (vlan_aware) goto VLAN_Aware_Lookup process
else goto VLAN_Unaware_Lookup process

```

#### **14.3.2.7.2.2 VLAN\_Aware Lookup Process**

```

if ((unicast packet) and (destination address found with or without VLAN) and dlr_unicast)
then portmask is the vlan_member_list less the host port
and goto Egress process

if ((unicast packet) and (destination address found with or without VLAN) and (not super))
then portmask is the logical "AND" of the port_number and the vlan_member_list and goto Egress process

if ((unicast packet) and (destination address found with or without VLAN) and (super))
then portmask is the port_number and goto Egress process

if (Unicast packet) # destination address not found
then portmask is vlan_member_list less host port and goto Egress process

if ((Multicast packet) and (destination address found with or without VLAN) and (not super))
then portmask is the logical "AND" of reg_mcast_flood_mask and found destination address/VLAN portmask (port_mask) and
vlan_member_list and goto Egress process

if ((Multicast packet) and (destination address found with or without VLAN) and (super))
then portmask is the port_mask and goto Egress process

if (Multicast packet) # destination address not found
then portmask is the logical "AND" of unreg_mcast_flood_mask and vlan_member_list
then goto Egress process

if (Broadcast packet)
then use found vlan_member_list and goto Egress process

```

#### **14.3.2.7.2.3 VLAN\_Unaware Lookup Process**

```

if ((unicast packet) and (destination address found with or without VLAN) and dlr_unicast)
then portmask is the vlan_member_list less the host port
and goto Egress process

if ((unicast packet) and (destination address found with or without VLAN) and (not super))
then portmask is the logical "AND" of the port_number and the vlan_member_list and goto Egress process

if ((unicast packet) and (destination address found with or without VLAN) and (super))
then portmask is the port_number and goto Egress process

if (Unicast packet) # destination address not found
then portmask is vlan_member_list less host port and goto Egress process

if ((Multicast packet) and (destination address found with or without VLAN) and (not super))
then portmask is the logical "AND" of reg_mcast_flood_mask and found destination address/VLAN portmask (port_mask) and
vlan_member_list and goto Egress process

if ((Multicast packet) and (destination address found with or without VLAN) and (super))
then portmask is the port_mask and goto Egress process

if (Multicast packet) # destination address not found
then portmask is the logical "AND" of unreg_mcast_flood_mask and vlan_member_list
then goto Egress process

```

|  |
|--|
| if (Broadcast packet)<br>then use found vlan_member_list and goto Egress process |
|--|

#### **14.3.2.7.2.4 Egress Process**

|   |
|---|
| Clear Rx port from portmask (don't send packet to Rx port).   |
| Clear disabled ports from portmask.   |
| if ((enable_oui_deny) and (OUI source address not found) and (not ale_bypass) and (not error packet) and not ((mcast destination address) and (super)))<br>then Clear host port from portmask                   |
| if ((enable_rate_limit) and (rate_limit_tx))<br>then if (not super) and (rate limit exceeded on any tx port)<br>then clear rate limited tx port from portmask<br>If address not found then super cannot be set. |
| If portmask is zero then discard packet   |
| Send packet to portmask ports.  |

#### **14.3.2.7.3 Learning/Updating/Touching Processes**

The learning, updating, and touching processes are applied to each receive packet that is not aborted. The processes are concurrent with the packet forwarding process. In addition to the following, a packet must be received without error in order to learn/update/touch an address.

##### **14.3.2.7.3.1 Learning Process**

|  |
|--|
| If (not(Learning or Forwarding) or (enable_auth_mode) or (packet error) or (no_learn))<br>then do not learn address        |
| if ((Non-tagged packet) and (drop_untagged))<br>then do not learn address  |
| if ((vlan_aware) and (VLAN not found) and (unknown_vlan_member_list = "000"))<br>then do not learn address                 |
| if ((vid_ingress_check) and (Rx port is not VLAN member) and (VLAN found))<br>then do not learn address                    |
| if ((source address not found) and (vlan_aware) and not(learn_no_vid))<br>then learn address with VLAN                     |
| if ((source address not found) and ((not vlan_aware) or (vlan_aware and learn_no_vid)))<br>then learn address without VLAN |

##### **14.3.2.7.3.2 Updating Process**

|  |
|--|
| if (dlr_unicast)<br>then do not update address   |
| If (not(Learning or Forwarding) or (enable_auth_mode) or (packet error) or (no_sa_update))<br>then do not update address |
| if ((Non-tagged packet) and (drop_untagged))<br>then do not update address   |
| if ((vlan_aware) and (VLAN not found) and (unknown_vlan_member_list = "000"))<br>then do not update address              |

```

if ((vid_ingress_check) and (Rx port is not VLAN member) and (VLAN found))
then do not update address

if ((source address found) and (receive port number != port_number) and (secure or block))
then do not update address

if ((source address found) and (receive port number != port_number))
then update address

```

#### 14.3.2.7.3.3 Touching Process

```

if ((source address found) and (ageable) and (not touched))
then set touched

```

#### 14.3.2.8 Packet Priority Handling

Packets are received on three ports, two of which are CPGMAC\_SL Ethernet ports and the third port is the CPPI host port. Received packets have a received packet priority (0 to 7 with 7 being the highest priority). The received packet priority is determined as shown:

1. If the first packet LTYPE = 0x8100 then the received packet priority is the packet priority (VLAN tagged and priority tagged packets).
2. If the first packet LTYPE = 0x0800 and byte 14 (following the LTYPE) is equal to 0x45 then the received packet priority is the 6-bit TOS field in byte 15 (upper 6-bits) mapped through the port's DSCP priority mapping register (IPV4 packet).
3. The received packet priority is the source (ingress) port priority (untagged non-IPV4 packet).

The received packet priority is mapped through the receive ports associated "packet priority to header packet priority mapping register" to obtain the header packet priority (the CPDMA Rx and Tx nomenclature is reversed from the CPGMAC\_SL nomenclature for legacy reasons). The header packet priority is mapped through the "header priority to switch priority mapping register" to obtain the hardware switch priority (0 to 3 with 3 being the highest priority). The header packet priority is then used as the actual transmit packet priority if the VLAN information is to be sent on egress.

#### 14.3.2.9 FIFO Memory Control

Each of the three CPSW\_3G ports has an identical associated FIFO. Each FIFO contains a single logical receive (ingress) queue and four logical transmit queues (priority 0 through 3). Each FIFO memory contains 20,480 bytes (20k) total organized as 2560 by 64-bit words contained in a single memory instance. The FIFO memory is used for the associated port transmit and receive queues. The tx\_max\_blk field in the FIFO's associated Max\_Blk register determines the maximum number of 1k FIFO memory blocks to be allocated to the four logical transmit queues (transmit total).

The rx\_max\_blk field in the FIFO's associated Max\_Blk register determines the maximum number of 1k memory blocks to be allocated to the logical receive queue. The tx\_max\_blk value plus the rx\_max\_blk value must sum to 20 (the total number of blocks in the FIFO). If the sum were less than 20 then some memory blocks would be unused. The default is 17 (decimal) transmit blocks and three receive blocks. The FIFO's follow the naming convention of the Ethernet ports. Host Port is Port0 and External Ports are Port1,2

#### 14.3.2.10 FIFO Transmit Queue Control

There are four transmit queues in each transmit FIFO. Software has some flexibility in determining how packets are loaded into the queues and on how packet priorities are selected for transmission (how packets are removed and transmitted from queues). All ports on the switch have identical FIFO's. For the purposes of the below the transmit FIFO is switch egress even though the port 0 transmit FIFO is connected to the CPDMA receive (also switch egress). The CPDMA nomenclature is reversed from the CPGMAC\_SL nomenclature due to legacy reasons.

#### **14.3.2.10.1 Normal Priority Mode**

When operating in normal mode, lower priority frames are dropped before higher priority frames. The intention is to give preference to higher priority frames. Priority 3 is the highest priority and is allowed to fill the FIFO. Priority 2 will drop packets if the packet is going to take space in the last 2k available. Priority 1 will drop packets if the packet is going to take space in the last 4k available. Priority 0 will drop packets if the packet is going to take space in the last 6k available. If fewer than 4 priorities are to be implemented then the priorities should be mapped such that the highest priorities are used.

For example, if two priorities are going to be used then all packets should be mapped to priorities 3 and 2 and priorities 1 and 0 should be unused. Priority escalation may be used in normal priority mode if desired. Normal priority mode is configured as described below:

- Select normal priority mode by setting tx\_in\_sel[1:0] = 00 for all ports (default value in P0/1/2\_Tx\_In\_Ctl)
- Configure priority mapping to use only the highest priorities if less than 4 priorities are used. Refer to the Packet Priority Handling section of this chapter.

#### **14.3.2.10.2 Dual Mac Mode**

When operating in dual mac mode the intention is to transfer packets between ports 0 and 1 and ports 0 and 2, but not between ports 1 and 2. Each CPGMAC\_SL appears as a single MAC with no bridging between MAC's. Each CPGMAC\_SL has at least one unique (not the same) mac address.

Dual mac mode is configured as described below:

- Set the ale\_vlan\_aware bit in the ALE\_Control register. This bit configures the ALE to process in vlan aware mode. The CPSW\_3G vlan aware bit (vlan\_aware in CPSW\_Control) determines how packets VLAN's are processed on CPGMAC\_SL egress and does not affect how the ALE processes packets or the packet destination. The CPSW\_3G vlan aware bit may be set or not as required (must be set if VLAN's are to exit the switch).
- **Configure the Port 1 to Port 0 VLAN**

Add a VLAN Table Entry with ports 0 and 1 as members (clear the flood masks).

Add a VLAN/Unicast Address Table Entry with the Port1/0 VLAN and a port number of 0. Packets received on port 1 with this unicast address will be sent only to port 0 (egress). If multiple mac addresses are desired for this port then multiple entries of this type may be configured.

- **Configure the Port 2 to Port 0 VLAN**

Add a VLAN Table Entry with ports 0 and 2 as members (clear the flood masks).

Add a VLAN/Unicast Address Table Entry with the Port2/0 VLAN and a port number of 0. Packets received on port 2 with this unicast address will be sent only to port 0 (egress). If multiple mac addresses are desired for this port then multiple entries of this type may be configured.

- Packets from the host (port 0) to ports 1 and 2 should be directed. If directed packets are not desired then VLAN with addresses can be added for both destination ports.
- Select the dual mac mode on the port 0 FIFO by setting tx\_in\_sel[1:0] = 01 in P0\_Tx\_In\_Ctl. The intention of this mode is to allow packets from both ethernet ports to be written into the FIFO without one port starving the other port.
- The priority levels may be configured such that packets received on port 1 egress on one CPDMA RX channel while packets received on port 2 egress on a different CPDMA RX channel.

#### **14.3.2.10.3 Rate Limit Mode**

Rate-limit mode is intended to allow some CPDMA transmit (switch ingress) channels and some CPGMAC\_SL FIFO priorities (switch egress) to be rate-limited. Non rate-limited traffic (bulk traffic) is allowed on non rate-limited channels and FIFO priorities. The bulk traffic does not impact the rate-limited traffic. Rate-limited traffic must be configured to be sent to rate-limited queues (via packet priority handling).

The allocated rates for rate-limited traffic must not be oversubscribed. For example, if port 1 is sending 15% rate limited traffic to port 2 priority 3, and port 0 is also sending 10% rate-limited traffic to port 2 priority 3, then the port 2 priority 3 egress rate must be configured to be 25% plus a percent or two for margin. The switch must be configured to allow some percentage of non rate-limited traffic. Non-rate-limited traffic must be configured to be sent to non rate-limited queues. No packets from the host should be dropped, but non rate-limited traffic received on an ethernet port can be dropped. Rate-limited mode is configured as shown:

- Set tx\_in\_sel[1:0] = 10 in P1/2\_Tx\_In\_Ctl to enable ports 1 and 2 transmit FIFO inputs to be configured for rate-limiting queues. Enabling a queue to be rate-limiting with this field affects only the packet being loaded into the FIFO, it does not configure the transmit for queue shaping.
  - Configure the number of rate-limited queues for port 1 and 2 transmit FIFO's by setting the tx\_rate\_en[3:0] field in P1/2\_Tx\_In\_Ctl. Rate limited queues must be the highest number. For example, if there are two rate limited queues then 1100 would be written to this field for priorities 3 and 2. This field enables the FIFO to allow rate-limited traffic into rate-limited queues while discriminating against non rate-limited queues.
  - Set p1\_priN\_shape\_en and p2\_priN\_shape\_en in the CPSW\_3G\_PTYPE register. These bits determine which queues actually shape the output data stream. In general, the same priorities that are set in tx\_rate\_en are set in these bits as well, but the FIFO input and output enable bits are separate to allow rate-limiting from the host to non shaped channels if desired.
- When queue shaping is not enabled for a queue then packets are selected for egress based on priority. When queue shaping is enabled then packets are selected for egress based on queue percentages. If shaping is required on a single queue then it must be priority 3 (priorities 2, 1 and 0 are strict priority). If shaping is required on two queues then it must be on priorities 2 and 3 (priorities 1 and 0 are strict priority). If shaping is required on three queues then it must be priorities 3, 2, and 1 (priority 0 would then get the leftovers). Priority shaping follows the requirements in the IEEE P802.1Qav/D6.0 specification. Priority shaping is not compatible with priority escalation (escalation must be disabled).
- P0\_Tx\_In\_Ctl[1:0] should remain at the default 00 value. Port 0 egress (CPDMA RX) should not be rate-limited.
  - The CPDMA is configured for rate-limited transmit (switch ingress) channels by setting the highest bits of the tx\_rlim[7:0] field in the CPDMA DMA\_Control register. If there are two rate limited channels then tx\_rlim[7:0] = 11000000 (the rate limited channels must be the highest priorities). Also, tx\_ptype in the DMA\_Control register must be set (fixed priority mode). Rate limited channels must go to rate-limited FIFO queues, and the FIFO queue rate must not be oversubscribed.

#### 14.3.2.11 Packet Padding

VLAN tagged ingress packets of 64 to 67-bytes will be padded to 64-bytes on egress (all ports) if the VLAN is removed on egress.

#### 14.3.2.12 Flow Control

There are two types of switch flow control – CPPI port flow control and Ethernet port flow control. The CPPI and Ethernet port naming conventions for data flow into and out of the switch are reversed. For the CPPI port (port 0), transmit operations move packets from external memory into the switch and then out to either or both Ethernet transmit ports (ports 1 and 2). CPPI receive operations move packets that were received on either or both Ethernet receive ports to external memory.

##### 14.3.2.12.1 CPPI Port Flow Control

The CPPI port has flow control available for transmit (switch ingress). CPPI receive operations (switch egress) do not require flow control. CPPI Transmit flow control is initiated when enabled and triggered. CPPI transmit flow control is enabled by setting the p0\_flow\_en bit in the **CPSW\_Flow\_Control** register. CPPI transmit flow control is enabled by default on reset because host packets should not be dropped in any mode of operation.

#### **14.3.2.12.2 Ethernet Port Flow Control**

The Ethernet ports have flow control available for transmit and receive. Transmit flow control stops the Ethernet port from transmitting packets to the wire (switch egress) in response to a received pause frame. Transmit flow control does not depend on FIFO usage.

The ethernet ports have flow control available for receive operations (packet ingress). Ethernet port receive flow control is initiated when enabled and triggered. Packets received on an ethernet port can be sent to the other ethernet port or the CPPI port (or both). Each destination port can trigger the receive ethernet port flow control. An ethernet destination port triggers another ethernet receive flow control when the destination port is full.

When a packet is received on an ethernet port interface with enabled flow control the below occurs:

- The packet will be sent to all ports that currently have room to take the entire packet.
- The packet will be retried until successful to all ports that indicate they don't have room for the packet.

The flow control trigger to the CPGMAC\_SL will be asserted until the packet has been sent, and there is room in the logical receive FIFO for packet runout from another flow control trigger (**rx\_pkt\_cnt** = 0). Ethernet port receive flow control is disabled by default on reset. Ethernet port receive flow control requires that the **rx\_flow\_en** bit in the associated CPGMAC\_SL be set to one.

When receive flow control is enabled on a port, the port's associated FIFO block allocation must be adjusted. The port RX allocation must increase from the default three blocks to accommodate the flow control runout. A corresponding decrease in the TX block allocation is required. If a sending port ignores a pause frame then packets may overrun on receive (and be dropped) but will not be dropped on transmit. If flow control is disabled for gmii ports, then any packets that are dropped are dropped on transmit and not on receive.

##### **14.3.2.12.2.1 Receive Flow Control**

When enabled and triggered, receive flow control is initiated to limit the CPGMAC\_SL from further frame reception. Half-duplex mode receive flow control is collision based while full duplex mode issues 802.3X pause frames. In either case, receive flow control prevents frame reception by issuing the flow control appropriate for the current mode of operation. Receive flow control is enabled by the **rx\_flow\_en** bit in the **MacControl** register. Receive flow control is triggered (when enabled) when the **RX\_FLOW\_TRIGGER** input is asserted. The CPGMAC\_SL is configured for collision or IEEE 802.3X flow control via the **fullduplex** bit in the **MacControl** register.

###### **14.3.2.12.2.1.1 Collision Based Receive Buffer Flow Control**

Collision-based receive buffer flow control provides a means of preventing frame reception when the port is operating in half-duplex mode (**fullduplex** is cleared in **MacControl**). When receive flow control is enabled and triggered, the port will generate collisions for received frames. The jam sequence transmitted will be the twelve byte sequence C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3.C3 (hex). The jam sequence will begin no later than approximately as the source address starts to be received. Note that these forced collisions will not be limited to a maximum of 16 consecutive collisions, and are independent of the normal back-off algorithm. Receive flow control does not depend on the value of the incoming frame destination address. A collision will be generated for any incoming packet, regardless of the destination address.

###### **14.3.2.12.2.1.2 IEEE 802.3X Based Receive Flow Control**

IEEE 802.3x based receive flow control provides a means of preventing frame reception when the port is operating in full-duplex mode (**fullduplex** is set in **MacControl**). When receive flow control is enabled and triggered, the port will transmit a pause frame to request that the sending station stop transmitting for the period indicated within the transmitted pause frame.

The CPGMAC\_SL will transmit a pause frame to the reserved multicast address at the first available opportunity (immediately if currently idle, or following the completion of the frame currently being transmitted). The pause frame will contain the maximum possible value for the pause time (0xFFFF). The MAC will count the receive pause frame time (decrements 0xFF00 down to zero) and retransmit an outgoing pause frame if the count reaches zero. When the flow control request is removed, the MAC will transmit a pause frame with a zero pause time to cancel the pause request.

Note that transmitted pause frames are only a request to the other end station to stop transmitting. Frames that are received during the pause interval will be received normally (provided the Rx FIFO is not full).

Pause frames will be transmitted if enabled and triggered regardless of whether or not the port is observing the pause time period from an incoming pause frame.

The CPGMAC\_SL will transmit pause frames as described below:

- The 48-bit reserved multicast destination address 01.80.C2.00.00.01.
- The 48-bit source address — SL\_SA(47:0).
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit pause time value FF.FF. A pause-quantum is 512 bit-times. Pause frames sent to cancel a pause request will have a pause time value of 00.00.
- Zero padding to 64-byte data length (The CPGMAC\_SL will transmit only 64 byte pause frames).
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

If **rx\_flow\_en** is cleared to zero while the pause time is nonzero, then the pause time will be cleared to zero and a zero count pause frame will be sent.

#### 14.3.2.12.2.2 Transmit Flow Control

Incoming pause frames are acted upon, when enabled, to prevent the CPGMAC\_SL from transmitting any further frames. Incoming pause frames are only acted upon when the **fullduplex** and **tx\_flow\_en** bits in the **MacControl** register are set. Pause frames are not acted upon in half-duplex mode. Pause frame action will be taken if enabled, but normally the frame will be filtered and not transferred to memory.

MAC control frames will be transferred to memory if the **rx\_cmf\_en** (Copy MAC Frames) bit in the **MacControl** register is set. The **tx\_flow\_en** and **fullduplex** bits effect whether or not MAC control frames are acted upon, but they have no effect upon whether or not MAC control frames are transferred to memory or filtered.

Pause frames are a subset of MAC Control Frames with an opcode field=0x0001. Incoming pause frames will only be acted upon by the port if:

- **tx\_flow\_en** is set in **MacControl**, and
- the frame's length is 64 to **rx\_maxlen** bytes inclusive, and
- the frame contains no crc error or align/code errors.

The pause time value from valid frames will be extracted from the two bytes following the opcode. The pause time will be loaded into the port's transmit pause timer and the transmit pause time period will begin.

If a valid pause frame is received during the transmit pause time period of a previous transmit pause frame then:

- if the destination address is not equal to the reserved multicast address or any enabled or disabled unicast address, then the transmit pause timer will immediately expire, or
- if the new pause time value is zero then the transmit pause timer will immediately expire, else
- the port transmit pause timer will immediately be set to the new pause frame pause time value. (Any remaining pause time from the previous pause frame will be discarded).

If **tx\_flow\_en** in **MacControl** is cleared, then the pause-timer will immediately expire.

The port will not start the transmission of a new data frame any sooner than 512-bit times after a pause frame with a non-zero pause time has finished being received (**MRXDV** going inactive). No transmission will begin until the pause timer has expired (the port may transmit pause frames in order to initiate outgoing flow control). Any frame already in transmission when a pause frame is received will be completed and unaffected.

Incoming pause frames consist of the below:

- A 48-bit destination address equal to:
- The reserved multicast destination address 01.80.C2.00.00.01, or
- The SL\_SA(47:0) input mac source address.
- The 48-bit source address of the transmitting device.
- The 16-bit length/type field containing the value 88.08
- The 16-bit pause opcode equal to 00.01
- The 16-bit pause\_time. A pause-quantum is 512 bit-times.
- Padding to 64-byte data length.
- The 32-bit frame-check sequence (CRC word).

All quantities above are hexadecimal and are transmitted most-significant byte first. The least-significant bit is transferred first in each byte.

The padding is required to make up the frame to a minimum of 64 bytes. The standard allows pause frames longer than 64 bytes to be discarded or interpreted as valid pause frames. The CPGMAC\_SL will recognize any pause frame between 64 bytes and **rx\_maxlen** bytes in length.

#### 14.3.2.13 Packet Drop Interface

The packet drop interface supports an external packet drop engine. The port 1 (and port 2) CPGMAC\_SL receive FIFO VBUSP interface signals are CPSW\_3G outputs. The receive packet interface has an associated packet drop input P1\_RFIFO\_DROP (P2\_RFIFO\_DROP). An external packet drop engine may "snoop" the received packet header and data to determine whether or not the packet should be dropped.

If the packet is to be dropped the external logic must assert the drop signal by no later than the second clock after the end of packet (or abort) indication from the CPGMAC\_SL. The drop signal should remain asserted until the second clock after the end of packet (or abort) indication. If the packet is not to be dropped then the drop signal should remain deasserted. The CPGMAC\_SL section contains more information on the receive FIFO VBUSP interface signals and end of packet indication.

#### 14.3.2.14 Short Gap

The port 1 (and port 2) transmit inter-packet gap (IPG) may be shortened by eight bit times when enabled and triggered. The **tx\_short\_gap\_en** bit in the **SL1\_MacControl (SL2\_MacControl)** register enables the gap to be shortened when triggered. The condition is triggered when the port 1 (port 2) transmit FIFO has a user defined number of FIFO blocks used. The port 1 transmit FIFO blocks used determines if the port 1 gap is shortened, and the port 2 transmit FIFO blocks used determines if the port 2 gap is shortened. The **CPSW\_Gap\_Thresh** register value determines the port 1 short gap threshold, and the **CPSW\_Gap\_Thresh** register value determines the port 2 short gap threshold.

#### 14.3.2.15 Switch Latency

The CPSW\_3G is a store and forward switch. The switch latency is defined as the amount of time between the end of packet reception of the received packet to the start of the output packet transmit.

| Mode       | Latency |
|------------|---------|
| Gig (1000) | 880ns   |
| 100        | 1.3us   |
| 10         | 6.5us   |

#### 14.3.2.16 Emulation Control

The emulation control input (EMUSU) and submodule emulation control registers allow CPSW\_3G operation to be completely or partially suspended. There are three CPSW\_3G submodules that contain emulation control registers (CPGMAC\_SL1, CPGMAC\_SL2, and CPDMA). The submodule emulation control registers must be accessed to facilitate CPSW\_3G emulation control. The CPSW\_3G module enters the emulation suspend state if all three submodules are configured for emulation suspend and the emulation suspend input is asserted.

A partial emulation suspend state is entered if one or two submodules are configured for emulation suspend and the emulation suspend input is asserted. Emulation suspend occurs at packet boundaries. The emulation control feature is implemented for compatibility with other peripherals.

##### CPGMAC\_SL Emulation Control

The emulation control input (**TBEMUSUP**) and register bits (**soft** and **free** in the **EMControl** register) allow CPGMAC\_SL operation to be suspended. When the emulation suspend state is entered, the CPGMAC\_SL will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For receive, frames that are detected by the CPGMAC\_SL after the suspend state is entered are ignored. Emulation control is implemented for compatibility with other peripherals.

##### CPDMA Emulation Control

The emulation control input (**TBEMUSUP**) and register bits (**soft** and **free** in the **EMControl** register) allow CPDMA operation to be suspended. When the emulation suspend state is entered, the CPDMA will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. For transmission, any complete or partial frame in the tx cell fifo will be transmitted. For receive, frames that are detected by the CPDMA after the suspend state is entered are ignored. No statistics will be kept for ignored frames. Emulation control is implemented for compatibility with other peripherals

The following table shows the operations of the emulation control input and register bits:

**Table 14-18. Operations of Emulation Control Input and Register Bits**

| EMUSU | soft | free | Description       |
|-------|------|------|-------------------|
| 0     | X    | X    | Normal Operation  |
| 1     | 0    | 0    | Normal Operation  |
| 1     | 1    | 0    | Emulation Suspend |
| 1     | X    | 1    | Normal Operation  |

#### 14.3.2.17 Software IDLE

The submodule software idle register bits enable CPSW\_3G operation to be completely or partially suspended by software control. There are three CPSW\_3G submodules that contain software idle register bits (CPGMAC\_SL1, CPGMAC\_SL2, and CPDMA). Each of the three submodules may be individually commanded to enter the idle state. The idle state is entered at packet boundaries, and no further packet operations will occur on an idled submodule until the idle command is removed. The CPSW\_3G module enters the idle state when all three submodules are commanded to enter and have entered the idle state. Idle status is determined by reading or polling the three submodule idle bits. The CPSW\_3G is in the idle state when all three submodules are in the idle state. The **CPSW\_Soft\_Idle** bit may be set if desired after the submodules are in the idle state. The **CPSW\_Soft\_Idle** bit causes packets to not be transferred from one FIFO to another FIFO internal to the switch.

#### 14.3.2.18 Software Reset

The CPSW\_3G software reset register, CPSW\_3GSS software reset register and the three submodule software reset registers enable the CPSW\_3GSS to be reset by software. There are three CPSW\_3G submodules that contain software reset registers (CPGMAC\_SL1, CPGMAC\_SL2, and CPDMA). Each of the three submodules may be individually commanded to be reset by software.

For the CPDMA, the reset state is entered at packet boundaries, at which time the CPDMA reset occurs. The CPGMAC\_SL soft reset is immediate. Submodule reset status is determined by reading or polling the submodule reset bit. If the submodule reset bit is read as a one, then the reset process has not yet completed. The submodule soft reset process could take up to 2ms each. The reset has completed if the submodule reset bit is read as a zero.

After all three submodules (in any order) have been reset and a read of each submodule reset bit indicates that the reset process is complete, the CPSW\_3G software reset register bit may be written to complete the CPSW\_3G module software reset operation. The CPSW\_3G software reset bit controls the reset of the FIFO's, the statistics submodule, and the address lookup engine (ALE). The CPSW\_3G software reset is immediate and will be indicated by reading a zero from the soft reset bit.

The CPSW\_3GSS software reset bit controls the reset of the INT, REGS and CPPI. The CPSW\_3GSS software reset is immediate and will be indicated by reading a zero from the soft reset bit.

#### **14.3.2.19 FIFO Loopback**

FIFO loopback mode is entered when the fifo\_loopback bit in the CPSW\_Control register is set. FIFO loopback mode causes packets received on a port to be turned around and transmitted back on the same port. Port 0 receive is fixed on channel zero in FIFO loopback mode. The RXSOFOVERRUN statistic is incremented for each packet sent in FIFO loopback mode. Packets sent in with errors are returned with errors (they are not dropped). FIFO loopback is intended as a simple mechanism for test purposes. FIFO loopback should be performed in fullduplex mode only.

#### **14.3.2.20 CPSW\_3G Network Statistics**

The CPSW\_3G has a set of statistics that record events associated with frame traffic on selected switch ports. The statistics values are cleared to zero 38 clocks after the rising edge of VBUSP\_RST\_N. When one or more port enable bits (stat\_port\_en[2:0]) are set, all statistics registers are write to decrement. The value written will be subtracted from the register value with the result being stored in the register. If a value greater than the statistics value is written, then zero will be written to the register (writing 0xffffffff will clear a statistics location).

When all port enable bits are cleared to zero, all statistics registers are read/write (normal write direct, so writing 0x00000000 will clear a statistics location). All write accesses must be 32-bit accesses. In the below statistics descriptions, "the port" refers to any enabled port (with a corresponding set stat\_port\_en[2:0] bit).

The statistics interrupt (STAT\_PEND) will be issued if enabled when any statistics value is greater than or equal to 0x80000000. The statistics interrupt is removed by writing to decrement any statistics value greater than 0x80000000. The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from 0xFFFFFFFF to 0x00000000.

##### **14.3.2.20.1 Rx-only Statistics Descriptions**

###### **14.3.2.20.1.1 Good Rx Frames (Offset = 0h)**

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Had a length of 64 to rx\_maxlen bytes inclusive
- Had no CRC error, alignment error or code error.

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

###### **14.3.2.20.1.2 Broadcast Rx Frames (Offset = 4h)**

The total number of good broadcast frames received on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for only address 0xFFFFFFFFFFFF

- Had a length of 64 to rx\_maxlen bytes inclusive
- Had no CRC error, alignment error or code error.

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### **14.3.2.20.1.3 Multicast Rx Frames (Offset = 8h)**

The total number of good multicast frames received on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than 0xFFFFFFFFFFFF
- Had a length of 64 to rx\_maxlen bytes inclusive
- Had no CRC error, alignment error or code error

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### **14.3.2.20.1.4 Pause Rx Frames (Offset = Ch)**

The total number of IEEE 802.3X pause frames received by the port (whether acted upon or not). Such a frame:

- Contained any unicast, broadcast, or multicast address
- Contained the length/type field value 88.08 (hex) and the opcode 0x0001
- Was of length 64 to rx\_maxlen bytes inclusive
- Had no CRC error, alignment error or code error
- Pause-frames had been enabled on the port (tx\_flow\_en = 1).

The port could have been in either half or full-duplex mode.

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### **14.3.2.20.1.5 Rx CRC Errors (Offset = 10h)**

The total number of frames received on the port that experienced a CRC error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to rx\_maxlen bytes inclusive
- Had no code/align error,
- Had a CRC error

Overruns have no effect upon this statistic.

A CRC error is defined to be:

- A frame containing an even number of nibbles
- Failing the Frame Check Sequence test

#### **14.3.2.20.1.6 Rx Align/Code Errors (Offset = 14h)**

The total number of frames received on the port that experienced an alignment error or code error. Such a frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was of length 64 to rx\_maxlen bytes inclusive
- Had either an alignment error or a code error

Overruns have no effect upon this statistic.

An alignment error is defined to be:

- A frame containing an odd number of nibbles
- Failing the Frame Check Sequence test if the final nibble is ignored

A code error is defined to be a frame which has been discarded because the port's MRXER pin driven with a one for at least one bit-time's duration at any point during the frame's reception.

Note: RFC 1757 etherStatsCRCAlignErrors Ref. 1.5 can be calculated by summing Rx Align/Code Errors and Rx CRC errors.

#### 14.3.2.20.1.7 Oversize Rx Frames (Offset = 18h)

The total number of oversized frames received on the port. An oversized frame is defined to be:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than rx\_maxlen in bytes
- Had no CRC error, alignment error or code error

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### 14.3.2.20.1.8 Rx Jabbers (Offset = 1Ch)

The total number of jabber frames received on the port. A jabber frame:

- Was any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than rx\_maxlen in bytes
- Had no CRC error, alignment error or code error

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### 14.3.2.20.1.9 Undersize (Short) Rx Frames (Offset = 20h)

The total number of undersized frames received on the port. An undersized frame is defined to be:

- Was any data frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Was greater than rx\_maxlen in bytes
- Had no CRC error, alignment error or code error

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### 14.3.2.20.1.10 Rx Fragments (Offset = 24h)

The total number of frame fragments received on the port. A frame fragment is defined to be:

- Any data frame (address matching does not matter)
- Less than 64 bytes long
- Having a CRC error, an alignment error, or a code error
- Not the result of a collision caused by half duplex, collision based flow control

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### 14.3.2.20.1.11 Rx Start of Frame Overruns (Offset = 84h)

The total number of frames received on the port that had a CPDMA start of frame (SOF) overrun or were dropped by due to FIFO resource limitations. SOF overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than rx\_maxlen bytes)
- The CPDMA had a start of frame overrun or the packet was dropped due to FIFO resource limitations

#### **14.3.2.20.1.12 Rx Middle of Frame Overruns (Offset = 88h)**

The total number of frames received on the port that had a CPDMA middle of frame (MOF) overrun. MOF overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than rx\_maxlen bytes)
- The CPDMA had a middle of frame overrun

#### **14.3.2.20.1.13 Rx DMA Overruns (Offset = 8Ch)**

The total number of frames received on the port that had either a DMA start of frame (SOF) overrun or a DMA MOF overrun. An Rx DMA overrun frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length (including less than 64 bytes and greater than rx\_maxlen bytes)
- The CPGMAC\_SL was unable to receive it because it did not have the DMA buffer resources to receive it (zero head descriptor pointer at the start or during the middle of the frame reception)

CRC errors, alignment errors and code errors have no effect upon this statistic.

#### **14.3.2.20.1.14 Rx Octets (Offset = 30h)**

The total number of bytes in all good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Of length 64 to rx\_maxlen bytes inclusive
- Had no CRC error, alignment error or code error

See the Rx Align/Code Errors and Rx CRC errors statistic descriptions for definitions of alignment, code and CRC errors. Overruns have no effect upon this statistic.

#### **14.3.2.20.1.15 Net Octets (Offset = 80h)**

The total number of bytes of frame data received and transmitted on the port. Each frame counted:

- was any data or MAC control frame destined for any unicast, broadcast or multicast address (address match does not matter)
- Any length (including less than 64 bytes and greater than rx\_maxlen bytes)

Also counted in this statistic is:

- Every byte transmitted before a carrier-loss was experienced
- Every byte transmitted before each collision was experienced, (i.e. multiple retries are counted each time)
- Every byte received if the port is in half-duplex mode until a jam sequence was transmitted to initiate flow control. (The jam sequence was not counted to prevent double-counting)

Error conditions such as alignment errors, CRC errors, code errors, overruns and underruns do not affect the recording of bytes by this statistic.

The objective of this statistic is to give a reasonable indication of ethernet utilization

#### 14.3.2.20.2 Tx-only Statistics Descriptions

The maximum and minimum transmit frame size is software controllable.

##### 14.3.2.20.2.1 Good Tx Frames (Offset = 34h)

The total number of good frames received on the port. A good frame is defined to be:

- Any data or MAC control frame which matched a unicast, broadcast or multicast address, or matched due to promiscuous mode
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

##### 14.3.2.20.2.2 Broadcast Tx Frames (Offset = 38h)

The total number of good broadcast frames received on the port. A good broadcast frame is defined to be:

- Any data or MAC control frame which was destined for only address 0xFFFFFFFFFFFF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

##### 14.3.2.20.2.3 Multicast Tx Frames (Offset = 3Ch)

The total number of good multicast frames received on the port. A good multicast frame is defined to be:

- Any data or MAC control frame which was destined for any multicast address other than 0xFFFFFFFFFFFF
- Any length
- Had no late or excessive collisions, no carrier loss and no underrun

##### 14.3.2.20.2.4 Pause Tx Frames (Offset = 40h)

This statistic indicates the number of IEEE 802.3X pause frames transmitted by the port.

Pause frames cannot underrun or contain a CRC error because they are created in the transmitting MAC, so these error conditions have no effect upon the statistic. Pause frames sent by software will not be included in this count.

Since pause frames are only transmitted in full duplex carrier loss and collisions have no effect upon this statistic.

Transmitted pause frames are always 64 byte multicast frames so will appear in the Tx Multicast Frames and 64octet Frames statistics.

##### 14.3.2.20.2.5 Collisions (Offset = 48h)

This statistic records the total number of times that the port experienced a collision. Collisions occur under two circumstances.

1. When a transmit data or MAC control frame:

- Was destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced a collision. A jam sequence is sent for every non-late collision, so this statistic will increment on each occasion if a frame experiences multiple collisions (and increments on late collisions)

CRC errors have no effect upon this statistic.

2. When the port is in half-duplex mode, flow control is active, and a frame reception begins.

#### **14.3.2.20.2.6 Single Collision Tx Frames (Offset = 4Ch)**

The total number of frames transmitted on the port that experienced exactly one collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced one collision before successful transmission. The collision was not late.

CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.7 Multiple Collision Tx Frames (Offset = 50h)**

The total number of frames transmitted on the port that experienced multiple collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 2 to 15 collisions before being successfully transmitted. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.8 Excessive Collisions (Offset = 54h)**

The total number of frames for which transmission was abandoned due to excessive collisions. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced 16 collisions before abandoning all attempts at transmitting the frame. None of the collisions were late.

CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.9 Late Collisions (Offset = 58h)**

The total number of frames on the port for which transmission was abandoned because they experienced a late collision. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Experienced a collision later than 512 bit-times into the transmission. There may have been up to 15 previous (non-late) collisions which had previously required the transmission to be re-attempted. The Late Collisions statistic dominates over the single, multiple and excessive Collisions statistics - if a late collision occurs the frame will not be counted in any of these other three statistics.

CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.10 Tx Underrun (Offset = 5Ch)**

There should be no transmitted frames that experience underrun.

#### **14.3.2.20.2.11 Deferred Tx Frames (Offset = 44h)**

The total number of frames transmitted on the port that first experienced deferment. Such a frame:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- Had no carrier loss and no underrun
- Experienced no collisions before being successfully transmitted

- Found the medium busy when transmission was first attempted, so had to wait.
- CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.12 Carrier Sense Errors (Offset = 60h)**

The total number of frames received on the port that had a CPDMA middle of frame (MOF) overrun. MOF overrun frame is defined to be:

- Was any data or MAC control frame destined for any unicast, broadcast or multicast address
- Was any size
- The carrier sense condition was lost or never asserted when transmitting the frame (the frame is not retransmitted). This is a transmit only statistic. Carrier Sense is a don't care for received frames.
- Transmit frames with carrier sense errors are sent until completion and are not aborted.

CRC errors have no effect upon this statistic.

#### **14.3.2.20.2.13 Tx Octets (Offset = 64h)**

The total number of bytes in all good frames transmitted on the port. A good frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Was any size
- Had no late or excessive collisions, no carrier loss and no underrun.

#### **14.3.2.20.3 Rx- and Tx-Shared Statistics Descriptions**

##### **14.3.2.20.3.1 Rx + Tx 64 Octet Frames (Offset = 68h)**

The total number of 64-byte frames received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was exactly 64 bytes long. (If the frame was being transmitted and experienced carrier loss that resulted in a frame of this size being transmitted, then the frame will be recorded in this statistic).

CRC errors, code/align errors and overruns do not affect the recording of frames in this statistic.

##### **14.3.2.20.3.2 Rx + Tx 65–127 Octet Frames (Offset = 6Ch)**

The total number of frames of size 65 to 127 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 65 to 127 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

##### **14.3.2.20.3.3 Rx + Tx 128–255 Octet Frames (Offset = 70h)**

The total number of frames of size 128 to 255 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 128 to 255 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

**14.3.2.20.3.4 Rx + Tx 256–511 Octet Frames (Offset = 74h)**

The total number of frames of size 256 to 511 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 256 to 511 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

**14.3.2.20.3.5 Rx + Tx 512–1023 Octet Frames (Offset = 78h)**

The total number of frames of size 512 to 1023 bytes received and transmitted on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 512 to 1023 bytes long

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

**14.3.2.20.3.6 Rx + Tx 1024\_Up Octet Frames (Offset = 7Ch)**

The total number of frames of size 1024 to rx\_maxlen bytes for receive or 1024 up for transmit on the port. Such a frame is defined to be:

- Any data or MAC control frame which was destined for any unicast, broadcast or multicast address
- Did not experience late collisions, excessive collisions, or carrier sense error
- Was 1024 to rx\_maxlen bytes long on receive, or any size on transmit

CRC errors, code/align errors, underruns and overruns do not affect the recording of frames in this statistic.

**Table 14-19. Rx Statistics Summary**

| Rx Statistic         | Frame/<br>Oct | Rx/<br>Rx+Tx | Frame Type     |                    |                |                | Frame Size (bytes) |    |            |             |             |              |                        |                | Event         |              |                |              |                |   |   |
|----------------------|---------------|--------------|----------------|--------------------|----------------|----------------|--------------------|----|------------|-------------|-------------|--------------|------------------------|----------------|---------------|--------------|----------------|--------------|----------------|---|---|
|                      |               |              | MAC control    |                    | Data           |                | <64                | 64 | 65-<br>127 | 128-<br>255 | 256-<br>511 | 512-<br>1023 | 1024-<br>rx_<br>maxlen | >rx_<br>maxlen | Flow<br>Coll. | CRC<br>Error | Align/<br>Code | Over-<br>run | Addr.<br>Disc. |   |   |
|                      |               |              | Pause<br>frame | Non-<br>paus-<br>e | Multi-<br>cast | Broad-<br>cast | Uni-<br>cast       |    |            |             |             |              |                        |                |               |              |                |              |                |   |   |
| Good Rx Frames       | F             | Rx           | (y             | y                  | y              | y              | y)                 | n  | (y         | y           | y           | y            | y                      | y              | n             | -            | n              | n            | -              | n |   |
| Broadcast Rx Frames  | F             | Rx           | (%             | %                  | n              | y)             | n                  | n  | (y         | y           | y           | y            | y                      | y              | n             | -            | n              | n            | -              | n |   |
| Multicast Rx Frames  | F             | Rx           | (%             | %                  | y)             | n              | n                  | n  | (y         | y           | y           | y            | y                      | y              | n             | -            | n              | n            | -              | n |   |
| Pause Rx Frames      | F             | Rx           | y              | n                  | n              | n              | n                  | (y | y          | y           | y           | y            | y                      | y              | n             | -            | n              | n            | -              | - |   |
| Rx CRC Errors        | F             | Rx           | (y             | y                  | y              | y              | y)                 | n  | (y         | y           | y           | y            | y                      | y              | n             | -            | y              | n            | -              | n |   |
| Rx Align/Code Errors | F             | Rx           | (y             | y                  | y              | y              | y)                 | n  | (y         | y           | y           | y            | y                      | y              | n             | -            | -              | y            | -              | n |   |
| Oversized Rx Frames  | F             | Rx           | (y             | y                  | y              | y              | y)                 | n  | n          | n           | n           | n            | n                      | n              | y             | -            | n              | n            | -              | n |   |
| Rx Jabbers           | F             | Rx           | (y             | y                  | y              | y              | y)                 | n  | n          | n           | n           | n            | n                      | n              | y             | -            | (y             | y)           | -              | n |   |
| Undersized Rx Frames | F             | Rx           | n              | n                  | (y             | y              | y)                 | y  | n          | n           | n           | n            | n                      | n              | n             | -            | n              | n            | -              | n |   |
| Rx Fragments         | F             | Rx           | n              | n                  | (y             | y              | y)                 | y^ | n          | n           | n           | n            | n                      | n              | n             | -            | (y             | y)           | -              | - |   |
| Rx Overruns          | F             | Rx           | (y             | y                  | y              | y              | y)                 | (y | y          | y           | y           | y            | y                      | y              | y             | -            | -              | -            | y              | n |   |
| 64octet Frames       | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | y          | n           | n           | n            | n                      | n              | n             | -            | -              | -            | -              | - | n |
| 65-127octet Frames   | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | n          | y           | n           | n            | n                      | n              | n             | -            | -              | -            | -              | - | n |
| 128-255octet Frames  | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | n          |             | y           | n            | n                      | n              | n             | -            | -              | -            | -              | - | n |
| 256-511octet Frames  | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | n          | n           | y           | n            | n                      | n              | n             | -            | -              | -            | -              | - | n |
| 512-1023octet Frames | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | n          | n           | n           | y            | n                      | n              | n             | -            | -              | -            | -              | - | n |
| 1024-UPoctet Frames  | F             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | n  | n          | n           | n           | n            | n                      | n              | y             | n            | -              | -            | -              | - | n |
| Rx Octets            | O             | Rx           | (y             | y                  | y              | y              | y)                 | n  | (y         | y           | y           | y            | y                      | y              | y             | n            | -              | n            | n              | - | n |
| Net Octets           | O             | Rx+Tx        | (y             | y                  | y              | y              | y)                 | (y | y          | y           | y           | y            | y                      | y              | y             | y            | y              | y            | -              | - | - |

**Notes for the Rx Statistics Summary:**

1. "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
2. "-" indicates conditions which are ignored in the formations of the statistic.
3. Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
4. The non-pause column refers to all MAC control frames (i.e. frames with length/type=88.08) with opcodes other than 0x0001. The pauseframe column refers to MAC frames with the opcode=0x0001.
5. The multicast, broadcast and unicast columns in the table refer to non-MAC Control/non-pause frames (i.e. data frames).
6. "%" If either a MAC control frame or pause frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
7. "y^" Frame fragments are not counted if less than 8 bytes.
8. flow coll. are half-duplex collisions forced by the MAC to achieve flow-control. A collision will be forced during the first 8 bytes so should not show in frame fragments. Some of the '-'s in this column might in reality be 'n's.
9. The rx\_overruns stat show above is for rx\_mof\_overruns and rx\_sof\_overruns added together.

**Table 14-20. Tx Statistics Summary**

| Tx Statistic                 | Frame/<br>Oct | Tx/<br>Rx+<br>Tx | Frame Type         |                  |                    |                    |                  | Frame Size (bytes) |            |             |             |              |               |           |              | Event          |          |    |      |   |                   |            |              |                   |   |   |
|------------------------------|---------------|------------------|--------------------|------------------|--------------------|--------------------|------------------|--------------------|------------|-------------|-------------|--------------|---------------|-----------|--------------|----------------|----------|----|------|---|-------------------|------------|--------------|-------------------|---|---|
|                              |               |                  | MAC control        |                  | Data               |                    |                  | 64                 | 65-<br>127 | 128-<br>255 | 256-<br>511 | 512-<br>1023 | 1024-<br>1535 | ><br>1535 | CRC<br>Error | Collision Type |          |    |      |   | No<br>Carri<br>er | Que<br>ued | Defe<br>rred | Und<br>er-<br>run |   |   |
|                              |               |                  | Pause<br>(MA<br>C) | Any<br>(CP<br>U) | Multi<br>-<br>cast | Broad<br>-<br>cast | Uni<br>-<br>cast |                    |            |             |             |              |               |           | Flow         | 1              | 2-<br>15 | 16 | Late |   |                   |            |              |                   |   |   |
| Good Tx Frames               | F             | Tx               | (y                 | y                | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | -        | -  | -    | n | n                 | n          | -            | -                 | n |   |
| Broadcast Tx Frames          | F             | Tx               | n                  | (%               | n                  | y)                 | n                | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | -        | -  | -    | n | n                 | n          | -            | -                 | n |   |
| Multicast Tx Frames          | F             | Tx               | (y                 | %                | y)                 | n                  | n                | y                  | y          | y           | y           | y            | y             | y         | y)           | -              | -        | -  | -    | n | n                 | n          | -            | -                 | n |   |
| Pause Tx Frames              | F             | Tx               | y                  | n                | n                  | n                  | n                | y                  | n          | n           | n           | n            | n             | n         | n            | -              | -        | -  | -    | - | -                 | -          | -            | -                 | - |   |
| Collisions                   | F             | Tx               | n                  | (y               | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | (+       | +  | +    | + | +                 | )          | n            | -                 | - | - |
| Single Collision Tx Frames   | F             | Tx               | n                  | (y               | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | -        | y  | n    | n | n                 | n          | -            | -                 | - | - |
| Multiple Collision Tx Frames | F             | Tx               | n                  | (y               | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | -        | n  | y    | n | n                 | n          | -            | -                 | - | - |
| Excessive Collisions         | F             | Tx               | n                  | (y               | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y)           | -              | -        | n  | n    | y | n                 | n          | -            | -                 | - | - |
| Late Collisions              | F             | Tx               | n                  | (y               | y                  | y                  | y)               | n                  | (y         | y           | y           | y            | y             | y         | y            | y)             | -        | -  | -    | - | -                 | y          | -            | -                 | - | - |
| Deferred Tx Frames           | F             | Tx               | n                  | (y               | y                  | y                  | y)               | (y                 | y          | y           | y           | y            | y             | y         | y            | -              | -        | n  | n    | n | n                 | n          | -            | y                 | n |   |

**Table 14-20. Tx Statistics Summary (continued)**

|                      |   |       |    |   |   |   |    |    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |   |   |   |
|----------------------|---|-------|----|---|---|---|----|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|---|---|---|
| Carrier Sense Errors | F | Tx    | (y | y | y | y | y) | (y | y | y | y | y | y | y | y | - | - | -  | -  | -  | -  | y  | -  | - | - |   |
| 64octet Frames       | F | Rx+Tx | (y | y | y | y | y) | y  | n | n | n | n | n | n | n | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| 65-127octet Frames   | F | Rx+Tx | (y | y | y | y | y) | n  | y | n | n | n | n | n | n | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| 128-255octet Frames  | F | Rx+Tx | (y | y | y | y | y) | n  | n | y | n | n | n | n | n | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| 256-511octet Frames  | F | Rx+Tx | (y | y | y | y | y) | n  | n | n | y | n | n | n | n | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| 512-1023octet Frames | F | Rx+Tx | (y | y | y | y | y) | n  | n | n | n | y | n | n | n | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| 1024-UPoctet Frames  | F | Rx+Tx | (y | y | y | y | y) | n  | n | n | n | n | y | y | y | - | - | -  | -  | n  | n  | n  | n  | - | - | - |
| Tx Octets            | O | Tx    | (y | y | y | y | y) | (y | y | y | y | y | y | y | y | - | - | -  | -  | n  | n  | n  | n  | - | - | n |
| Net Octets           | O | Rx+Tx | (y | y | y | y | y) | (y | y | y | y | y | y | y | y | - | - | \$ | \$ | \$ | \$ | \$ | \$ | - | - | - |

**Notes for the Tx Statistics Summary:**

1. "AND" is assumed horizontally across the table between all conditions which form the statistic (marked y or n) except where (y|y), meaning "OR" is indicated. Parentheses are significant.
2. "-" indicates conditions which are ignored in the formations of the statistic.
3. Statistics marked "Rx+Tx" are formed by summing the Rx and Tx statistics, each of which is formed independently.
4. Pause (MAC) frames are issued in the MAC as perfect (no CRC error) 64 byte frames in full duplex only, so they cannot collide.
5. "%" If a CPU sourced MAC control frame has a multicast or broadcast destination address then the appropriate statistics will be updated.
6. "+" indicates collisions which are "summed" (i.e. every collision is counted in the Collisions statistic). Jam sequences used for halfduplex flow control are also counted.
7. "\$" Every byte written on the wire during each retry attempt is also counted in addition to frames which experience no collisions or carrier loss.
8. The flow collision type is for half-duplex collisions forced by the MAC to achieve flow control. Some of the '-'s in this column might in reality be 'n's. To prevent double-counting, Net Octets are unaffected by the jam sequence – the 'received' bytes, however, are counted. (See [Table 14-19](#).)
9. When the transmit Tx FIFO is drained due to the MAC being disabled or link being lost, then the frames being purged will not appear in the Tx statistics.

### 14.3.3 Ethernet Mac Sliver (CPGMAC\_SL)

The CPGMAC\_SL peripheral shall be compliant to the IEEE Std 802.3 Specification. Half duplex mode is supported in 10/100 Mbps mode, but not in 1000 Mbps (gigabit) mode.

Features:

- Synchronous 10/100/1000 Mbit operation.
- G/MII Interface.
- Hardware Error handling including CRC.
- Full Duplex Gigabit operation (half duplex gigabit is not supported).
- EtherStats and 802.3Stats RMON statistics gathering support for external statistics collection module.
- Transmit CRC generation selectable on a per channel basis.
- Emulation Support.
- VLAN Aware Mode Support.
- Hardware flow control.
- Programmable Inter Packet Gap (IPG)

#### 14.3.3.1 GMII/MII Media Independent Interface

The following sections cover operation of the Media Independent Interface in 10/100/1000 Mbps modes. An IEEE 802.3 compliant Ethernet MAC controls the interface.

##### 14.3.3.1.1 Data Reception

###### 14.3.3.1.1.1 Receive Control

Data received from the PHY is interpreted and output. Interpretation involves detection and removal of the preamble and start of frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation.

###### 14.3.3.1.1.2 Receive Inter-Frame Interval

The 802.3 required inter-packet gap (IPG) is 24 GMII clocks (96 bit times) for 10/100 Mbit modes, and 12 GMII clocks (96 bit times) for 1000 Mbit mode. However, the MAC can tolerate a reduced IPG (2 GMII clocks in 10/100 mode and 5 GMII clocks in 1000 mode) with a correct preamble and start frame delimiter.

This interval between frames must comprise (in the following order):

- An Inter-Packet Gap (IPG).
- A seven octet preamble (all octets 0x55).
- A one octet start frame delimiter (0x5D).

#### **14.3.3.1.2 Data Transmission**

The Gigabit Ethernet Mac Sliver (GMII) passes data to the PHY when enabled. Data is synchronized to the transmit clock rate. The smallest frame that can be sent is two bytes of data with four bytes of CRC (6 byte frame).

##### **14.3.3.1.2.1 Transmit Control**

A jam sequence is output if a collision is detected on a transmit packet. If the collision was late (after the first 64 bytes have been transmitted) the collision is ignored. If the collision is not late, the controller will back off before retrying the frame transmission. When operating in full duplex mode the carrier sense (CRS) and collision sensing modes are disabled.

##### **14.3.3.1.2.2 CRC Insertion**

The MAC generates and appends a 32-bit Ethernet CRC onto the transmitted data if the transmit packet header **pass\_crc** bit is zero. For the CPMAC\_SL generated CRC case, a CRC at the end of the input packet data is not allowed.

If the header word **pass\_crc** bit is set, then the last four bytes of the TX data are transmitted as the frame CRC. The four CRC data bytes should be the last four bytes of the frame and should be included in the packet byte count value. The MAC performs no error checking on the outgoing CRC when the **pass\_crc** bit is set.

##### **14.3.3.1.2.3 MTXER**

The GMII\_MTXER signal is not used. If an underflow condition occurs on a transmitted frame, the frame CRC will be inverted to indicate the error to the network. Underflow is a hardware error.

##### **14.3.3.1.2.4 Adaptive Performance Optimization (APO)**

The Ethernet MAC port incorporates Adaptive Performance Optimization (APO) logic that may be enabled by setting the **tx\_pace** bit in the **MacControl** register. Transmission pacing to enhance performance is enabled when set. Adaptive performance pacing introduces delays into the normal transmission of frames, delaying transmission attempts between stations, reducing the probability of collisions occurring during heavy traffic (as indicated by frame deferrals and collisions) thereby increasing the chance of successful transmission.

When a frame is deferred, suffers a single collision, multiple collisions or excessive collisions, the pacing counter is loaded with an initial value of 31. When a frame is transmitted successfully (without experiencing a deferral, single collision, multiple collision or excessive collision) the pacing counter is decremented by one, down to zero.

With pacing enabled, a new frame is permitted to immediately (after one IPG) attempt transmission only if the pacing counter is zero. If the pacing counter is non zero, the frame is delayed by the pacing delay, a delay of approximately four inter-packet gap delays. APO only affects the IPG preceding the first attempt at transmitting a frame. It does not affect the back-off algorithm for retransmitted frames.

##### **14.3.3.1.2.5 Inter-Packet-Gap Enforcement**

The measurement reference for the IPG of 96 bit times is changed depending on frame traffic conditions. If a frame is successfully transmitted without collision, and **MCRS** is de-asserted within approximately 48 bit times of **MTXEN** being de-asserted, then 96 bit times is measured from **MTXEN**. If the frame suffered a collision, or if **MCRS** is not de-asserted until more than approximately 48 bit times after **MTXEN** is de-asserted, then 96 bit times (approximately, but not less) is measured from **MCRS**.

The transmit IPG can be shortened by eight bit times when enabled and triggered. The **tx\_short\_gap\_en** bit in the **MacControl** register enables the **TX\_SHORT\_GAP** input to determine whether the transmit IPG is shorted by eight bit times.

#### **14.3.3.1.2.6 Back Off**

The Gigabit Ethernet Mac Sliver (GMII) implements the 802.3 binary exponential back-off algorithm.

#### **14.3.3.1.2.7 Programmable Transmit Inter-Packet Gap**

The transmit inter-packet gap (IPG) is programmable through the **Tx\_Gap** register. The default value is decimal 12. The transmit IPG may be increased to the maximum value of 0x1ff. Increasing the IPG is not compatible with transmit pacing. The short gap feature will override the increased gap value, so the short gap feature may not be compatible with an increased IPG.

#### **14.3.3.1.2.8 Speed, Duplex, and Pause Frame Support Negotiation**

The CPMAC\_SL can operate in half duplex or full duplex in 10/100 Mbit modes, and can operate in full duplex only in 1000 Mbit mode. Pause frame support is included in 10/100/1000 Mbit modes as configured by the host.

### **14.3.3.2 Frame Classification**

Received frames are proper (good) frames if they are between 64 and **rx maxlen** in length (inclusive) and contain no errors (code/align/CRC).

Received frames are long frames if their frame count exceeds the value in the **rx maxlen** register. The **rx maxlen** register reset (default) value is 1518 (dec). Long received frames are either oversized or jabber frames. Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment errors are jabber frames.

Received frames are short frames if their frame count is less than 64 bytes. Short frames that contain no errors are undersized frames. Short frames with CRC, code, or alignment errors are fragment frames.

A received long packet will always contain **rx maxlen** number of bytes transferred to memory (if **rx\_cef\_en** = 1). An example with **rx maxlen** = 1518 is below:

- If the frame length is 1518, then the packet is not a long packet and there will be 1518 bytes transferred to memory.
- If the frame length is 1519, there will be 1518 bytes transferred to memory. The last three bytes will be the first three CRC bytes.
- If the frame length is 1520, there will be 1518 bytes transferred to memory. The last two bytes will be the first two CRC bytes.
- If the frame length is 1521, there will be 1518 bytes transferred to memory. The last byte will be the first CRC byte.

If the frame length is 1522, there will be 1518 bytes transferred to memory. The last byte will be the last data byte.

### **14.3.4 Command IDLE**

The **cmd\_idle** bit in the **MAC\_CONTROL** register allows CPGMAC\_SL operation to be suspended. When the idle state is commanded, the CPGMAC\_SL will stop processing receive and transmit frames at the next frame boundary. Any frame currently in reception or transmission will be completed normally without suspension. Received frames that are detected after the suspend state is entered are ignored. Commanded idle is similar in operation to emulation control and clock stop.

### 14.3.5 RMII Interface

The CPRMII peripheral shall be compliant to the RMII specification document.

Features:

- Source Synchronous 10/100 Mbit operation.
- Full and Half Duplex support.

#### 14.3.5.1 RMII Receive (RX)

The CPRMII receive (RX) interface converts the input data from the external RMII PHY (or switch) into the required MII (CPGMAC) signals. The carrier sense and collision signals are determined from the RMII input data stream and transmit inputs as defined in the RMII specification.

An asserted RMII\_RXER on any di-bit in the received packet will cause an MII\_RXER assertion to the CPGMAC during the packet. In 10Mbps mode, the error is not required to be duplicated on 10 successive clocks. Any di-bit which has an asserted RMII\_RXER during any of the 10 replications of the data will cause the error to be propagated.

Any received packet that ends with an improper nibble boundary aligned RMII\_CRS\_DV toggle will issue an MII\_RXER during the packet to the CPGMAC. Also, a change in speed or duplex mode during packet operations will cause packet corruption.

The CPRMII can accept receive packets with shortened preambles, but 0x55 followed by a 0x5d is the shortest preamble that will be recognized (1 preamble byte with the start of frame byte). At least one byte of preamble with the start of frame indicator is required to begin a packet. An asserted RMII\_CRS\_DV without at least a single correct preamble byte followed by the start of frame indicator will be ignored.

#### 14.3.5.2 RMII Transmit (TX)

The CPRMII transmit (TX) interface converts the 3PSW MII input data into the RMII transmit format. The data is then output to the external RMII PHY.

The 3PSW does not source the transmit error (MII TXERR) signal. Any transmit frame from the CPGMAC with an error (ie. underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRMII module. Zeroes are output on RMII\_TXD[1:0] for each clock that RMII\_TXEN is deasserted.

### 14.3.6 RGMII Interface

The CPRGMII peripheral shall be compliant to the RGMII specification document.

Features:

- Supports 1000/100/10 Mbps Speed.
- MII mode is not supported.

#### 14.3.6.1 RGMII Receive (RX)

The CPRGMII receive (RX) interface converts the source synchronous DDR input data from the external RGMII PHY into the required G/MII (CPGMAC) signals.

#### 14.3.6.2 In-Band Mode of Operation

The CPRGMII is operating in the in-band mode of operation when the **RGMII\_RX\_INBAND** input is asserted. RGMII\_RX\_INPUT is asserted by configuring the ext\_en bit to 1 of sl\_mac\_control register. The link status, duplexity, and speed are determined from the RGMII input data stream as defined in the RGMII specification. The link speed is indicated as shown in the following table:

|                  |               |
|------------------|---------------|
| RGMII_SPEED(1:0) | Link Speed    |
| 00               | 10 Mbs mode   |
| 01               | 100 Mbs mode  |
| 10               | 1000 Mbs mode |
| 11               | reserved      |

#### 14.3.6.3 Forced Mode of Operation

The CPRGMII is operating in the forced mode of operation when the **RGMII\_RX\_INBAND** input is deasserted. In the forced mode of operation, the in-band data is ignored if present. The link status is forced high, and the duplexity and speed are determined from the **RGMII\_FULLDUPLEX\_IN** and **RGMII\_GIG\_IN** inputs. If the **RGMII\_GIG\_IN** input is asserted, then operation is gigabit mode (**RGMII\_SPEED (1:0) = 10**). If the **RGMII\_GIG\_IN** input is deasserted, the operation is 100mbps mode (**RGMII\_SPEED (1:0) = 01**).

#### 14.3.6.4 RGMII Transmit (TX)

The CPRGMII transmit (TX) interface converts the CPGMAC G/MII input data into the DDR RGMII format. The DDR data is then output to the external PHY.

The CPGMAC does not source the transmit error (MTXERR) signal. Any transmit frame from the CPGMAC with an error (ie. underrun) will be indicated as an error by an error CRC. Transmit error is assumed to be deasserted at all times and is not an input into the CPRGMII module.

In 10/100 mode, the MTXD (7:0) data bus uses only the lower nibble. The CPRGMII will output the lower nibble twice in 10/100 mode to avoid unnecessary signal switching.

Packets will be precluded from transmission through the CPRGMII module for 4096 transmit clocks after the rising edge of **RGMII\_LINK**. Packet transmission will begin on the first **GMII\_MTXEN** rising edge after the 4096 transmit clock count has expired.

The RGMII0/1\_ID\_MODE bit value in the GMII\_SEL register should only be set to 1 for 'no internal delay'. The device does not support internal delay mode for RGMII.

### 14.3.7 Common Platform Time Sync (CPTS)

The CPTS module is used to facilitate host control of time sync operations. It enables compliance with the IEEE 1588-2008(v2) standard for a precision clock synchronization protocol.

#### 14.3.7.1 Architecture

**Figure 14-10. CPTS Block Diagram**

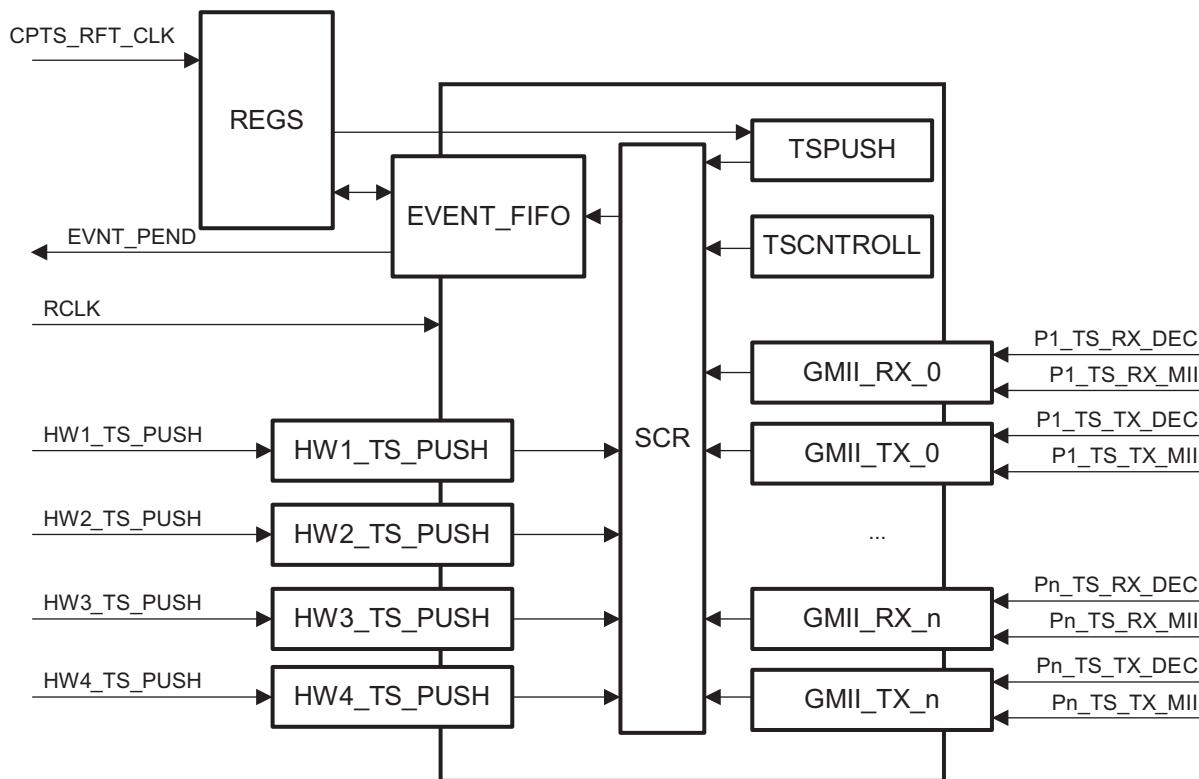


Figure 14-10 shows the architecture of the CPTS module inside the 3PSW Ethernet Subsystem. Time stamp values for every packet transmitted or received on either port of the 3PSW are recorded. At the same time, each packet is decoded to determine if it is a valid time sync event. If so, an event is loaded into the Event FIFO for processing containing the recorded time stamp value when the packet was transmitted or received.

In addition, both hardware (HW<sub>x</sub>\_TS\_PUSH) and software (TS\_PUSH) can be used to read the current time stamp value through the Event FIFO.

The reference clock used for the time stamp (RCLK) is sourced from one of the two sources, as shown in Figure 14-10. The source can be selected by configuring the CM\_CPTS\_RFT\_CLKSEL register in the Control Module. For more details, see Chapter 9, Control Module.

#### 14.3.7.2 Time Sync Overview

The CPTS module is used to facilitate host control of time sync operations. The CPTS collects time sync events and then presents them to the host for processing. There are five types of time sync events (ethernet receive event, ethernet transmit event, time stamp push event, time stamp rollover event, and time stamp half-rollover event). Each ethernet port can cause transmit and receive events. The time stamp push is initiated by software.

##### 14.3.7.2.1 Time Sync Initialization

The CPTS module should be configured as shown:

- Complete the reset sequence (VBUSP\_RST\_N) to reset the module.

- Write the rftclk\_sel[4:0] value in the RFTCLK\_Sel register with the desired reference clock multiplexor value. This value is allowed to be written only when the cpts\_en bit is cleared to zero.
- Write a one to the cpts\_en bit in the TS\_Control register. The RCLK domain is in reset while this bit is low.
- Enable the interrupt by writing a one to the ts\_pend\_en bit in the TS\_Int\_Enable register (if using interrupts and not polling).

#### 14.3.7.2.2 Time Stamp Value

The time stamp value is a 32-bit value that increments on each RCLK rising edge when CPTS\_EN is set to one. When CPTS\_EN is cleared to zero the time stamp value is reset to zero. If more than 32-bits of time stamp are required by the application, the host software must maintain the necessary number of upper bits. The upper time stamp value should be incremented by the host when the rollover event is detected.

For test purposes, the time stamp can be written via the time stamp load function (CPTS\_LOAD\_VAL and CPTS\_LOAD\_EN registers).

#### 14.3.7.2.3 Event FIFO

All time sync events are pushed onto the Event FIFO. There are 16 locations in the event FIFO with no overrun indication supported. Software must service the event FIFO in a timely manner to prevent FIFO overrun.

#### 14.3.7.2.4 Time Sync Events

Time Sync events are 64-bit values that are pushed onto the event FIFO and read in two 32-bit reads. CPTS\_EVENT\_LOW and CPTS\_EVENT\_HIGH are defined in [Section 14.5.3.10](#) and [Section 14.5.3.11](#), respectively.

There are six types of sync events

- Time stamp push event
- Hardware time stamp push event
- Time stamp counter rollover event
- Time stamp counter half-rollover event
- Ethernet receive event
- Ethernet transmit event

##### 14.3.7.2.4.1 Time Stamp Push Event

Software can obtain the current time stamp value (at the time of the write) by initiating a time stamp push event. The push event is initiated by setting the TS\_PUSH bit of the CPTS\_TS\_PUSH register. The time stamp value is returned in the event, along with a time stamp push event code. Software should not push a second time stamp event on to the FIFO until the first time stamp value has been read from the event FIFO.

##### 14.3.7.2.4.2 Time Stamp Counter Rollover Event

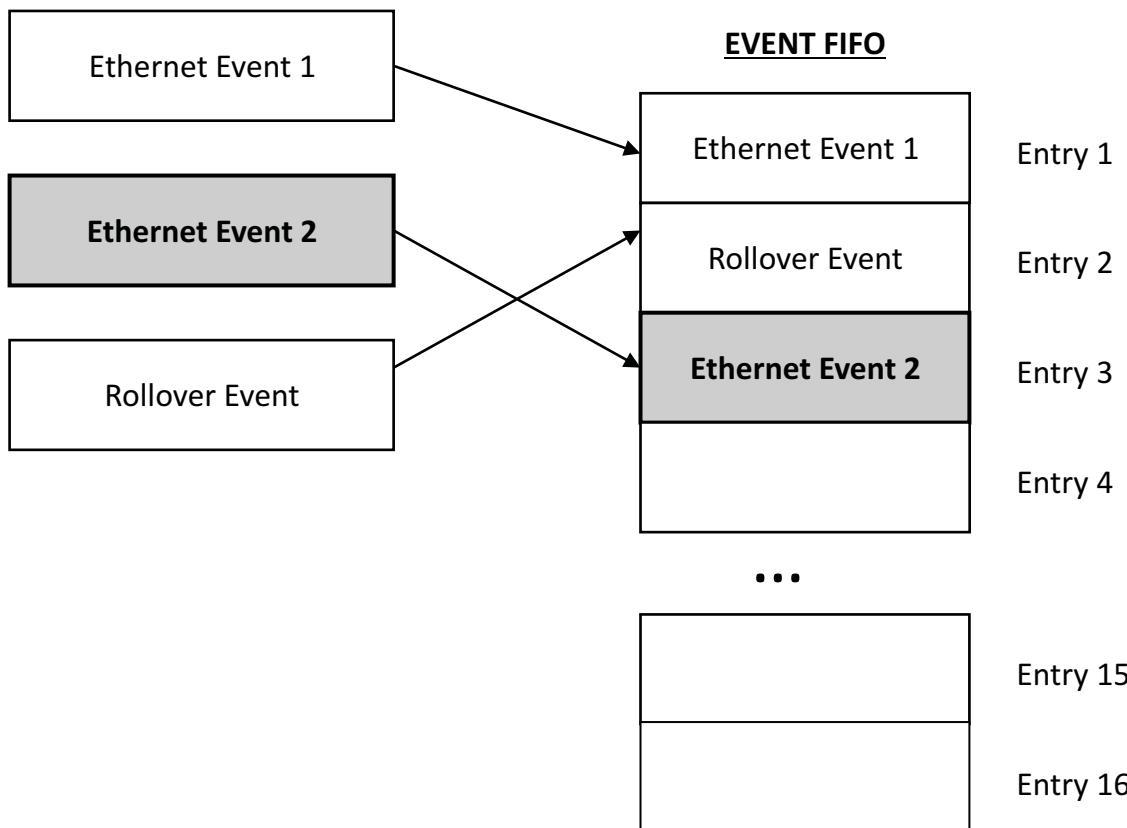
The CPTS module contains a 32-bit time stamp value. The counter upper bits are maintained by host software. The rollover event indicates to software that the time stamp counter has rolled over from 0xFFFF\_FFFF to 0x0000\_0000, and the software maintained upper count value should be incremented.

#### 14.3.7.2.4.3 Time Stamp Counter Half-Rollover Event

The CPTS includes a time stamp counter half-rollover event. The half-rollover event indicates to software that the time stamp value has incremented from 0x7FFF\_FFFF to 0x8000\_0000. The half-rollover event is included to enable software to correct a misaligned event condition. The half-rollover event is included to enable software to determine the correct time for each event that contains a valid time stamp value – such as an Ethernet event. If an Ethernet event occurs around a counter rollover (full rollover), the rollover event could possibly be loaded into the event FIFO before the Ethernet event, even though the Ethernet event time was actually taken before the rollover. Figure 3 below shows a misalignment condition.

This misaligned event condition arises because an ethernet event time stamp occurs at the beginning of a packet and time passes before the packet is determined to be a valid synchronization packet. The misaligned event condition occurs if the rollover occurs in the middle, after the packet time stamp has been taken, but before the packet has been determined to be a valid time sync packet.

**Figure 14-11. Event FIFO Misalignment Condition**



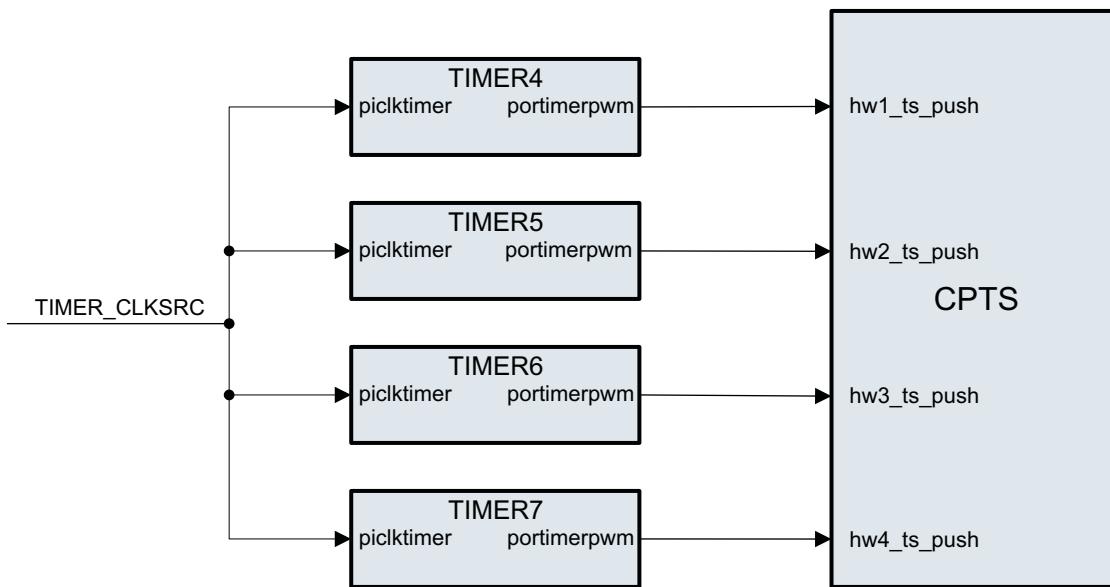
Host software must detect and correct for misaligned event conditions. For every event after a rollover and before a half-rollover, software must examine the time stamp most significant bit. If bit 31 of the time stamp value is low (0x0000\_0000 through 0x7FFF\_FFFF), then the event time stamp was taken after the rollover and no correction is required.

If the value is high (0x8000\_0000 through 0xFFFF\_FFFF), the time stamp value was taken before the rollover and a misalignment is detected. The misaligned case indicates to software that it must subtract one from the upper count value stored in software to calculate the correct time for the misaligned event. The misaligned event occurs only on the rollover boundary and not on the half-rollover boundary. Software only needs to check for misalignment from a rollover event to a half-rollover event.

#### 14.3.7.2.4.4 Hardware Time Stamp Push Event

There are four hardware time stamp inputs (HW1/4\_TS\_PUSH) that can cause hardware time stamp push events to be loaded into the Event FIFO. Each hardware time stamp input is internally connected to the PORTIMERPW output of each timer as shown in Figure 4.

Figure 14-12. HW1/4\_TSP\_PUSH Connection



The event is loaded into the event FIFO on the rising edge of the timer, and the PORT\_NUMBER field in the EVENT\_HIGH register indicates the hardware time stamp input that caused the event.

Each hardware time stamp input must be asserted for at least 10 periods of the selected RCLK clock. Each input can be enabled or disabled by setting the respective bits in the **CONTROL** register.

Hardware time stamps are intended to be an extremely low frequency signals, such that the event FIFO does not overrun. Software must keep up with the event FIFO and ensure that there is no overrun, or events will be lost.

#### 14.3.7.2.4.5 Ethernet Port Events

##### 14.3.7.2.4.5.1 Ethernet Port Receive Event

Each ethernet port can generate a receive ethernet event. Receive ethernet events are generated for valid received time sync packets. There are two CPTS interfaces for each ethernet receive port. The first is the Px\_TS\_RX\_MII interface and the second is the Px\_TS\_RX\_DEC interface. Information from these interfaces is used to generates an ethernet receive event for each ethernet time sync packet received on the associated port.

The Px\_TS\_RX\_MII interface issues a record signal (pX\_ts\_rx\_mii\_rec) along with a handle (pX\_ts\_rx\_mii\_hdl) for each packet (every packet) that is received on the associated ethernet port. The record signal is a single clock pulse indicating that a receive packet has been detected at the associated port MII interface. The handle value is incremented with each packet and rolls over to zero after 15.

There are 16 possible handle values so there can be a maximum of 16 packets “in flight” from the TS\_RX\_MII to the TS\_RX\_DEC block at any given time. A handle value is reused (not incremented) for any received packet that is shorter than about 31 octets (including preamble). Handle reuse on short packets prevents any possible overrun condition (more than 16 “in flight” packets) if multiple fragments are consecutively received.

Valid receive ethernet time sync events are signaled to the CPTS via the Px\_TS\_RX\_DEC interface. When the pX\_ts\_rx\_dec\_evt is asserted, a valid event is detected and will be loaded into the event FIFO. Only valid receive time sync packets are indicated on the Px\_TS\_RX\_DEC interface. The pX\_ts\_rx\_dec\_hdl, pX\_ts\_rx\_dec\_msg\_type, and pX\_ts\_rx\_dec\_seq\_id signals are registered on an asserted pX\_ts\_rx\_dec\_evt. When a Tx\_TS\_RX\_DEC event is asserted, the handle value is used to retrieve the time stamp that was loaded with the same handle value from the Px\_TS\_RX\_MII interface.

#### 14.3.7.2.4.5.2 Ethernet Port Transmit Event

Each ethernet port can generate a transmit ethernet event. Transmit ethernet events are generated for valid transmitted time sync packets. There are two CPTS interfaces for each ethernet transmit port. The first is the Px\_TS\_TX\_DEC interface and the second is the Px\_TS\_TX\_MII interface. Information from these interfaces is used to generates an ethernet transmit event for each ethernet time sync packet transmitted on the associated port.

Valid ethernet transmit time sync events are signaled to the CPTS via the Px\_TS\_TX\_DEC interface. When the pX\_ts\_tx\_dec\_evnt signal is asserted, a valid time sync event has been detected and will be loaded into the event FIFO. Only valid transmit time sync packets are indicated on the Px\_TS\_RX\_DEC interface. The pX\_ts\_tx\_dec\_hdl, pX\_ts\_tx\_dec\_msg\_type, pX\_ts\_tx\_dec\_seq\_id signals are registered on an asserted pX\_ts\_tx\_dec\_evnt.

The time stamp for the event will be generated and signaled from the Px\_TS\_TX\_MII interface when the packet is actually transmitted. The event will be loaded into the event FIFO when the time stamp is recorded as controlled by the Px\_TS\_TX\_MII interface. The handle value is incremented with each time sync event packet and rolls over to zero after 7. There are 8 possible handle values so there can be a maximum of 8 time sync event packets “in flight” from the TS\_TX\_DEC to the TS\_TX\_MII block at any given time. The handle value increments only on time sync event packets.

The Px\_TS\_TX\_MII interface issues a single clock record signal (pX\_ts\_tx\_mii\_rec) at the beginning of each transmitted packet. If the packet is a time sync event packet then a single clock event signal (pX\_ts\_tx\_mii\_evnt) along with a handle (pX\_ts\_rx\_mii\_hdl) will be issued before the next record signal for the next transmitted packet. The event signal will not be issued for packets that were not indicated as valid time sync event packets on the Px\_TS\_TX\_DEC interface. If consecutive record indications occur without an interleaving event indication, then the packet associated with the first record was not a time sync event packet. The record signal is a single clock pulse indicating that a transmit packet egress has been detected at the associated port MII interface.

**Table 14-21. Values of messageType field**

| Message Type          | Value (hex) |
|-----------------------|-------------|
| Sync                  | 0           |
| Delay_Req             | 1           |
| Pdelay_Req            | 2           |
| Pdelay_Resp           | 3           |
| Reserved              | 4-7         |
| Follow_Up             | 8           |
| Delay_Resp            | 9           |
| Pdelay_Resp_Follow_Up | A           |
| Announce              | B           |
| Signaling             | C           |
| Management            | D           |
| Reserved              | E-F         |

#### 14.3.7.3 Interrupt Handling

When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred. The following steps should be taken to process time sync events using interrupts:

- Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPTS\_INT\_ENABLE register.
- Upon interrupt, read the CPTS\_EVENT\_LOW and CPTS\_EVENT\_HIGH register values.
- Set the EVENT\_POP field (bit zero) of the CPTS\_EVENT\_POP register to pop the previously read value off of the event FIFO.
- Process the interrupt as required by the application software

Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way:

1. Enable the TS\_PEND interrupt by setting the TS\_PEND\_EN bit of the CPTS\_INT\_ENABLE register.
2. Upon interrupt enter the CPTS service routine.
3. Read the CPTS\_EVENT\_LOW and CPTS\_EVENT\_HIGH register values.
4. Set the EVENT\_POP bit of the CPTS\_EVENT\_POP register to pop the previously read value off of the event FIFO.
5. Wait for an amount of time greater than eight CPTS\_RFT\_CLK periods
6. Read the ts\_pend\_raw bit in the CPTS\_INTSTAT\_RAW register to determine if another valid event is in the event FIFO. If it is asserted then goto step 3. Otherwise goto step 7.
7. Process the interrupt(s) as required by the application software

Software also has the option of disabling the interrupt and polling the ts\_pend\_raw bit of the CPTS\_INTSTAT\_RAW register to determine if a valid event is on the event FIFO.

### 14.3.8 MDIO

The MII Management I/F module implements the 802.3 serial management interface to interrogate and control two Ethernet PHYs simultaneously using a shared two-wire bus. Two user access registers to control and monitor up to two PHYs simultaneously.

#### 14.3.8.1 MII Management Interface Frame Formats

The following tables show the read and write format of the 32-bit MII Management interface frames, respectively.

**Table 14-22. MDIO Read Frame Format**

| Preamble       | Start Delimiter | Operation Code | PHY Address | Register Address | Turnaround | Data                    |
|----------------|-----------------|----------------|-------------|------------------|------------|-------------------------|
| 0xFFFFF<br>FFF | 01              | 10             | AAAAA       | RRRRR            | Z0         | DDDD.DDDD.DDD<br>D.DDDD |

**Table 14-23. MDIO Write Frame Format**

| Preamble       | Start Delimiter | Operation Code | PHY Address | Register Address | Turnaround | Data                    |
|----------------|-----------------|----------------|-------------|------------------|------------|-------------------------|
| 0xFFFFF<br>FFF | 01              | 00             | AAAAA       | RRRRR            | 10         | DDDD.DDDD.DDD<br>D.DDDD |

The default or idle state of the two wire serial interface is a logic one. All tri-state drivers should be disabled and the PHY's pull-up resistor will pull the **MDIO** line to a logic one. Prior to initiating any other transaction, the station management entity shall send a preamble sequence of 32 contiguous logic one bits on the **MDIO** line with 32 corresponding cycles on **MDCLK** to provide the PHY with a pattern that it can use to establish synchronization. A PHY shall observe a sequence of 32 contiguous logic one bits on **MDIO** with 32 corresponding **MDCLK** cycles before it responds to any other transaction.

#### Preamble

The start of a frame is indicated by a preamble, which consists of a sequence of 32 contiguous bits all of which are a "1". This sequence provides the PHY a pattern to use to establish synchronization.

#### Start Delimiter

The preamble is followed by the start delimiter which is indicated by a "01" pattern. The pattern assures transitions from the default logic one state to zero and back to one.

#### Operation Code

The operation code for a read is "10", while the operation code for a write is a "00".

#### PHY Address

The PHY address is 5 bits allowing 32 unique values. The first bit transmitted is the MSB of the PHY address.

### Register Address

The Register address is 5 bits allowing 32 registers to be addressed within each PHY. Refer to the 10/100 PHY address map for addresses of individual registers.

### Turnaround

An idle bit time during which no device actively drives the MDIO signal shall be inserted between the register address field and the data field of a read frame in order to avoid contention. During a read frame, the PHY shall drive a zero bit onto MDIO for the first bit time following the idle bit and preceding the Data field. During a write frame, this field shall consist of a one bit followed by a zero bit.

### Data

The Data field is 16 bits. The first bit transmitted and received is the MSB of the data word.

#### 14.3.8.2 Functional Description

The MII Management I/F will remain idle until enabled by setting the enable bit in the MDIOControl register. The MII Management I/F will then continuously poll the link status from within the Generic Status Register of all possible 32 PHY addresses in turn recording the results in the MDIOLink register.

The linksel bit in the MDIOUserPhySel register determines the status input that is used. A change in the link status of the two PHYs being monitored will set the appropriate bit in the MDIOLinkIntRaw register and the MDIOLinkIntMasked register, if enabled by the linkint\_enable bit in the MDIOUserPhySel register.

The MDIOAlive register is updated by the MII Management I/F module if the PHY acknowledged the read of the generic status register. In addition, any PHY register read transactions initiated by the host also cause the MDIOAlive register to be updated.

At any time, the host can define a transaction for the MII Management interface module to undertake using the data, phyadr, regadr, and write fields in a MDIOUserAccess register. When the host sets the go bit in this register, the MII Management interface module will begin the transaction without any further intervention from the host. Upon completion, the MII Management interface will clear the go bit and set the userinraw bit in the MDIOUserIntRaw register corresponding to the MDIOUserAccess register being used.

The corresponding bit in the MDIOUserIntMasked register may also be set depending on the mask setting in the MDIOUserIntMaskSet and MDIOUserIntMaskClr registers. A round-robin arbitration scheme is used to schedule transactions which may queued by the host in different MDIOUserAccess registers. The host should check the status of the go bit in the MDIOUserAccess register before initiating a new transaction to ensure that the previous transaction has completed. The host can use the ack bit in the MDIOUserAccess register to determine the status of a read transaction.

It is necessary for software to use the MII Management interface module to setup the auto-negotiation parameters of each PHY attached to a MAC port, retrieve the negotiation results, and setup the MACControl register in the corresponding MAC.

## 14.4 Software Operation

### 14.4.1 Transmit Operation

After reset the host must write zeroes to all Tx DMA State head descriptor pointers. The Tx port may then be enabled. To initiate packet transmission the host constructs transmit queues in memory (one or more packets for transmission) and then writes the appropriate Tx DMA state head descriptor pointers. For each buffer added to a transmit queue, the host must initialize the Tx buffer descriptor values as follows:

1. Write the Next Descriptor Pointer with the 32-bit aligned address of the next descriptor in the queue (zero if last descriptor).
2. Write the Buffer Pointer with the byte aligned address of the buffer data.
3. Write the Buffer Length with the number of bytes in the buffer.
4. Write the Buffer Offset with the number of bytes in the offset to the data (nonzero with SOP only).
5. Set the SOP, EOP, and Ownership bits as appropriate.
6. Clear the End Of Queue bit.

The port begins Tx packet transmission on a given channel when the host writes the channel's Tx queue head descriptor pointer with the address of the first buffer descriptor in the queue (nonzero value). Each channel may have one or more queues, so each channel may have one or more head descriptor pointers. The first buffer descriptor for each Tx packet must have the Start of Packet (SOP) bit and the Ownership bit set to one by the host. The last buffer descriptor for each Tx packet must have the End of Packet (EOP) bit set to one by the host.

The port will transmit packets until all queued packets have been transmitted and the queue(s) are empty. When each packet transmission is complete, the port will clear the Ownership bit in the packet's SOP buffer descriptor and issue an interrupt to the host by writing the packet's last buffer descriptor address to the queue's Tx DMA State Completion Pointer. The interrupt is generated by the write, regardless of the value written.

When the last packet in a queue has been transmitted, the port sets the End Of Queue bit in the EOP buffer descriptor, clears the Ownership bit in the SOP Descriptor, zeroes the appropriate DMA state head descriptor pointer, and then issues a Tx interrupt to the host by writing to the queue's associated Tx completion pointer (address of the last buffer descriptor processed by the port). The port issues a maskable level interrupt (which may then be routed through external interrupt control logic to the host).

On interrupt from the port, the host processes the buffer queue, detecting transmitted packets by the status of the Ownership bit in the SOP buffer descriptor. If the Ownership bit is cleared to zero, then the packet has been transmitted and the host may reclaim the buffers associated with the packet. The host continues queue processing until the end of the queue or until a SOP buffer descriptor is read that contains a set Ownership bit indicating that the packet transmission is not complete.

The host determines that all packets in the queue have been transmitted when the last packet in the queue has a cleared Ownership bit in the SOP buffer descriptor, the End of Queue bit is set in the last packet EOP buffer descriptor, and the Next Descriptor Pointer of the last packet EOP buffer descriptor is zero. The host acknowledges an interrupt by writing the address of the last buffer descriptor to the queue's associated Tx Completion Pointer in the Tx DMA State.

If the host written buffer address value is different from the buffer address written by the port, then the level interrupt remains asserted. If the host written buffer address value is equal to the port written value, then the level interrupt is deasserted. The port write to the completion pointer actually stores the value in the state register (ram). The host written value is actually not written to the register location. The host written value is compared to the register contents (which was written by the port) and if the two values are equal, the interrupt is removed, otherwise the interrupt remains asserted. The host may process multiple packets previous to acknowledging an interrupt, or the host may acknowledge interrupts for every packet.

A misqueued packet condition may occur when the host adds a packet to a queue for transmission as the port finishes transmitting the previous last packet in the queue. The misqueued packet is detected by the host when queue processing detects a cleared Ownership bit in the SOP buffer descriptor, a set End of Queue bit in the EOP buffer descriptor, and a nonzero Next Descriptor Pointer in the EOP buffer descriptor.

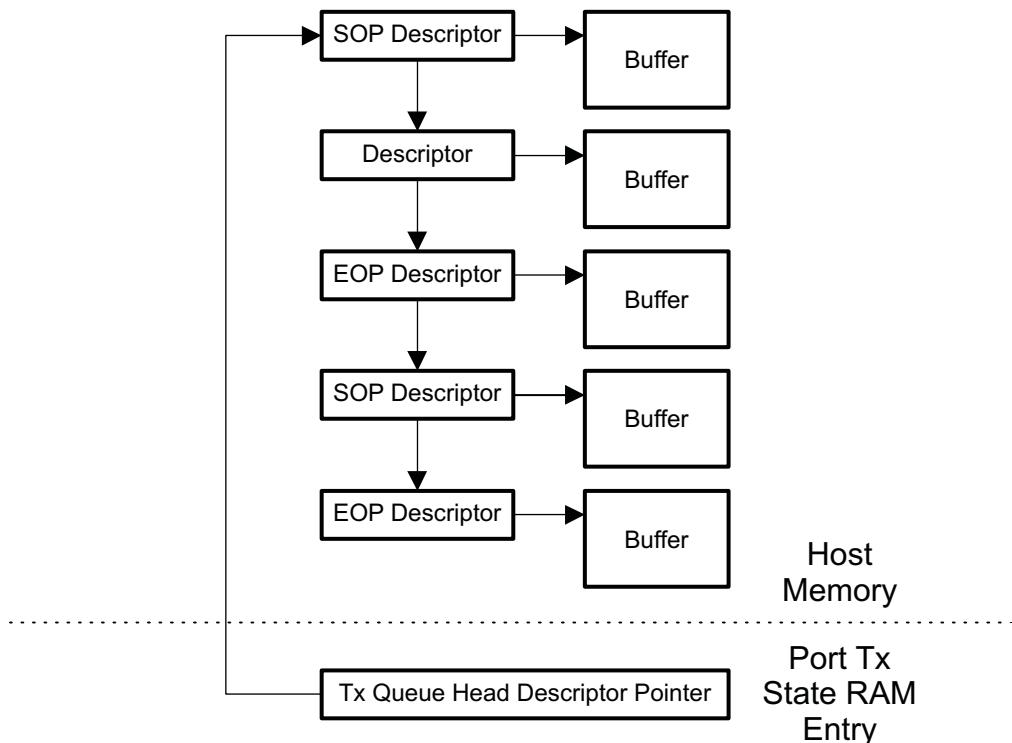
A misqueued packet means that the port read the last EOP buffer descriptor before the host added the new last packet to the queue, so the port determined queue empty just before the last packet was added. The host corrects the misqueued packet condition by initiating a new packet transfer for the misqueued packet by writing the misqueued packet's SOP buffer descriptor address to the appropriate DMA State Tx Queue head Descriptor Pointer.

The host may add packets to the tail end of an active Tx queue at any time by writing the Next Descriptor Pointer to the current last descriptor in the queue. If a Tx queue is empty (inactive), the host may initiate packet transmission at any time by writing the appropriate Tx DMA State head descriptor pointer.

The host software should always check for and reinitiate transmission for misqueued packets during queue processing on interrupt from the port. In order to preclude software underrun, the host should avoid adding buffers to an active queue for any Tx packet that is not complete and ready for transmission.

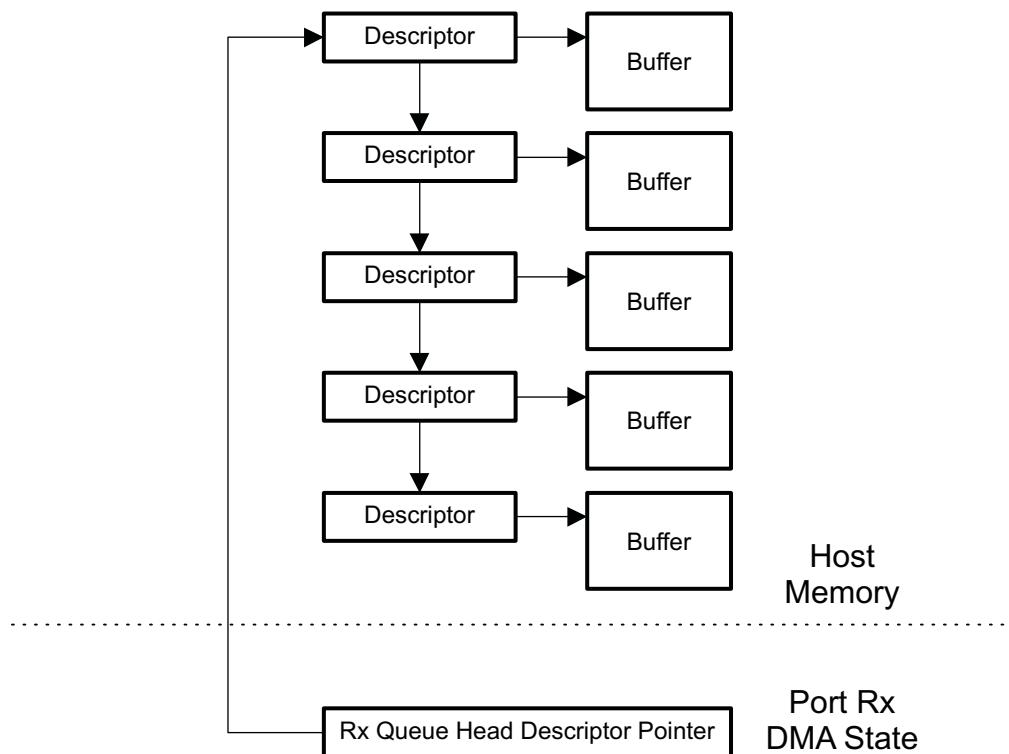
The port determines that a packet is the last packet in the queue by detecting the End of Packet bit set with a zero Next Descriptor Pointer in the packet buffer descriptor. If the End of Packet bit is set and the Next Descriptor Pointer is nonzero, then the queue still contains one or more packets to be transmitted. If the EOP bit is set with a zero Next Descriptor Pointer, then the port will set the EOQ bit in the packet's EOP buffer descriptor and then zero the appropriate head descriptor pointer previous to interrupting the port (by writing the completion pointer) when the packet transmission is complete.

**Figure 14-13. Port TX State RAM Entry**



#### 14.4.2 Receive Operation

**Figure 14-14. Port RX DMA State**



After reset the host must write zeroes to all Rx DMA State head descriptor pointers. The Rx port may then be enabled. To initiate packet reception, the host constructs receive queues in memory and then writes the appropriate Rx DMA state head descriptor pointer. For each Rx buffer descriptor added to the queue, the host must initialize the Rx buffer descriptor values as follows:

- Write the Next Descriptor Pointer with the 32-bit aligned address of the next descriptor in the queue (zero if last descriptor)
- Write the Buffer Pointer with the byte aligned address of the buffer data
- Clear the Offset field
- Write the Buffer Length with the number of bytes in the buffer
- Clear the SOP, EOP, and EOQ bits
- Set the Ownership bit

The host enables packet reception on a given channel by writing the address of the first buffer descriptor in the queue (nonzero value) to the channel's head descriptor pointer in the channel's Rx DMA state.

When packet reception begins on a given channel, the port fills each Rx buffer with data in order starting with the first buffer and proceeding through the Rx queue. If the Buffer Offset in the Rx DMA State is nonzero, then the port will begin writing data after the offset number of bytes in the SOP buffer. The port performs the following operations at the end of each packet reception:

- Overwrite the buffer length in the packet's EOP buffer descriptor with the number of bytes actually received in the packet's last buffer. The host initialized value is the buffer size. The overwritten value will be less than or equal to the host initialized value.
- Set the EOP bit in the packet's EOP buffer descriptor.
- Set the EOQ bit in the packet's EOP buffer descriptor if the current packet is the last packet in the queue.
- Overwrite the packet's SOP buffer descriptor Buffer Offset with the Rx DMA state value (the host initialized the buffer descriptor Buffer Offset value to zero). All non SOP buffer descriptors must have a zero Buffer Offset initialized by the host.

- Overwrite the packet's SOP buffer descriptor buffer length with the number of valid data bytes in the buffer. If the buffer is filled up, the buffer length will be the buffer size minus buffer offset.
- Set the SOP bit in the packet's SOP buffer descriptor.
- Write the SOP buffer descriptor Packet Length field.
- Clear the Ownership bit in the packet's SOP buffer descriptor.
- Issue an Rx host interrupt by writing the address of the packet's last buffer descriptor to the queue's Rx DMA State Completion Pointer. The interrupt is generated by the write to the Rx DMA State Completion Pointer address location, regardless of the value written.

On interrupt the host processes the Rx buffer queue detecting received packets by the status of the Ownership bit in each packet's SOP buffer descriptor. If the Ownership bit is cleared then the packet has been completely received and is available to be processed by the host.

The host may continue Rx queue processing until the end of the queue or until a buffer descriptor is read that contains a set Ownership bit indicating that the next packet's reception is not complete. The host determines that the Rx queue is empty when the last packet in the queue has a cleared Ownership bit in the SOP buffer descriptor, a set End of Queue bit in the EOP buffer descriptor, and the Next Descriptor Pointer in the EOP buffer descriptor is zero.

A misqueued buffer may occur when the host adds buffers to a queue as the port finishes the reception of the previous last packet in the queue. The misqueued buffer is detected by the host when queue processing detects a cleared Ownership bit in the SOP buffer descriptor, a set End of Queue bit in the EOP buffer descriptor, and a nonzero Next Descriptor Pointer in the EOP buffer descriptor.

A misqueued buffer means that the port read the last EOP buffer descriptor before the host added buffer descriptor(s) to the queue, so the port determined queue empty just before the host added more buffer descriptor(s). In the transmit case, the packet transmission is delayed by the time required for the host to determine the condition and reinitiate the transaction, but the packet is not actually lost. In the receive case, receive overrun condition may occur in the misqueued buffer case.

If a new packet reception is begun during the time that the port has determined the end of queue condition, then the received packet will overrun (start of packet overrun). If the misqueued buffer occurs during the middle of a packet reception then middle of packet overrun may occur. If the misqueued buffer occurs after the last packet has completed, and is corrected before the next packet reception begins, then overrun will not occur. The host acts on the misqueued buffer condition by writing the added buffer descriptor address to the appropriate Rx DMA State Head Descriptor Pointer.

#### **14.4.3 Initializing the MDIO Module**

The following steps are performed by the application software or device driver to initialize the MDIO device:

1. Configure the PREAMBLE and CLKDIV bits in the MDIO control register (MDIOCONTROL).
2. Enable the MDIO module by setting the ENABLE bit in MDIOCONTROL.
3. The MDIO PHY alive status register (MDIOALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIOLINK) can determine whether this PHY already has a link.
4. Setup the appropriate PHY addresses in the MDIO user PHY select register (MDIOUSERPHYSEL $n$ ), and set the LINKINTENB bit to enable a link change event interrupt if desirable.
- If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET) to use the MDIO user access register (MDIOUSERACCESS $n$ ). Since only one PHY is used in this device, the application software can use one MDIOUSERACCESS $n$  to trigger a completion interrupt; the other MDIOUSERACCESS $n$  is not setup.

#### **14.4.4 Writing Data to a PHY Register**

The MDIO module includes a user access register (MDIOUSERACCESS $n$ ) to directly access a specified PHY device. To write a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (MDIOUSERACCESS $n$ ) is cleared.

2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIOUSERACCESS $n$  corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIOUSERACCESS $n$  for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESS $n$  used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU.

#### **14.4.5 Reading Data from a PHY Register**

The MDIO module includes a user access register (MDIOUSERACCESS $n$ ) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Check to ensure that the GO bit in the MDIO user access register (MDIOUSERACCESS $n$ ) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in MDIOUSERACCESS $n$  corresponding to the PHY and PHY register you want to read.
3. The read data value is available in the DATA bits in MDIOUSERACCESS $n$  after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in MDIOUSERACCESS $n$ . After the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESS $n$  used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU.

#### **14.4.6 Initialization and Configuration of CPSW**

To configure the 3PSW Ethernet Subsystem for operation the host must perform the following:

- Select the Interface (GMII/RGMII/MII) Mode in the Control Module.
- Configure pads (PIN muxing) as per the Interface Selected using the appropriate pin muxing conf\_xxx registers in the Control Module.
- Enable the 3PSW Ethernet Subsystem Clocks. See [Section 14.2.2](#) and [Section 8.1](#) to enable the appropriate clocks.
- Configure the PRCM Registers CM\_PER\_CPSW\_CLKSTCTRL and CM\_PER\_CPSW\_CLKSTCTRL to enable power and clocks to the 3PSW Ethernet Subsystem. See [Section 8.1](#) for register details.
- Apply soft reset to 3PSW Subsystem, CPSW\_3G, CPGMAC\_SL1/2, and CPDMA (see the soft reset registers in the following sections).
- Initialize the HDPs (Header Description Pointer) and CPs (Completion Pointer) to NULL
- Configure the Interrupts (see [Chapter 6](#)).
- Configure the CPSW\_3G Control register.
- Configure the Statistics Port Enable register
- Configure the ALE. (See [Section 14.3.2.7](#).)
- Configure the MDIO.
- Configure the CPDMA receive DMA controller.
- Configure the CPDMA transmit DMA controller.
- Configure the CPPI Tx and Rx Descriptors.
- Configure CPGMAC\_SL1 and CPGMAC\_SL2 as per the desired mode of operations.
- Start up RX and TX DMA (write to HDP of Rx and Tx).
- Wait for the completion of the transfer (HDP cleared to zero).

## 14.5 Ethernet Subsystem Registers

### 14.5.1 CPSW\_ALE Registers

Table 14-24 lists the memory-mapped registers for the CPSW\_ALE. All register offset addresses not listed in Table 14-24 should be considered as reserved locations and the register contents should not be modified.

**Table 14-24. CPSW\_ALE REGISTERS**

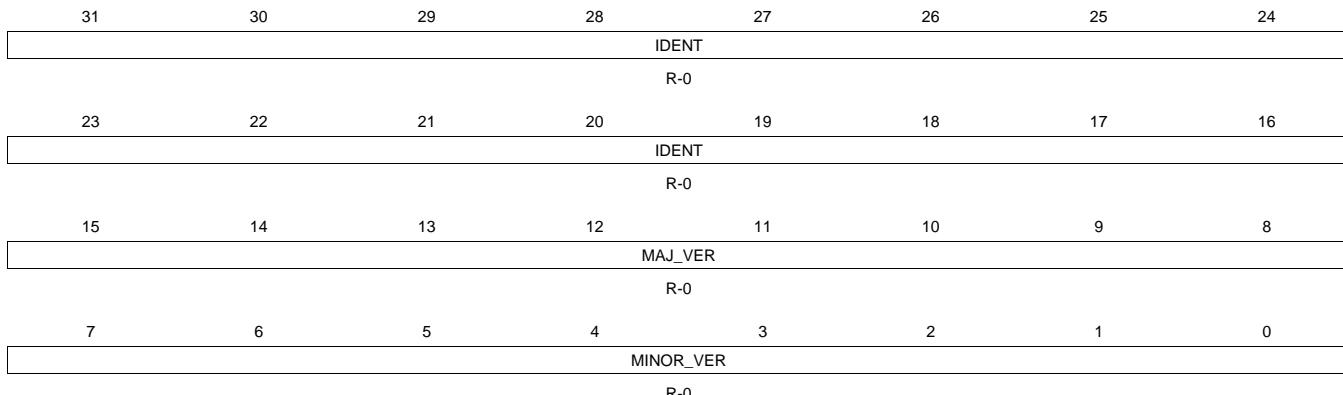
| Offset | Acronym      | Register Name                                 | Section                           |
|--------|--------------|---|-----------------------------------|
| 0h     | IDVER        | ADDRESS LOOKUP ENGINE ID/VERSION REGISTER     | <a href="#">Section 14.5.1.1</a>  |
| 8h     | CONTROL      | ADDRESS LOOKUP ENGINE CONTROL REGISTER        | <a href="#">Section 14.5.1.2</a>  |
| 10h    | PRESCALE     | ADDRESS LOOKUP ENGINE PRESCALE REGISTER       | <a href="#">Section 14.5.1.3</a>  |
| 18h    | UNKNOWN_VLAN | ADDRESS LOOKUP ENGINE UNKNOWN VLAN REGISTER   | <a href="#">Section 14.5.1.4</a>  |
| 20h    | TBLCTL       | ADDRESS LOOKUP ENGINE TABLE CONTROL           | <a href="#">Section 14.5.1.5</a>  |
| 34h    | TBLW2        | ADDRESS LOOKUP ENGINE TABLE WORD 2 REGISTER   | <a href="#">Section 14.5.1.6</a>  |
| 38h    | TBLW1        | ADDRESS LOOKUP ENGINE TABLE WORD 1 REGISTER   | <a href="#">Section 14.5.1.7</a>  |
| 3Ch    | TBLW0        | ADDRESS LOOKUP ENGINE TABLE WORD 0 REGISTER   | <a href="#">Section 14.5.1.8</a>  |
| 40h    | PORTCTL0     | ADDRESS LOOKUP ENGINE PORT 0 CONTROL REGISTER | <a href="#">Section 14.5.1.9</a>  |
| 44h    | PORTCTL1     | ADDRESS LOOKUP ENGINE PORT 1 CONTROL REGISTER | <a href="#">Section 14.5.1.10</a> |
| 48h    | PORTCTL2     | ADDRESS LOOKUP ENGINE PORT 2 CONTROL REGISTER | <a href="#">Section 14.5.1.11</a> |
| 4Ch    | PORTCTL3     | ADDRESS LOOKUP ENGINE PORT 3 CONTROL REGISTER | <a href="#">Section 14.5.1.12</a> |
| 50h    | PORTCTL4     | ADDRESS LOOKUP ENGINE PORT 4 CONTROL REGISTER | <a href="#">Section 14.5.1.13</a> |
| 54h    | PORTCTL5     | ADDRESS LOOKUP ENGINE PORT 5 CONTROL REGISTER | <a href="#">Section 14.5.1.14</a> |

#### 14.5.1.1 IDVER Register (offset = 0h) [reset = 290104h]

IDVER is shown in [Figure 14-15](#) and described in [Table 14-25](#).

#### ADDRESS LOOKUP ENGINE ID/VERSION REGISTER

**Figure 14-15. IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-25. IDVER Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description              |
|-------|-----------|------|-------|--------------------------|
| 31-16 | IDENT     | R-0  | 0     | ALE Identification Value |
| 15-8  | MAJ_VER   | R-0  | 0     | ALE Major Version Value  |
| 7-0   | MINOR_VER | R-0  | 0     | ALE Minor Version Value  |

#### 14.5.1.2 CONTROL Register (offset = 8h) [reset = 0h]

CONTROL is shown in [Figure 14-16](#) and described in [Table 14-26](#).

#### ADDRESS LOOKUP ENGINE CONTROL REGISTER

**Figure 14-16. CONTROL Register**

| 31           | 30           | 29              | 28     | 27            | 26         | 25               | 24                |
|--------------|--------------|-----------------|--------|---------------|------------|------------------|-------------------|
| ENABLE_ALE   | CLEAR_TABLE  | AGE_OUT_NOW     |        |               | Reserved   |                  |                   |
| R/W-0        | R/W-0        | R/W-0           |        |               |            |                  |                   |
| 23           | 22           | 21              | 20     | 19            | 18         | 17               | 16                |
|              |              |                 |        | Reserved      |            |                  |                   |
| 15           | 14           | 13              | 12     | 11            | 10         | 9                | 8                 |
|              |              |                 |        | Reserved      |            | EN_P0_UNI_FLOOD  |                   |
|              |              |                 |        |               |            |                  | R/W-0             |
| 7            | 6            | 5               | 4      | 3             | 2          | 1                | 0                 |
| LEARN_NO_VID | EN_VID0_MODE | ENABLE_OUI_DENY | BYPASS | RATE_LIMIT_TX | VLAN_AWARE | ENABLE_AUTH_MODE | ENABLE_RATE_LIMIT |
| R/W-0        | R/W-0        | R/W-0           | R/W-0  | R/W-0         | R/W-0      | R/W-0            | R/W-0             |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-26. CONTROL Register Field Descriptions**

| Bit | Field           | Type  | Reset | Description   |
|-----|-----------------|-------|-------|---|
| 31  | ENABLE_ALE      | R/W-0 | 0     | Enable ALE -<br>0 - Drop all packets<br>1 - Enable ALE packet processing  |
| 30  | CLEAR_TABLE     | R/W-0 | 0     | Clear ALE address table - Setting this bit causes the ALE hardware to write all table bit values to zero.<br>Software must perform a clear table operation as part of the ALE setup/configuration process.<br>Setting this bit causes all ALE accesses to be held up for 64 clocks while the clear is performed.<br>Access to all ALE registers will be blocked (wait states) until the 64 clocks have completed.<br>This bit cannot be read as one because the read is blocked until the clear table is completed at which time this bit is cleared to zero. |
| 29  | AGE_OUT_NOW     | R/W-0 | 0     | Age Out Address Table Now - Setting this bit causes the ALE hardware to remove (free up) any ageable table entry that does not have a set touch bit.<br>This bit is cleared when the age out process has completed.<br>This bit may be read.<br>The age out process takes 4096 clocks best case (no ale packet processing during ageout) and 66550 clocks absolute worst case.  |
| 8   | EN_P0_UNI_FLOOD | R/W-0 | 0     | Enable Port 0 (Host Port) unicast flood<br>0 - do not flood unknown unicast packets to host port (p0)<br>1 - flood unknown unicast packets to host port (p0)  |
| 7   | LEARN_NO_VID    | R/W-0 | 0     | Learn No VID -<br>0 - VID is learned with the source address<br>1 - VID is not learned with the source address (source address is not tied to VID).   |
| 6   | EN_VID0_MODE    | R/W-0 | 0     | Enable VLAN ID = 0 Mode<br>0 - Process the packet with VID = PORT_VLAN[11:0].<br>1 - Process the packet with VID = 0.   |
| 5   | ENABLE_OUI_DENY | R/W-0 | 0     | Enable OUI Deny Mode - When set this bit indicates that a packet with a non OUI table entry matching source address will be dropped to the host unless the destination address matches a multicast table entry with the super bit set.  |
| 4   | BYPASS          | R/W-0 | 0     | ALE Bypass - When set, all packets received on ports 0 and 1 are sent to the host (only to the host).   |

**Table 14-26. CONTROL Register Field Descriptions (continued)**

| Bit | Field             | Type  | Reset | Description   |
|-----|-------------------|-------|-------|---|
| 3   | RATE_LIMIT_TX     | R/W-0 | 0     | Rate Limit Transmit mode -<br>0 - Broadcast and multicast rate limit counters are received port based<br>1 - Broadcast and multicast rate limit counters are transmit port based.   |
| 2   | VLAN_AWARE        | R/W-0 | 0     | ALE VLAN Aware - Determines what is done if VLAN not found.<br>0 - Flood if VLAN not found<br>1 - Drop packet if VLAN not found   |
| 1   | ENABLE_AUTH_MODE  | R/W-0 | 0     | Enable MAC Authorization Mode - Mac authorization mode requires that all table entries be made by the host software.<br>There are no learned address in authorization mode and the packet will be dropped if the source address is not found (and the destination address is not a multicast address with the super table entry bit set).<br>0 - The ALE is not in MAC authorization mode<br>1 - The ALE is in MAC authorization mode |
| 0   | ENABLE_RATE_LIMIT | R/W-0 | 0     | Enable Broadcast and Multicast Rate Limit<br>0 - Broadcast/Multicast rates not limited<br>1 - Broadcast/Multicast packet reception limited to the port control register rate limit fields.  |

#### 14.5.1.3 PRESCALE Register (offset = 10h) [reset = 0h]

PREScale is shown in [Figure 14-17](#) and described in [Table 14-27](#).

##### ADDRESS LOOKUP ENGINE PRESCALE REGISTER

**Figure 14-17. PRESCALE Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16       | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-------|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PREScale |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R/W-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-27. PRESCALE Register Field Descriptions**

| Bit  | Field    | Type  | Reset | Description  |
|------|----------|-------|-------|--|
| 19-0 | PREScale | R/W-0 | 0     | ALE Prescale Register - The input clock is divided by this value for use in the multicast/broadcast rate limiters.<br>The minimum operating value is 0x10.<br>The prescaler is off when the value is zero. |

#### 14.5.1.4 UNKNOWN\_VLAN Register (offset = 18h) [reset = 0h]

UNKNOWN\_VLAN is shown in [Figure 14-18](#) and described in [Table 14-28](#).

##### ADDRESS LOOKUP ENGINE UNKNOWN VLAN REGISTER

**Figure 14-18. UNKNOWN\_VLAN Register**

|          |                               |    |    |    |    |    |    |
|----------|-------------------------------|----|----|----|----|----|----|
| 31       | 30                            | 29 | 28 | 27 | 26 | 25 | 24 |
| Reserved | UNKNOWN_FORCE_UNTAGGED_EGRESS |    |    |    |    |    |    |
| R/W-0    |                               |    |    |    |    |    |    |
| 23       | 22                            | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | UNKNOWN_REG_MCAST_FLOOD_MASK  |    |    |    |    |    |    |
| R/W-0    |                               |    |    |    |    |    |    |
| 15       | 14                            | 13 | 12 | 11 | 10 | 9  | 8  |
| Reserved | UNKNOWN_MCAST_FLOOD_MASK      |    |    |    |    |    |    |
| R/W-0    |                               |    |    |    |    |    |    |
| 7        | 6                             | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved | UNKNOWN_VLAN_MEMBER_LIST      |    |    |    |    |    |    |
| R/W-0    |                               |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-28. UNKNOWN\_VLAN Register Field Descriptions**

| Bit   | Field                         | Type  | Reset | Description                                  |
|-------|-------------------------------|-------|-------|--|
| 29-24 | UNKNOWN_FORCE_UNTAGGED_EGRESS | R/W-0 | 0     | Unknown VLAN Force Untagged Egress.          |
| 21-16 | UNKNOWN_REG_MCAST_FLOOD_MASK  | R/W-0 | 0     | Unknown VLAN Registered Multicast Flood Mask |
| 13-8  | UNKNOWN_MCAST_FLOOD_MASK      | R/W-0 | 0     | Unknown VLAN Multicast Flood Mask            |
| 5-0   | UNKNOWN_VLAN_MEMBER_LIST      | R/W-0 | 0     | Unknown VLAN Member List                     |

#### 14.5.1.5 TBLCTL Register (offset = 20h) [reset = 0h]

TBLCTL is shown in [Figure 14-19](#) and described in [Table 14-29](#).

##### ADDRESS LOOKUP ENGINE TABLE CONTROL

**Figure 14-19. TBLCTL Register**

| 31        | 30 | 29       | 28 | 27            | 26            | 25 | 24 |
|-----------|----|----------|----|---------------|---------------|----|----|
| WRITE_RDZ |    |          |    | Reserved      |               |    |    |
| R/W-0     |    |          |    |               |               |    |    |
| 23        | 22 | 21       | 20 | 19            | 18            | 17 | 16 |
|           |    |          |    | Reserved      |               |    |    |
| 15        | 14 | 13       | 12 | 11            | 10            | 9  | 8  |
|           |    | Reserved |    |               | ENTRY_POINTER |    |    |
|           |    |          |    |               | R/W-0         |    |    |
| 7         | 6  | 5        | 4  | 3             | 2             | 1  | 0  |
|           |    |          |    | ENTRY_POINTER |               |    |    |
|           |    |          |    |               | R/W-0         |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-29. TBLCTL Register Field Descriptions**

| Bit | Field         | Type  | Reset | Description   |
|-----|---------------|-------|-------|---|
| 31  | WRITE_RDZ     | R/W-0 | 0     | Write Bit - This bit is always read as zero. Writing a 1 to this bit causes the three table word register values to be written to the entry_pointer location in the address table. Writing a 0 to this bit causes the three table word register values to be loaded from the entry_pointer location in the address table so that they may be subsequently read. A read of any ALE address location will be stalled until the read or write has completed. |
| 9-0 | ENTRY_POINTER | R/W-0 | 0     | Table Entry Pointer - The entry_pointer contains the table entry value that will be read/written with accesses to the table word registers.   |

#### 14.5.1.6 TBLW2 Register (offset = 34h) [reset = 0h]

TBLW2 is shown in [Figure 14-20](#) and described in [Table 14-30](#).

#### ADDRESS LOOKUP ENGINE TABLE WORD 2 REGISTER

**Figure 14-20. TBLW2 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |            |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|------------|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5          | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | ENTRY71-64 |   |   |   |   |   |

R/W-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-30. TBLW2 Register Field Descriptions**

| Bit | Field      | Type  | Reset | Description               |
|-----|------------|-------|-------|---------------------------|
| 7-0 | ENTRY71-64 | R/W-0 | 0     | Table entry bits<br>71:64 |

#### **14.5.1.7 TBLW1 Register (offset = 38h) [reset = 0h]**

TBLW1 is shown in [Figure 14-21](#) and described in [Table 14-31](#).

ADDRESS LOOKUP ENGINE TABLE WORD 1 REGISTER

**Figure 14-21. TBLW1 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ENTRY63_32 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-31. TBLW1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description            |
|------|------------|------|-------|------------------------|
| 31-0 | ENTRY63_32 | R/W  | 0h    | Table entry bits 63:32 |

#### 14.5.1.8 TBLW0 Register (offset = 3Ch) [reset = 0h]

TBLW0 is shown in [Figure 14-22](#) and described in [Table 14-32](#).

##### ADDRESS LOOKUP ENGINE TABLE WORD 0 REGISTER

**Figure 14-22. TBLW0 Register**

| 31        | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ENTRY31_0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-32. TBLW0 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description           |
|------|-----------|------|-------|-----------------------|
| 31-0 | ENTRY31_0 | R/W  | 0h    | Table entry bits 31:0 |

#### 14.5.1.9 PORTCTL0 Register (offset = 40h) [reset = 0h]

PORTCTL0 is shown in [Figure 14-23](#) and described in [Table 14-33](#).

##### ADDRESS LOOKUP ENGINE PORT 0 CONTROL REGISTER

**Figure 14-23. PORTCTL0 Register**

| 31          | 30           | 29       | 28                | 27            | 26         | 25 | 24 |
|-------------|--------------|----------|-------------------|---------------|------------|----|----|
| BCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17 | 16 |
| MCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9  | 8  |
| Reserved    |              |          |                   |               |            |    |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1  | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |    |    |
|             | R/W-0        | R/W-0    | R/W-0             | R/W-0         | R/W-0      |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-33. PORTCTL0 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn<br>3 - Forward   |

#### 14.5.1.10 PORTCTL1 Register (offset = 44h) [reset = 0h]

PORTCTL1 is shown in [Figure 14-24](#) and described in [Table 14-34](#).

##### ADDRESS LOOKUP ENGINE PORT 1 CONTROL REGISTER

**Figure 14-24. PORTCTL1 Register**

|             |              |          |                   |               |            |       |    |
|-------------|--------------|----------|-------------------|---------------|------------|-------|----|
| 31          | 30           | 29       | 28                | 27            | 26         | 25    | 24 |
| BCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17    | 16 |
| MCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9     | 8  |
| Reserved    |              |          |                   |               |            |       |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1     | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |       |    |
| R/W-0       |              | R/W-0    | R/W-0             | R/W-0         | R/W-0      | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-34. PORTCTL1 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn<br>3 - Forward   |

#### 14.5.1.11 PORTCTL2 Register (offset = 48h) [reset = 0h]

PORTCTL2 is shown in [Figure 14-25](#) and described in [Table 14-35](#).

##### ADDRESS LOOKUP ENGINE PORT 2 CONTROL REGISTER

**Figure 14-25. PORTCTL2 Register**

| 31          | 30           | 29       | 28                | 27            | 26         | 25 | 24 |
|-------------|--------------|----------|-------------------|---------------|------------|----|----|
| BCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17 | 16 |
| MCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9  | 8  |
| Reserved    |              |          |                   |               |            |    |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1  | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |    |    |
|             | R/W-0        | R/W-0    | R/W-0             | R/W-0         | R/W-0      |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-35. PORTCTL2 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn<br>3 - Forward   |

#### 14.5.1.12 PORTCTL3 Register (offset = 4Ch) [reset = 0h]

PORTCTL3 is shown in [Figure 14-26](#) and described in [Table 14-36](#).

##### ADDRESS LOOKUP ENGINE PORT 3 CONTROL REGISTER

**Figure 14-26. PORTCTL3 Register**

|             |              |          |                   |               |            |       |    |
|-------------|--------------|----------|-------------------|---------------|------------|-------|----|
| 31          | 30           | 29       | 28                | 27            | 26         | 25    | 24 |
| BCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17    | 16 |
| MCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9     | 8  |
| Reserved    |              |          |                   |               |            |       |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1     | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |       |    |
| R/W-0       |              | R/W-0    | R/W-0             | R/W-0         | R/W-0      | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-36. PORTCTL3 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn<br>3 - Forward   |

#### 14.5.1.13 PORTCTL4 Register (offset = 50h) [reset = 0h]

PORTCTL4 is shown in [Figure 14-27](#) and described in [Table 14-37](#).

##### ADDRESS LOOKUP ENGINE PORT 4 CONTROL REGISTER

**Figure 14-27. PORTCTL4 Register**

| 31          | 30           | 29       | 28                | 27            | 26         | 25 | 24 |
|-------------|--------------|----------|-------------------|---------------|------------|----|----|
| BCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17 | 16 |
| MCAST_LIMIT |              |          |                   |               |            |    |    |
| R/W-0       |              |          |                   |               |            |    |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9  | 8  |
| Reserved    |              |          |                   |               |            |    |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1  | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |    |    |
|             | R/W-0        | R/W-0    | R/W-0             | R/W-0         | R/W-0      |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-37. PORTCTL4 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn<br>3 - Forward   |

#### 14.5.1.14 PORTCTL5 Register (offset = 54h) [reset = 0h]

PORTCTL5 is shown in [Figure 14-28](#) and described in [Table 14-38](#).

##### ADDRESS LOOKUP ENGINE PORT 5 CONTROL REGISTER

**Figure 14-28. PORTCTL5 Register**

|             |              |          |                   |               |            |       |    |
|-------------|--------------|----------|-------------------|---------------|------------|-------|----|
| 31          | 30           | 29       | 28                | 27            | 26         | 25    | 24 |
| BCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 23          | 22           | 21       | 20                | 19            | 18         | 17    | 16 |
| MCAST_LIMIT |              |          |                   |               |            |       |    |
| R/W-0       |              |          |                   |               |            |       |    |
| 15          | 14           | 13       | 12                | 11            | 10         | 9     | 8  |
| Reserved    |              |          |                   |               |            |       |    |
| 7           | 6            | 5        | 4                 | 3             | 2          | 1     | 0  |
| Reserved    | NO_SA_UPDATE | NO_LEARN | VID_INGRESS_CHECK | DROP_UNTAGGED | PORT_STATE |       |    |
| R/W-0       |              | R/W-0    | R/W-0             | R/W-0         | R/W-0      | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-38. PORTCTL5 Register Field Descriptions**

| Bit   | Field             | Type  | Reset | Description   |
|-------|-------------------|-------|-------|---|
| 31-24 | BCAST_LIMIT       | R/W-0 | 0     | Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Broadcast rate limiting is enabled by a non-zero value in this field  |
| 23-16 | MCAST_LIMIT       | R/W-0 | 0     | Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter.<br>The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse.<br>Multicast rate limiting is enabled by a non-zero value in this field. |
| 5     | NO_SA_UPDATE      | R/W-0 | 0     | No Source Address Update - When set the port is disabled from updating the source port number in an ALE table entry.  |
| 4     | NO_LEARN          | R/W-0 | 0     | No Learn Mode - When set the port is disabled from learning an address.   |
| 3     | VID_INGRESS_CHECK | R/W-0 | 0     | VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.   |
| 2     | DROP_UNTAGGED     | R/W-0 | 0     | Drop Untagged Packets - Drop non-VLAN tagged ingress packets.   |
| 1-0   | PORT_STATE        | R/W-0 | 0     | Port State<br>0 - Disabled<br>1 - Blocked<br>2 - Learn  |

#### 14.5.2 CPSW\_CPDMA Registers

[Table 14-39](#) lists the memory-mapped registers for the CPSW\_CPDMA. All register offset addresses not listed in [Table 14-39](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-39. CPSW\_CPDMA REGISTERS**

| Offset | Acronym            | Register Name  | Section                           |
|--------|--------------------|--|-----------------------------------|
| 0h     | TX_IDVER           | CPDMA_REGS TX IDENTIFICATION AND VERSION REGISTER      | <a href="#">Section 14.5.2.1</a>  |
| 4h     | TX_CONTROL         | CPDMA_REGS TX CONTROL REGISTER                         | <a href="#">Section 14.5.2.2</a>  |
| 8h     | TX_TEARDOWN        | CPDMA_REGS TX TEARDOWN REGISTER                        | <a href="#">Section 14.5.2.3</a>  |
| 10h    | RX_IDVER           | CPDMA_REGS RX IDENTIFICATION AND VERSION REGISTER      | <a href="#">Section 14.5.2.4</a>  |
| 14h    | RX_CONTROL         | CPDMA_REGS RX CONTROL REGISTER                         | <a href="#">Section 14.5.2.5</a>  |
| 18h    | RX_TEARDOWN        | CPDMA_REGS RX TEARDOWN REGISTER                        | <a href="#">Section 14.5.2.6</a>  |
| 1Ch    | CPDMA_SOFT_RESET   | CPDMA_REGS SOFT RESET REGISTER                         | <a href="#">Section 14.5.2.7</a>  |
| 20h    | DMACONTROL         | CPDMA_REGS CPDMA CONTROL REGISTER                      | <a href="#">Section 14.5.2.8</a>  |
| 24h    | DMASTATUS          | CPDMA_REGS CPDMA STATUS REGISTER                       | <a href="#">Section 14.5.2.9</a>  |
| 28h    | RX_BUFFER_OFFSET   | CPDMA_REGS RECEIVE BUFFER OFFSET                       | <a href="#">Section 14.5.2.10</a> |
| 2Ch    | EMCONTROL          | CPDMA_REGS EMULATION CONTROL                           | <a href="#">Section 14.5.2.11</a> |
| 30h    | TX_PRI0_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 0 RATE          | <a href="#">Section 14.5.2.12</a> |
| 34h    | TX_PRI1_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 1 RATE          | <a href="#">Section 14.5.2.13</a> |
| 38h    | TX_PRI2_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 2 RATE          | <a href="#">Section 14.5.2.14</a> |
| 3Ch    | TX_PRI3_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 3 RATE          | <a href="#">Section 14.5.2.15</a> |
| 40h    | TX_PRI4_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 4 RATE          | <a href="#">Section 14.5.2.16</a> |
| 44h    | TX_PRI5_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 5 RATE          | <a href="#">Section 14.5.2.17</a> |
| 48h    | TX_PRI6_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 6 RATE          | <a href="#">Section 14.5.2.18</a> |
| 4Ch    | TX_PRI7_RATE       | CPDMA_REGS TRANSMIT (INGRESS) PRIORITY 7 RATE          | <a href="#">Section 14.5.2.19</a> |
| 80h    | TX_INTSTAT_RAW     | CPDMA_INT TX INTERRUPT STATUS REGISTER (RAW VALUE)     | <a href="#">Section 14.5.2.20</a> |
| 84h    | TX_INTSTAT_MASKED  | CPDMA_INT TX INTERRUPT STATUS REGISTER (MASKED VALUE)  | <a href="#">Section 14.5.2.21</a> |
| 88h    | TX_INTMASK_SET     | CPDMA_INT TX INTERRUPT MASK SET REGISTER               | <a href="#">Section 14.5.2.22</a> |
| 8Ch    | TX_INTMASK_CLEAR   | CPDMA_INT TX INTERRUPT MASK CLEAR REGISTER             | <a href="#">Section 14.5.2.23</a> |
| 90h    | CPDMA_IN_VECTOR    | CPDMA_INT INPUT VECTOR (READ ONLY)                     | <a href="#">Section 14.5.2.24</a> |
| 94h    | CPDMA_EOI_VECTOR   | CPDMA_INT END OF INTERRUPT VECTOR                      | <a href="#">Section 14.5.2.25</a> |
| A0h    | RX_INTSTAT_RAW     | CPDMA_INT RX INTERRUPT STATUS REGISTER (RAW VALUE)     | <a href="#">Section 14.5.2.26</a> |
| A4h    | RX_INTSTAT_MASKED  | CPDMA_INT RX INTERRUPT STATUS REGISTER (MASKED VALUE)  | <a href="#">Section 14.5.2.27</a> |
| A8h    | RX_INTMASK_SET     | CPDMA_INT RX INTERRUPT MASK SET REGISTER               | <a href="#">Section 14.5.2.28</a> |
| ACh    | RX_INTMASK_CLEAR   | CPDMA_INT RX INTERRUPT MASK CLEAR REGISTER             | <a href="#">Section 14.5.2.29</a> |
| B0h    | DMA_INTSTAT_RAW    | CPDMA_INT DMA INTERRUPT STATUS REGISTER (RAW VALUE)    | <a href="#">Section 14.5.2.30</a> |
| B4h    | DMA_INTSTAT_MASKED | CPDMA_INT DMA INTERRUPT STATUS REGISTER (MASKED VALUE) | <a href="#">Section 14.5.2.31</a> |
| B8h    | DMA_INTMASK_SET    | CPDMA_INT DMA INTERRUPT MASK SET REGISTER              | <a href="#">Section 14.5.2.32</a> |
| BCh    | DMA_INTMASK_CLEAR  | CPDMA_INT DMA INTERRUPT MASK CLEAR REGISTER            | <a href="#">Section 14.5.2.33</a> |
| C0h    | RX0_PENDTHRESH     | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 0 | <a href="#">Section 14.5.2.34</a> |

**Table 14-39. CPSW\_CPDMA REGISTERS (continued)**

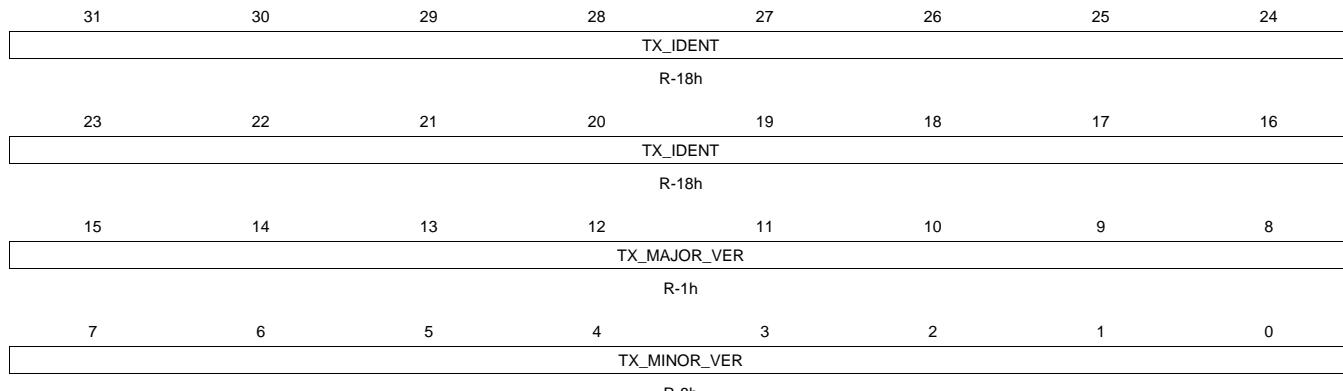
| <b>Offset</b>   | <b>Acronym</b> | <b>Register Name</b>                                   | <b>Section</b>                    |
|-----------------|----------------|--|-----------------------------------|
| C4h             | RX1_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 1 | <a href="#">Section 14.5.2.35</a> |
| C8h             | RX2_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 2 | <a href="#">Section 14.5.2.36</a> |
| CCh             | RX3_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 3 | <a href="#">Section 14.5.2.37</a> |
| D0h             | RX4_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 4 | <a href="#">Section 14.5.2.38</a> |
| D4h             | RX5_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 5 | <a href="#">Section 14.5.2.39</a> |
| D8h             | RX6_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 6 | <a href="#">Section 14.5.2.40</a> |
| DCh             | RX7_PENDTHRESH | CPDMA_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 7 | <a href="#">Section 14.5.2.41</a> |
| E0h             | RX0_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 0       | <a href="#">Section 14.5.2.42</a> |
| E4h             | RX1_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 1       | <a href="#">Section 14.5.2.43</a> |
| E8h             | RX2_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 2       | <a href="#">Section 14.5.2.44</a> |
| EC <sub>h</sub> | RX3_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 3       | <a href="#">Section 14.5.2.45</a> |
| F0h             | RX4_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 4       | <a href="#">Section 14.5.2.46</a> |
| F4h             | RX5_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 5       | <a href="#">Section 14.5.2.47</a> |
| F8h             | RX6_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 6       | <a href="#">Section 14.5.2.48</a> |
| FC <sub>h</sub> | RX7_FREEBUFFER | CPDMA_INT RECEIVE FREE BUFFER REGISTER CHANNEL 7       | <a href="#">Section 14.5.2.49</a> |

#### 14.5.2.1 TX\_IDVER Register (offset = 0h) [reset = 180108h]

TX\_IDVER is shown in [Figure 14-29](#) and described in [Table 14-40](#).

##### CPDMA\_REGS TX IDENTIFICATION AND VERSION REGISTER

**Figure 14-29. TX\_IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-40. TX\_IDVER Register Field Descriptions**

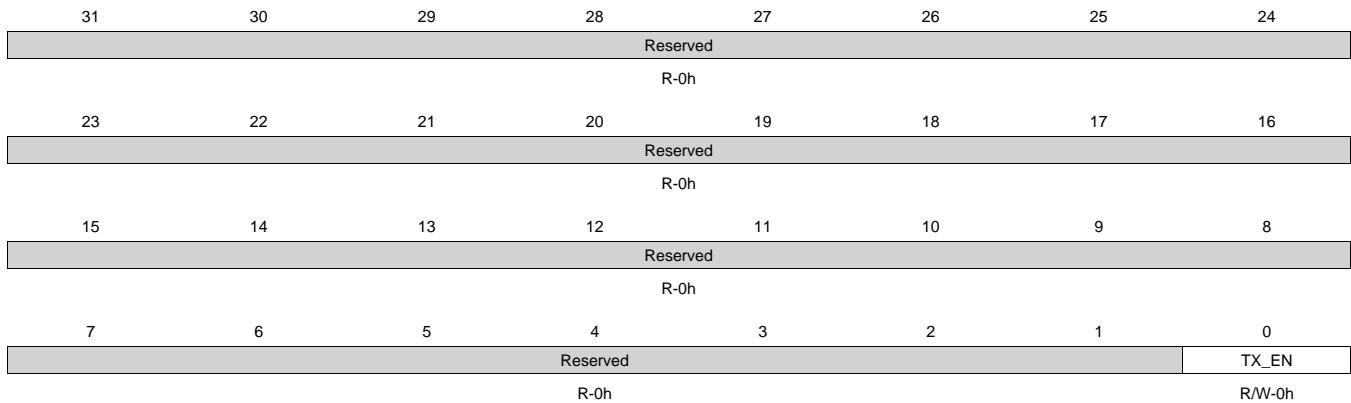
| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31-16 | TX_IDENT     | R    | 18h   | TX Identification Value   |
| 15-8  | TX_MAJOR_VER | R    | 1h    | TX Major Version Value - The value read is the major version number |
| 7-0   | TX_MINOR_VER | R    | 8h    | TX Minor Version Value - The value read is the minor version number |

#### 14.5.2.2 TX\_CONTROL Register (offset = 4h) [reset = 0h]

TX\_CONTROL is shown in [Figure 14-30](#) and described in [Table 14-41](#).

##### CPDMA\_REGS TX CONTROL REGISTER

**Figure 14-30. TX\_CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-41. TX\_CONTROL Register Field Descriptions**

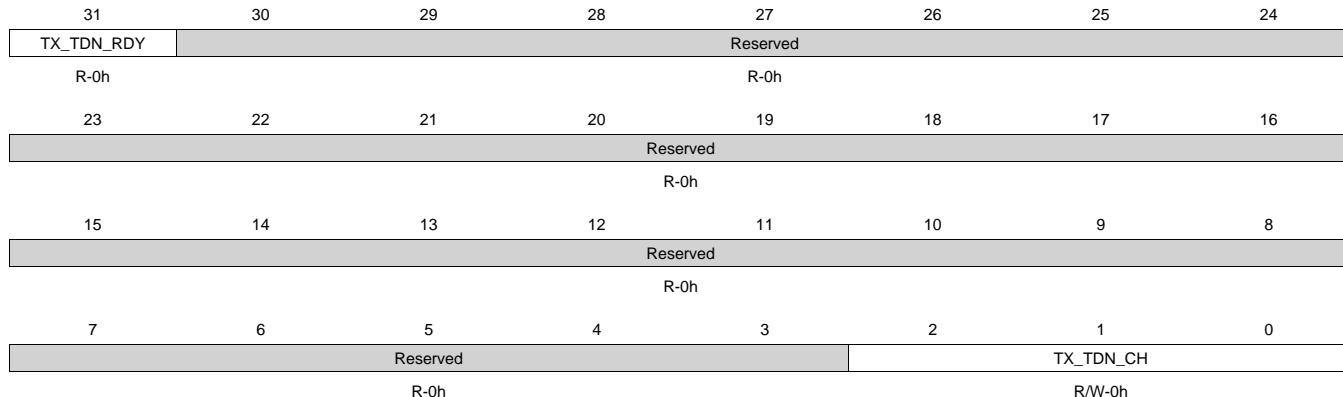
| Bit  | Field    | Type | Reset | Description                              |
|------|----------|------|-------|--|
| 31-1 | Reserved | R    | 0h    |  |
| 0    | TX_EN    | R/W  | 0h    | TX Enable<br>0 - Disabled<br>1 - Enabled |

#### 14.5.2.3 TX\_TEARDOWN Register (offset = 8h) [reset = 0h]

TX\_TEARDOWN is shown in [Figure 14-31](#) and described in [Table 14-42](#).

##### CPDMA\_REGS TX TEARDOWN REGISTER

**Figure 14-31. TX\_TEARDOWN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-42. TX\_TEARDOWN Register Field Descriptions**

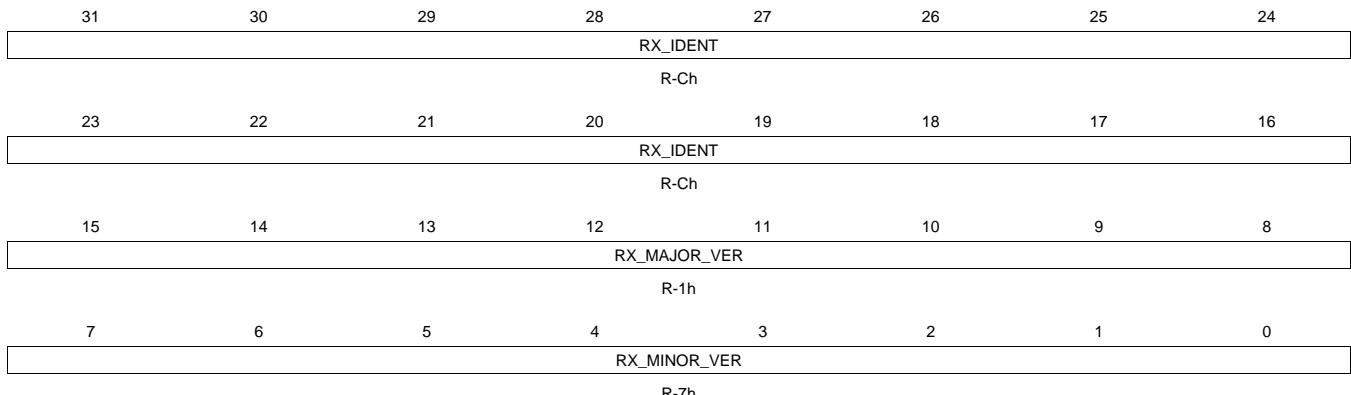
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31   | TX_TDNRDY | R    | 0h    | Tx Teardown Ready - read as zero, but is always assumed to be one (unused).  |
| 30-3 | Reserved  | R    | 0h    |  |
| 2-0  | TX_TDNRDY | R/W  | 0h    | Tx Teardown Channel - Transmit channel teardown is commanded by writing the encoded value of the transmit channel to be torn down.<br>The teardown register is read as zero.<br>000 - teardown transmit channel 0 ...<br>111 - teardown transmit channel 7 |

#### 14.5.2.4 RX\_IDVER Register (offset = 10h) [reset = C0107h]

RX\_IDVER is shown in [Figure 14-32](#) and described in [Table 14-43](#).

##### CPDMA\_REGS RX IDENTIFICATION AND VERSION REGISTER

**Figure 14-32. RX\_IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-43. RX\_IDVER Register Field Descriptions**

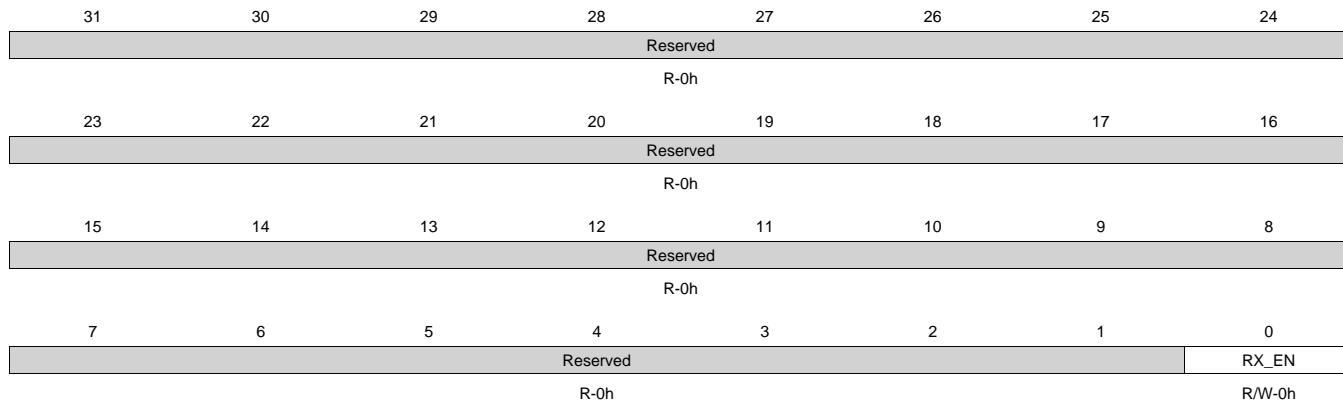
| Bit   | Field        | Type | Reset | Description             |
|-------|--------------|------|-------|-------------------------|
| 31-16 | RX_IDENT     | R    | Ch    | RX Identification Value |
| 15-8  | RX_MAJOR_VER | R    | 1h    | RX Major Version Value  |
| 7-0   | RX_MINOR_VER | R    | 7h    | RX Minor Version Value  |

#### 14.5.2.5 RX\_CONTROL Register (offset = 14h) [reset = 0h]

RX\_CONTROL is shown in [Figure 14-33](#) and described in [Table 14-44](#).

##### CPDMA\_REGS RX CONTROL REGISTER

**Figure 14-33. RX\_CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-44. RX\_CONTROL Register Field Descriptions**

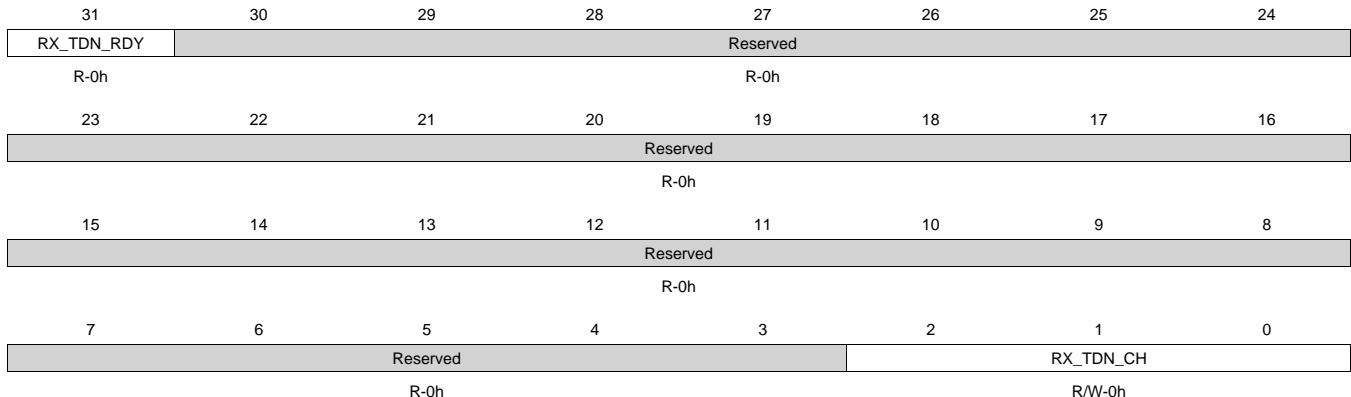
| Bit  | Field    | Type | Reset | Description                                  |
|------|----------|------|-------|--|
| 31-1 | Reserved | R    | 0h    |  |
| 0    | RX_EN    | R/W  | 0h    | RX DMA Enable<br>0 - Disabled<br>1 - Enabled |

#### 14.5.2.6 RX\_TEARDOWN Register (offset = 18h) [reset = 0h]

RX\_TEARDOWN is shown in [Figure 14-34](#) and described in [Table 14-45](#).

##### CPDMA\_REGS RX TEARDOWN REGISTER

**Figure 14-34. RX\_TEARDOWN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-45. RX\_TEARDOWN Register Field Descriptions**

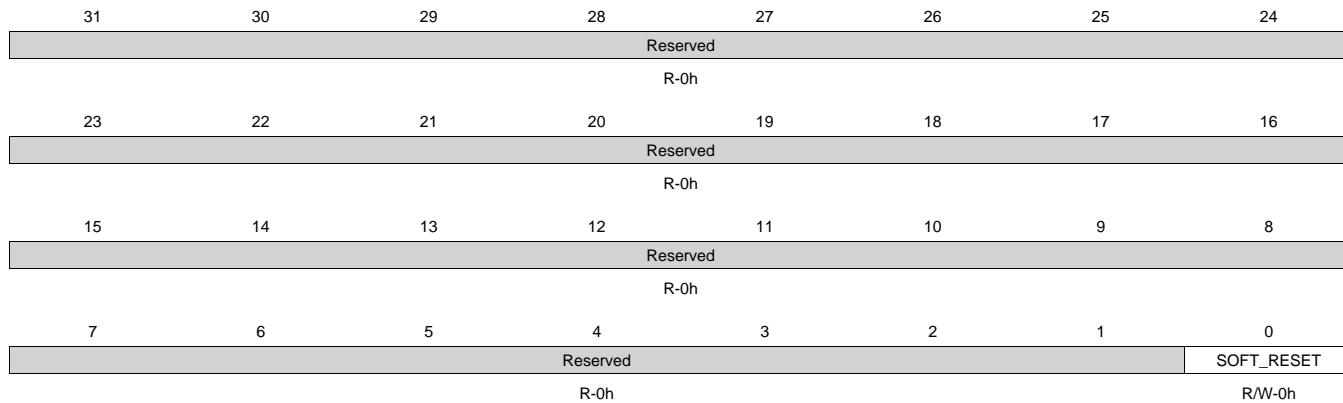
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31   | RX_TDN_RDY | R    | 0h    | Teardown Ready - read as zero, but is always assumed to be one (unused).   |
| 30-3 | Reserved   | R    | 0h    |  |
| 2-0  | RX_TDN_CH  | R/W  | 0h    | Rx Teardown Channel -Receive channel teardown is commanded by writing the encoded value of the receive channel to be torn down. The teardown register is read as zero.<br>000 - teardown receive channel 0 ...<br>111 - teardown receive channel 7 |

#### 14.5.2.7 CPDMA\_SOFT\_RESET Register (offset = 1Ch) [reset = 0h]

CPDMA\_SOFT\_RESET is shown in [Figure 14-35](#) and described in [Table 14-46](#).

##### CPDMA\_REGS SOFT RESET REGISTER

**Figure 14-35. CPDMA\_SOFT\_RESET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-46. CPDMA\_SOFT\_RESET Register Field Descriptions**

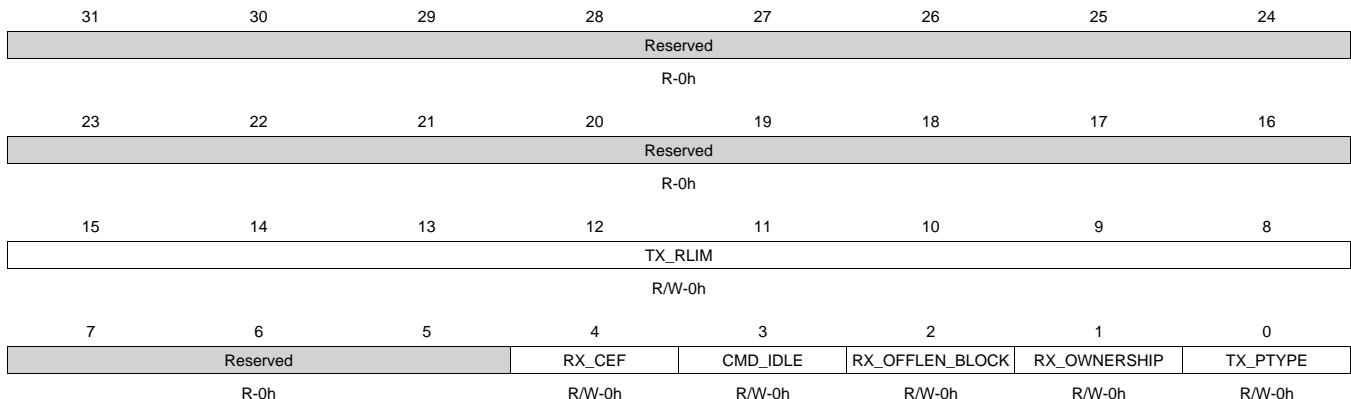
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-1 | Reserved   | R    | 0h    |   |
| 0    | SOFT_RESET | R/W  | 0h    | <p>Software reset - Writing a one to this bit causes the CPDMA logic to be reset.</p> <p>Software reset occurs when the RX and TX DMA Controllers are in an idle state to avoid locking up the VBUSP bus.</p> <p>After writing a one to this bit, it may be polled to determine if the reset has occurred.</p> <p>If a one is read, the reset has not yet occurred.</p> <p>If a zero is read then reset has occurred.</p> |

#### 14.5.2.8 DMACONTROL Register (offset = 20h) [reset = 0h]

DMACONTROL is shown in [Figure 14-36](#) and described in [Table 14-47](#).

##### CPDMA\_REGS CPDMA CONTROL REGISTER

**Figure 14-36. DMACONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-47. DMACONTROL Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-16 | Reserved        | R    | 0h    |   |
| 15-8  | TX_RLIM         | R/W  | 0h    | Transmit Rate Limit Channel Bus<br>00000000 - no rate-limited channels<br>10000000 - channel 7 is rate-limited<br>11000000 - channels 7 downto 6 are rate-limited<br>11100000 - channels 7 downto 5 are rate-limited<br>11110000 - channels 7 downto 4 are rate-limited<br>11111000 - channels 7 downto 3 are rate-limited<br>11111100 - channels 7 downto 2 are rate-limited<br>11111110 - channels 7 downto 1 are rate-limited<br>11111111 - channels 7 downto 0 are rate-limited all others invalid -<br>this bus must be set msb towards lsb.<br>tx_ptype must be set if any tx_rlim bit is set for fixed priority. |
| 7-5   | Reserved        | R    | 0h    |   |
| 4     | RX_CEF          | R/W  | 0h    | RX Copy Error Frames Enable - Enables DMA overrun frames to be transferred to memory (up to the point of overrun).<br>The overrun error bit will be set in the frame EOP buffer descriptor.<br>Overrun frame data will be filtered when rx_cef is not set.<br>Frames coming from the receive FIFO with other error bits set are not effected by this bit.<br>0 - Frames containing overrun errors are filtered.<br>1 - Frames containing overrun errors are transferred to memory.  |
| 3     | CMD_IDLE        | R/W  | 0h    | Command Idle<br>0 - Idle not commanded<br>1 - Idle Commanded (read idle in DMAStatus)   |
| 2     | RX_OFFSET_BLOCK | R/W  | 0h    | Receive Offset/Length word write block.<br>0 - Do not block the DMA writes to the receive buffer descriptor offset/buffer length word.<br>The offset/buffer length word is written as specified in CPPI 3.0.<br>1 - Block all CPDMA DMA controller writes to the receive buffer descriptor offset/buffer length words during CPPI packet processing.<br>when this bit is set, the CPDMA will never write the third word to any receive buffer descriptor.   |

**Table 14-47. DMACONTROL Register Field Descriptions (continued)**

| Bit | Field        | Type | Reset | Description   |
|-----|--------------|------|-------|---|
| 1   | RX_OWNERSHIP | R/W  | 0h    | <p>Receive Ownership Write Bit Value.</p> <p>0 - The CPDMA writes the receive ownership bit to zero at the end of packet processing as specified in CPPI 3.0.</p> <p>1 - The CPDMA writes the receive ownership bit to one at the end of packet processing.</p> <p>Users who do not use the ownership mechanism can use this mode to preclude the necessity of software having to set this bit each time the buffer descriptor is used.</p> |
| 0   | TX_PTYPE     | R/W  | 0h    | <p>Transmit Queue Priority Type</p> <p>0 - The queue uses a round robin scheme to select the next channel for transmission.</p> <p>1 - The queue uses a fixed (channel 7 highest priority) priority scheme to select the next channel for transmission</p>  |

#### 14.5.2.9 DMASTATUS Register (offset = 24h) [reset = 0h]

DMASTATUS is shown in [Figure 14-37](#) and described in [Table 14-48](#).

##### CPDMA\_REGS CPDMA STATUS REGISTER

**Figure 14-37. DMASTATUS Register**

| 31   | 30 | 29 | 28   | 27       | 26   | 25        | 24   |
|------|----|----|------|----------|------|-----------|------|
| IDLE |    |    |      | Reserved |      |           |      |
| R-0h |    |    |      | R-0h     |      |           |      |
| 23   | 22 | 21 | 20   | 19       | 18   | 17        | 16   |
|      |    |    |      | Reserved |      | TX_ERR_CH |      |
|      |    |    | R-0h |          | R-0h |           | R-0h |
| 15   | 14 | 13 | 12   | 11       | 10   | 9         | 8    |
|      |    |    |      | Reserved |      | RX_ERR_CH |      |
|      |    |    | R-0h |          | R-0h |           | R-0h |
| 7    | 6  | 5  | 4    | 3        | 2    | 1         | 0    |
|      |    |    |      | Reserved |      |           |      |
|      |    |    |      | R-0h     |      |           |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-48. DMASTATUS Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31    | IDLE             | R    | 0h    | Idle Status Bit - Indicates when set that the CPDMA is not transferring a packet on transmit or receive.   |
| 30-24 | Reserved         | R    | 0h    |  |
| 23-20 | TX_HOST_ERR_CODE | R    | 0h    | TX Host Error Code - This field is set to indicate CPDMA detected TX DMA related host errors.<br>The host should read this field after a HOST_ERR_INT to determine the error.<br>Host error Interrupts require hardware reset in order to recover.<br>A zero packet length is an error, but it is not detected.<br>0000 - No error<br>0001 - SOP error.<br>0010 - Ownership bit not set in SOP buffer.<br>0011 - Zero Next Buffer Descriptor Pointer Without EOP<br>0100 - Zero Buffer Pointer.<br>0101 - Zero Buffer Length<br>0110 - Packet Length Error (sum of buffers is less than packet length)<br>0111 - reserved ...<br>1111 - reserved |
| 19    | Reserved         | R    | 0h    |  |
| 18-16 | TX_ERR_CH        | R    | 0h    | TX Host Error Channel - This field indicates which TX channel (if applicable) the host error occurred on.<br>This field is cleared to zero on a host read.<br>000 - The host error occurred on TX channel 0 ...<br>111 - The host error occurred on TX channel 7   |
| 15-12 | RX_HOST_ERR_CODE | R    | 0h    | RX Host Error Code - This field is set to indicate CPDMA detected RX DMA related host errors.<br>The host should read this field after a HOST_ERR_INT to determine the error.<br>Host error Interrupts require hardware reset in order to recover.<br>0000 - No error<br>0001 - reserved<br>0010 - Ownership bit not set in input buffer.<br>0011 - reserved<br>0100 - Zero Buffer Pointer.<br>0101 - Zero buffer length on non-SOP descriptor<br>0110 - SOP buffer length not greater than offset ...<br>1111 - reserved  |

**Table 14-48. DMASTATUS Register Field Descriptions (continued)**

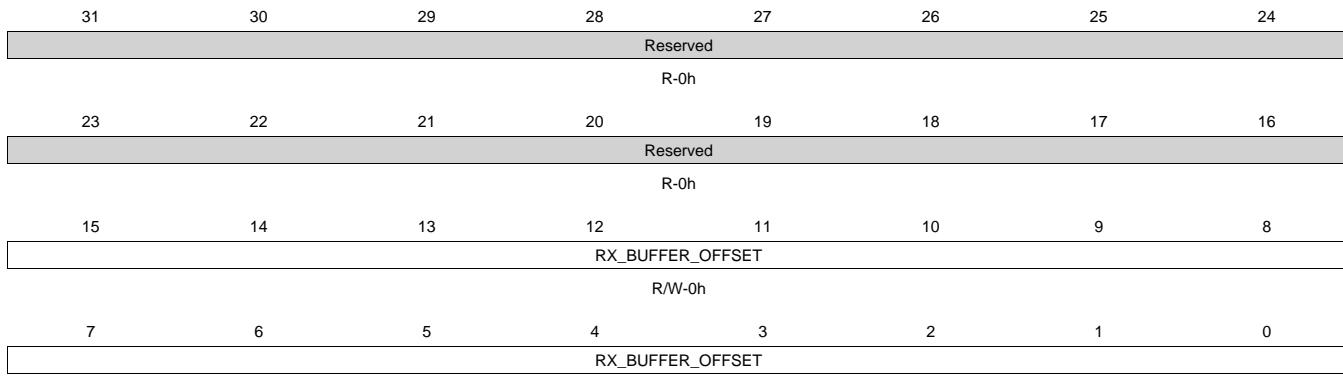
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 11   | Reserved  | R    | 0h    |  |
| 10-8 | RX_ERR_CH | R    | 0h    | RX Host Error Channel - This field indicates which RX channel the host error occurred on.<br>This field is cleared to zero on a host read.<br>000 - The host error occurred on RX channel 0 ...<br>111 - The host error occurred on RX channel 7 |
| 7-0  | Reserved  | R    | 0h    |  |

#### 14.5.2.10 RX\_BUFFER\_OFFSET Register (offset = 28h) [reset = 0h]

RX\_BUFFER\_OFFSET is shown in [Figure 14-38](#) and described in [Table 14-49](#).

CPDMA\_REGS RECEIVE BUFFER OFFSET

**Figure 14-38. RX\_BUFFER\_OFFSET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-49. RX\_BUFFER\_OFFSET Register Field Descriptions**

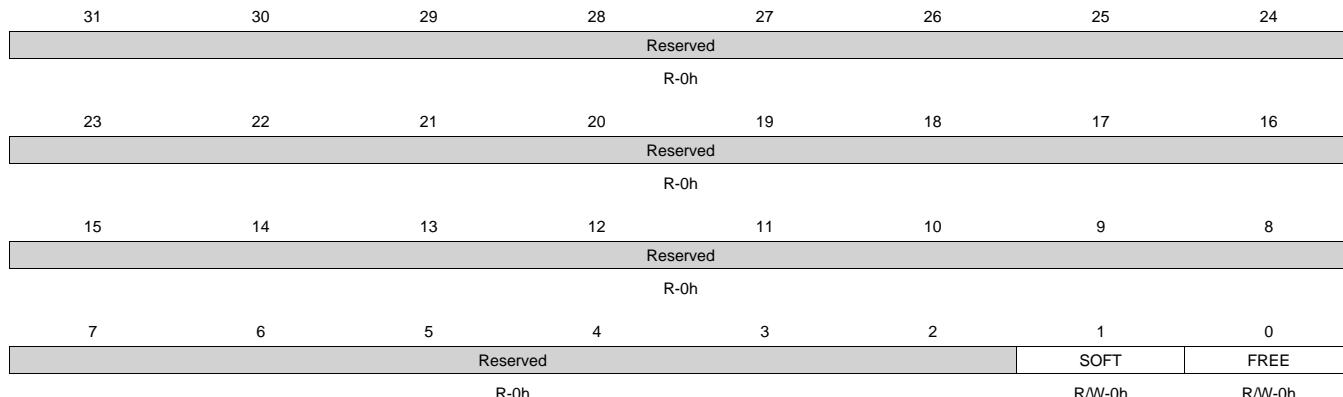
| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-16 | Reserved         | R    | 0h    |   |
| 15-0  | RX_BUFFER_OFFSET | R/W  | 0h    | <p>Receive Buffer Offset Value - The rx_buffer_offset will be written by the port into each frame SOP buffer descriptor buffer_offset field. The frame data will begin after the rx_buffer_offset value of bytes. A value of 0x0000 indicates that there are no unused bytes at the beginning of the data and that valid data begins on the first byte of the buffer.</p> <p>A value of 0x000F (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer.</p> <p>This value is used for all channels.</p> |

#### 14.5.2.11 EMCONTROL Register (offset = 2Ch) [reset = 0h]

EMCONTROL is shown in [Figure 14-39](#) and described in [Table 14-50](#).

CPDMA\_REGS EMULATION CONTROL

**Figure 14-39. EMCONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-50. EMCONTROL Register Field Descriptions**

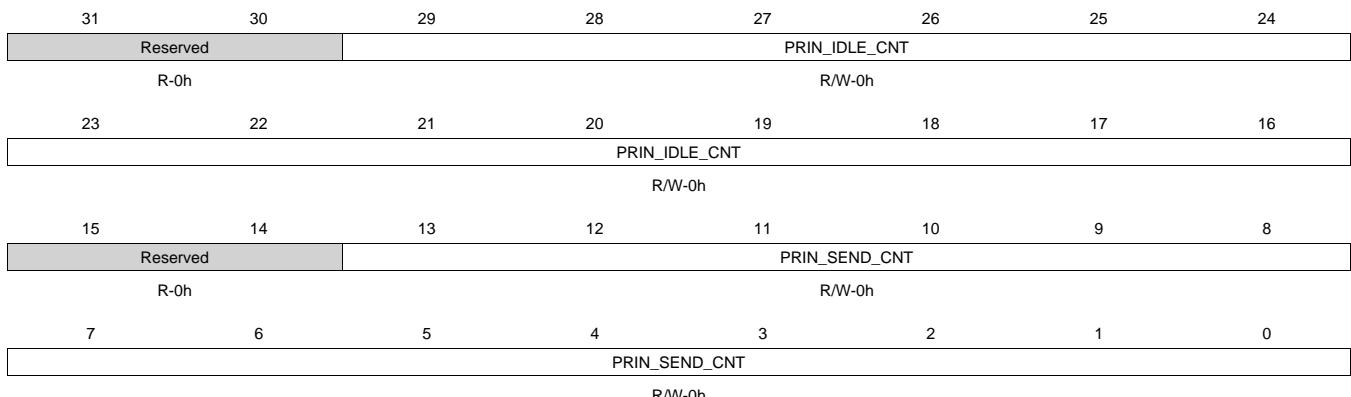
| Bit  | Field    | Type | Reset | Description        |
|------|----------|------|-------|--------------------|
| 31-2 | Reserved | R    | 0h    |                    |
| 1    | SOFT     | R/W  | 0h    | Emulation Soft Bit |
| 0    | FREE     | R/W  | 0h    | Emulation Free Bit |

#### 14.5.2.12 TX\_PRI0\_RATE Register (offset = 30h) [reset = 0h]

TX\_PRI0\_RATE is shown in [Figure 14-40](#) and described in [Table 14-51](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 0 RATE

**Figure 14-40. TX\_PRI0\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-51. TX\_PRI0\_RATE Register Field Descriptions**

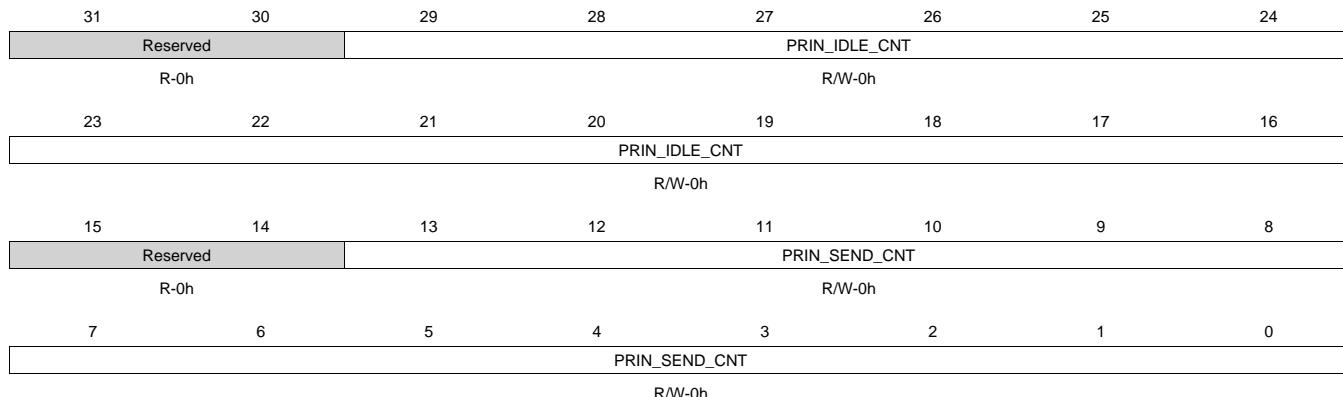
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.13 TX\_PRI1\_RATE Register (offset = 34h) [reset = 0h]

TX\_PRI1\_RATE is shown in [Figure 14-41](#) and described in [Table 14-52](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 1 RATE

**Figure 14-41. TX\_PRI1\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-52. TX\_PRI1\_RATE Register Field Descriptions**

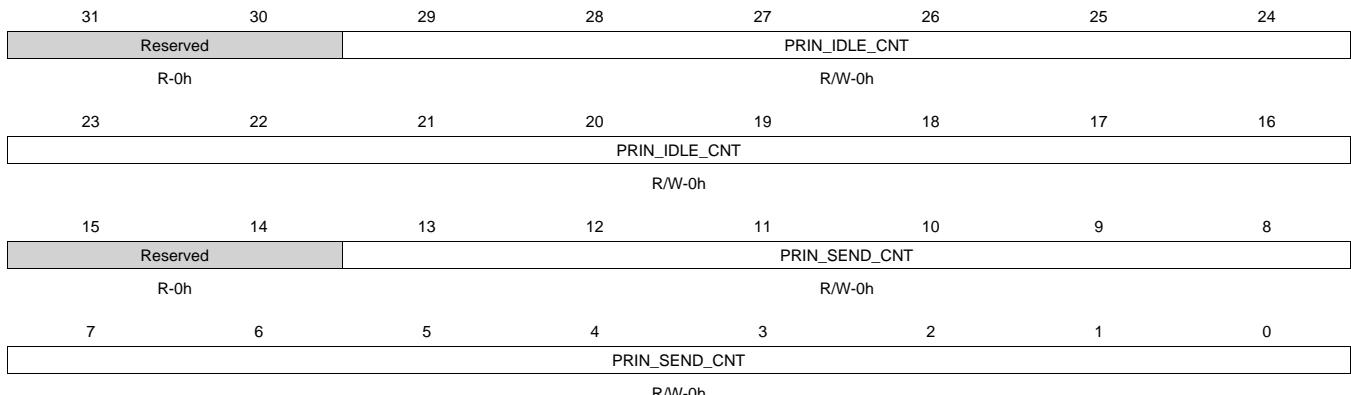
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.14 TX\_PRI2\_RATE Register (offset = 38h) [reset = 0h]

TX\_PRI2\_RATE is shown in [Figure 14-42](#) and described in [Table 14-53](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 2 RATE

**Figure 14-42. TX\_PRI2\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-53. TX\_PRI2\_RATE Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.15 TX\_PRI3\_RATE Register (offset = 3Ch) [reset = 0h]

TX\_PRI3\_RATE is shown in Figure 14-43 and described in Table 14-54.

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 3 RATE

**Figure 14-43. TX\_PRI3\_RATE Register**

| 31       | 30 | 29            | 28 | 27 | 26 | 25 | 24 |
|----------|----|---------------|----|----|----|----|----|
| Reserved |    | PRIN_IDLE_CNT |    |    |    |    |    |
| R-0h     |    | R/W-0h        |    |    |    |    |    |
| 23       | 22 | 21            | 20 | 19 | 18 | 17 | 16 |
|          |    | PRIN_IDLE_CNT |    |    |    |    |    |
|          |    | R/W-0h        |    |    |    |    |    |
| 15       | 14 | 13            | 12 | 11 | 10 | 9  | 8  |
| Reserved |    | PRIN_SEND_CNT |    |    |    |    |    |
| R-0h     |    | R/W-0h        |    |    |    |    |    |
| 7        | 6  | 5             | 4  | 3  | 2  | 1  | 0  |
|          |    | PRIN_SEND_CNT |    |    |    |    |    |
|          |    | R/W-0h        |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-54. TX\_PRI3\_RATE Register Field Descriptions**

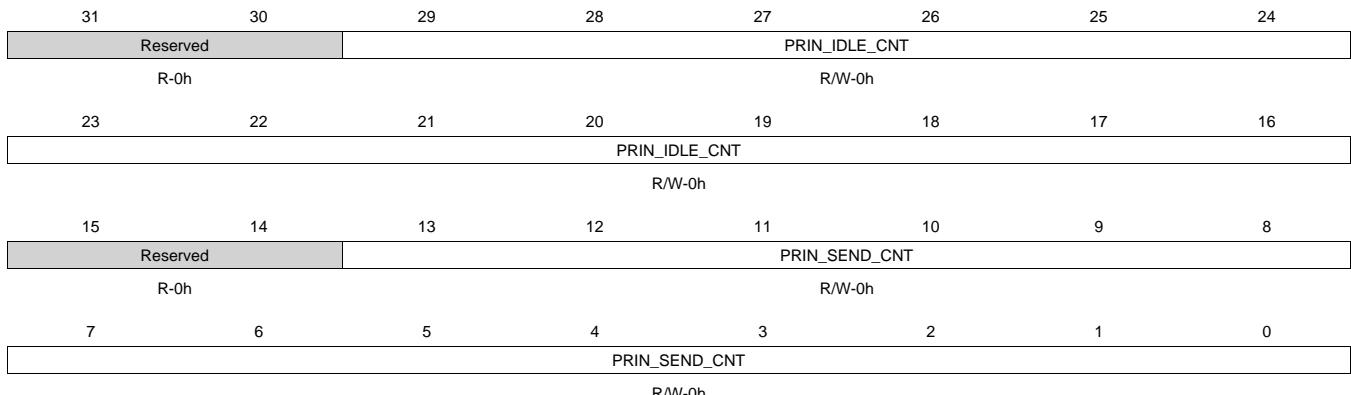
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.16 TX\_PRI4\_RATE Register (offset = 40h) [reset = 0h]

TX\_PRI4\_RATE is shown in [Figure 14-44](#) and described in [Table 14-55](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 4 RATE

**Figure 14-44. TX\_PRI4\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-55. TX\_PRI4\_RATE Register Field Descriptions**

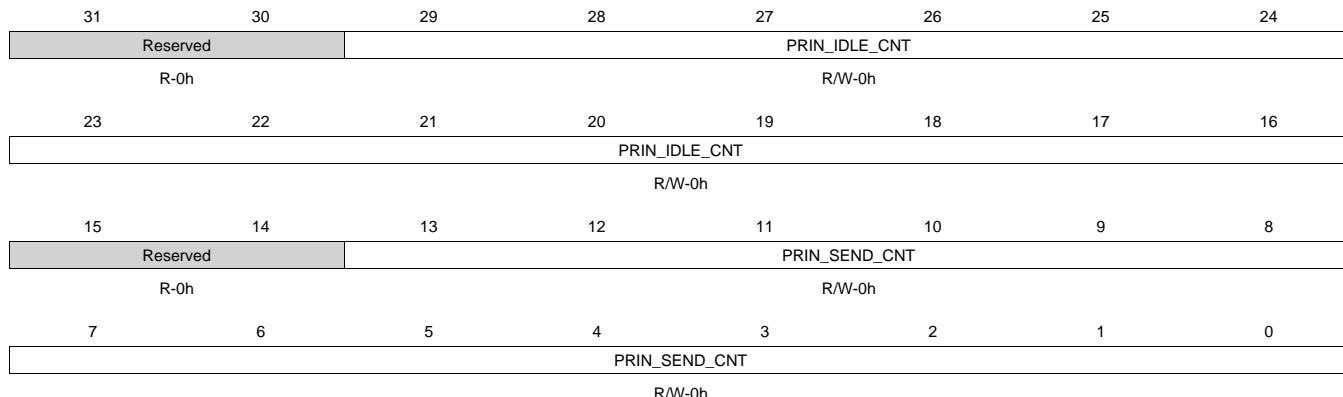
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.17 TX\_PRI5\_RATE Register (offset = 44h) [reset = 0h]

TX\_PRI5\_RATE is shown in [Figure 14-45](#) and described in [Table 14-56](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 5 RATE

**Figure 14-45. TX\_PRI5\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-56. TX\_PRI5\_RATE Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.18 TX\_PRI6\_RATE Register (offset = 48h) [reset = 0h]

TX\_PRI6\_RATE is shown in Figure 14-46 and described in Table 14-57.

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 6 RATE

**Figure 14-46. TX\_PRI6\_RATE Register**

| 31       | 30 | 29            | 28 | 27 | 26 | 25 | 24 |
|----------|----|---------------|----|----|----|----|----|
| Reserved |    | PRIN_IDLE_CNT |    |    |    |    |    |
| R-0h     |    | R/W-0h        |    |    |    |    |    |
| 23       | 22 | 21            | 20 | 19 | 18 | 17 | 16 |
|          |    | PRIN_IDLE_CNT |    |    |    |    |    |
|          |    | R/W-0h        |    |    |    |    |    |
| 15       | 14 | 13            | 12 | 11 | 10 | 9  | 8  |
| Reserved |    | PRIN_SEND_CNT |    |    |    |    |    |
| R-0h     |    | R/W-0h        |    |    |    |    |    |
| 7        | 6  | 5             | 4  | 3  | 2  | 1  | 0  |
|          |    | PRIN_SEND_CNT |    |    |    |    |    |
|          |    | R/W-0h        |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-57. TX\_PRI6\_RATE Register Field Descriptions**

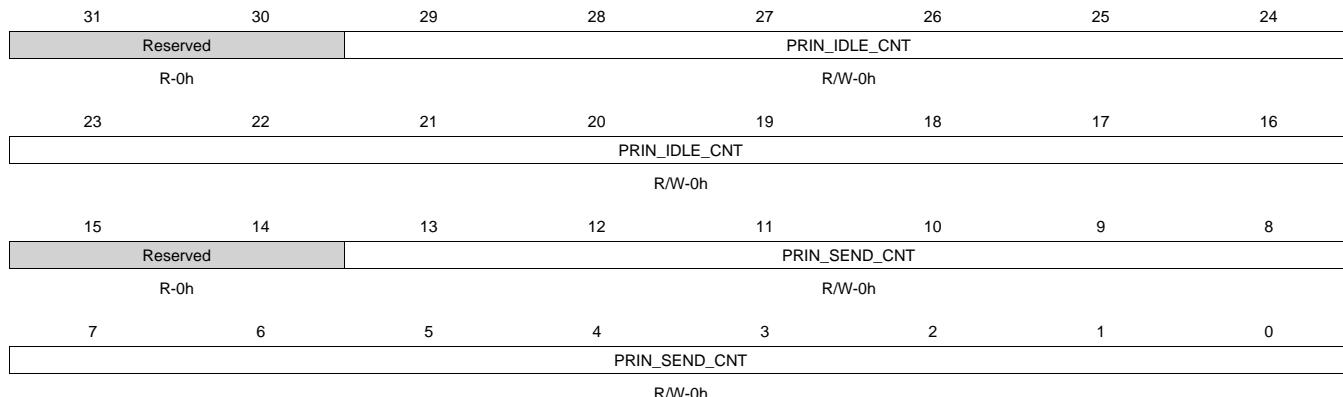
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

#### 14.5.2.19 TX\_PRI7\_RATE Register (offset = 4Ch) [reset = 0h]

TX\_PRI7\_RATE is shown in [Figure 14-47](#) and described in [Table 14-58](#).

CPDMA\_REGS TRANSMIT (INGRESS) PRIORITY 7 RATE

**Figure 14-47. TX\_PRI7\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-58. TX\_PRI7\_RATE Register Field Descriptions**

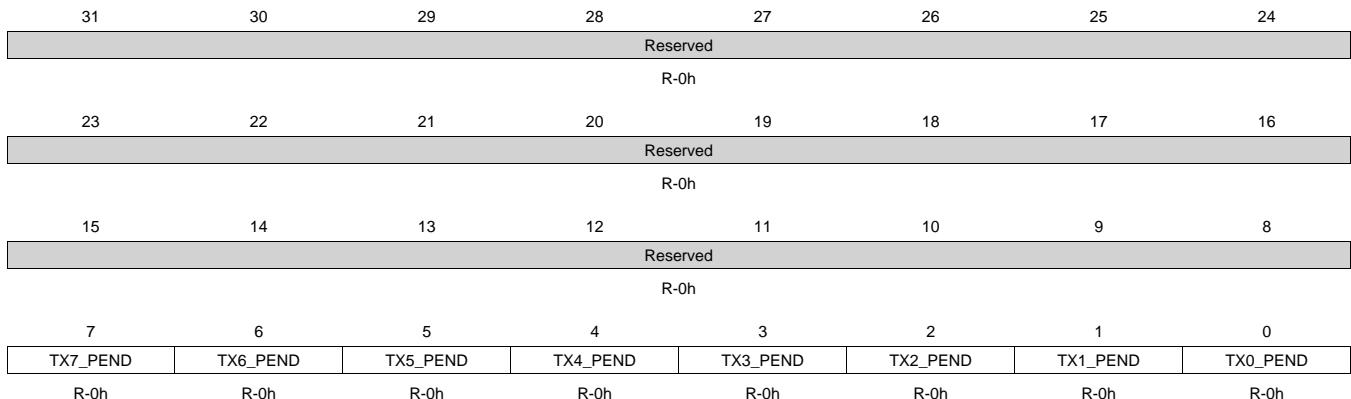
| Bit   | Field         | Type | Reset | Description               |
|-------|---------------|------|-------|---------------------------|
| 31-30 | Reserved      | R    | 0h    |                           |
| 29-16 | PRIN_IDLE_CNT | R/W  | 0h    | Priority (7:0) idle count |
| 15-14 | Reserved      | R    | 0h    |                           |
| 13-0  | PRIN_SEND_CNT | R/W  | 0h    | Priority (7:0) send count |

**14.5.2.20 TX\_INTSTAT\_RAW Register (offset = 80h) [reset = 0h]**

TX\_INTSTAT\_RAW is shown in [Figure 14-48](#) and described in [Table 14-59](#).

CPDMA\_INT TX INTERRUPT STATUS REGISTER (RAW VALUE)

**Figure 14-48. TX\_INTSTAT\_RAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-59. TX\_INTSTAT\_RAW Register Field Descriptions**

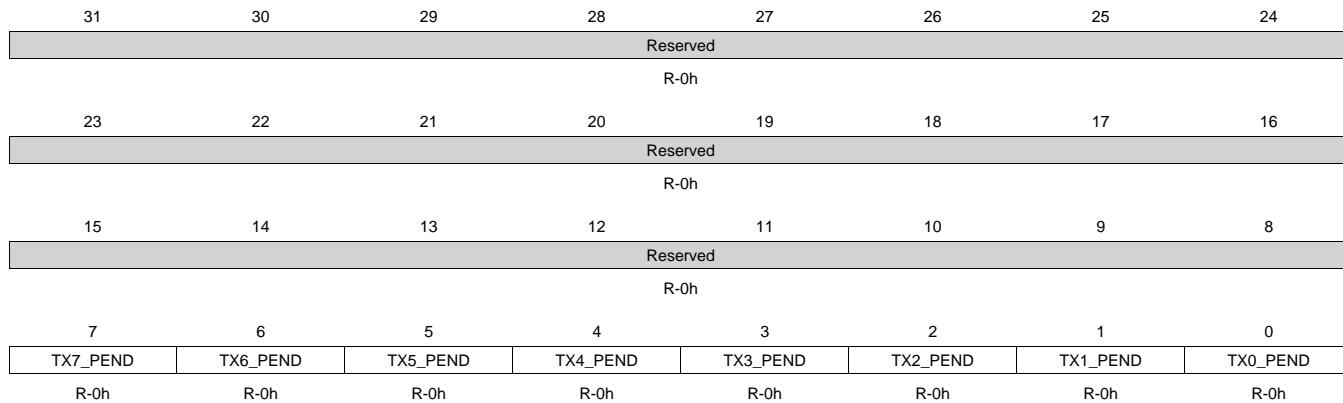
| Bit  | Field    | Type | Reset | Description                          |
|------|----------|------|-------|--------------------------------------|
| 31-8 | Reserved | R    | 0h    |                                      |
| 7    | TX7_PEND | R    | 0h    | TX7_PEND raw int read (before mask). |
| 6    | TX6_PEND | R    | 0h    | TX6_PEND raw int read (before mask). |
| 5    | TX5_PEND | R    | 0h    | TX5_PEND raw int read (before mask). |
| 4    | TX4_PEND | R    | 0h    | TX4_PEND raw int read (before mask). |
| 3    | TX3_PEND | R    | 0h    | TX3_PEND raw int read (before mask). |
| 2    | TX2_PEND | R    | 0h    | TX2_PEND raw int read (before mask). |
| 1    | TX1_PEND | R    | 0h    | TX1_PEND raw int read (before mask). |
| 0    | TX0_PEND | R    | 0h    | TX0_PEND raw int read (before mask). |

#### 14.5.2.21 TX\_INTSTAT\_MASKED Register (offset = 84h) [reset = 0h]

TX\_INTSTAT\_MASKED is shown in [Figure 14-49](#) and described in [Table 14-60](#).

CPDMA\_INT TX INTERRUPT STATUS REGISTER (MASKED VALUE)

**Figure 14-49. TX\_INTSTAT\_MASKED Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-60. TX\_INTSTAT\_MASKED Register Field Descriptions**

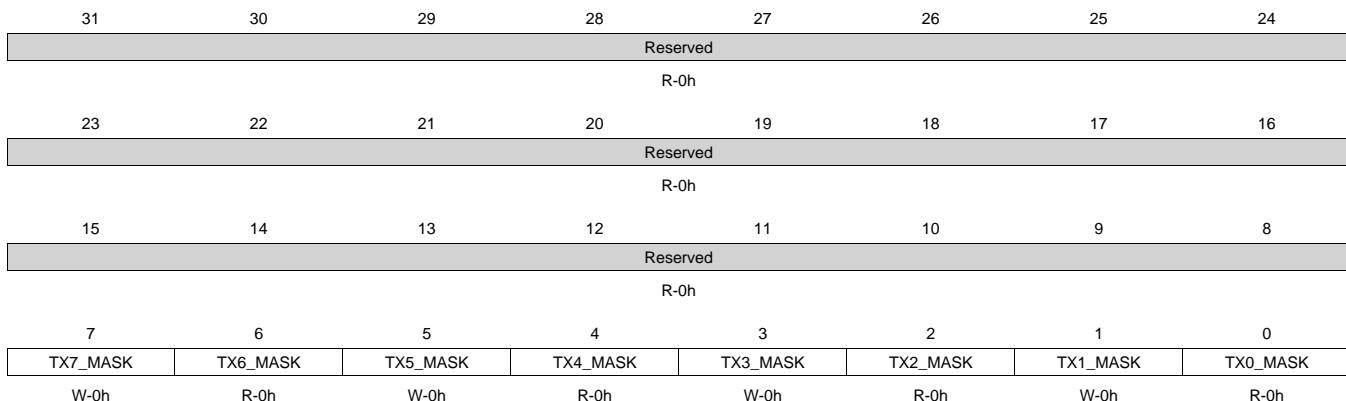
| Bit  | Field    | Type | Reset | Description                     |
|------|----------|------|-------|---------------------------------|
| 31-8 | Reserved | R    | 0h    |                                 |
| 7    | TX7_PEND | R    | 0h    | TX7_PEND masked interrupt read. |
| 6    | TX6_PEND | R    | 0h    | TX6_PEND masked interrupt read. |
| 5    | TX5_PEND | R    | 0h    | TX5_PEND masked interrupt read. |
| 4    | TX4_PEND | R    | 0h    | TX4_PEND masked interrupt read. |
| 3    | TX3_PEND | R    | 0h    | TX3_PEND masked interrupt read. |
| 2    | TX2_PEND | R    | 0h    | TX2_PEND masked interrupt read. |
| 1    | TX1_PEND | R    | 0h    | TX1_PEND masked interrupt read. |
| 0    | TX0_PEND | R    | 0h    | TX0_PEND masked interrupt read. |

#### 14.5.2.22 TX\_INTMASK\_SET Register (offset = 88h) [reset = 0h]

TX\_INTMASK\_SET is shown in [Figure 14-50](#) and described in [Table 14-61](#).

##### CPDMA\_INT TX INTERRUPT MASK SET REGISTER

**Figure 14-50. TX\_INTMASK\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-61. TX\_INTMASK\_SET Register Field Descriptions**

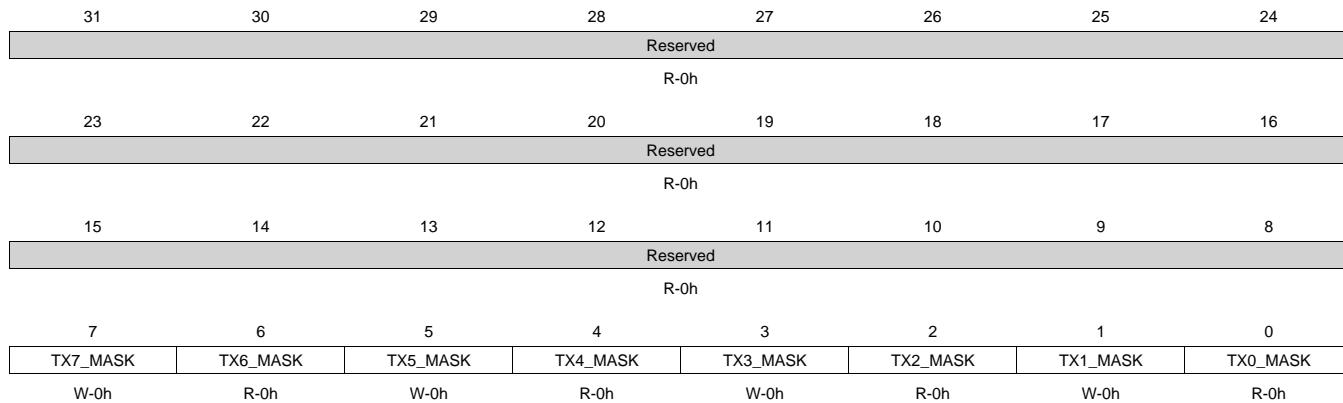
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7    | TX7_MASK | W    | 0h    | TX Channel 7 Mask - Write one to enable interrupt. |
| 6    | TX6_MASK | R    | 0h    | TX Channel 6 Mask - Write one to enable interrupt. |
| 5    | TX5_MASK | W    | 0h    | TX Channel 5 Mask - Write one to enable interrupt. |
| 4    | TX4_MASK | R    | 0h    | TX Channel 4 Mask - Write one to enable interrupt. |
| 3    | TX3_MASK | W    | 0h    | TX Channel 3 Mask - Write one to enable interrupt. |
| 2    | TX2_MASK | R    | 0h    | TX Channel 2 Mask - Write one to enable interrupt. |
| 1    | TX1_MASK | W    | 0h    | TX Channel 1 Mask - Write one to enable interrupt. |
| 0    | TX0_MASK | R    | 0h    | TX Channel 0 Mask - Write one to enable interrupt. |

#### 14.5.2.23 TX\_INTMASK\_CLEAR Register (offset = 8Ch) [reset = 0h]

TX\_INTMASK\_CLEAR is shown in [Figure 14-51](#) and described in [Table 14-62](#).

##### CPDMA\_INT TX INTERRUPT MASK CLEAR REGISTER

**Figure 14-51. TX\_INTMASK\_CLEAR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-62. TX\_INTMASK\_CLEAR Register Field Descriptions**

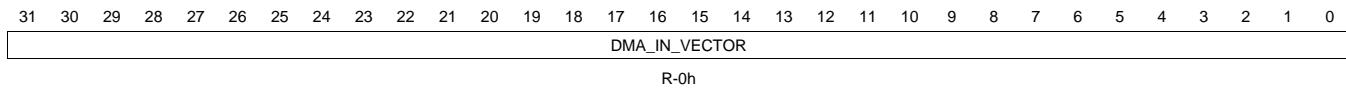
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7    | TX7_MASK | W    | 0h    | TX Channel 7 Mask - Write one to disable interrupt. |
| 6    | TX6_MASK | R    | 0h    | TX Channel 6 Mask - Write one to disable interrupt. |
| 5    | TX5_MASK | W    | 0h    | TX Channel 5 Mask - Write one to disable interrupt. |
| 4    | TX4_MASK | R    | 0h    | TX Channel 4 Mask - Write one to disable interrupt. |
| 3    | TX3_MASK | W    | 0h    | TX Channel 3 Mask - Write one to disable interrupt. |
| 2    | TX2_MASK | R    | 0h    | TX Channel 2 Mask - Write one to disable interrupt. |
| 1    | TX1_MASK | W    | 0h    | TX Channel 1 Mask - Write one to disable interrupt. |
| 0    | TX0_MASK | R    | 0h    | TX Channel 0 Mask - Write one to disable interrupt. |

#### 14.5.2.24 CPDMA\_IN\_VECTOR Register (offset = 90h) [reset = 0h]

CPDMA\_IN\_VECTOR is shown in [Figure 14-52](#) and described in [Table 14-63](#).

CPDMA\_INT INPUT VECTOR (READ ONLY)

**Figure 14-52. CPDMA\_IN\_VECTOR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-63. CPDMA\_IN\_VECTOR Register Field Descriptions**

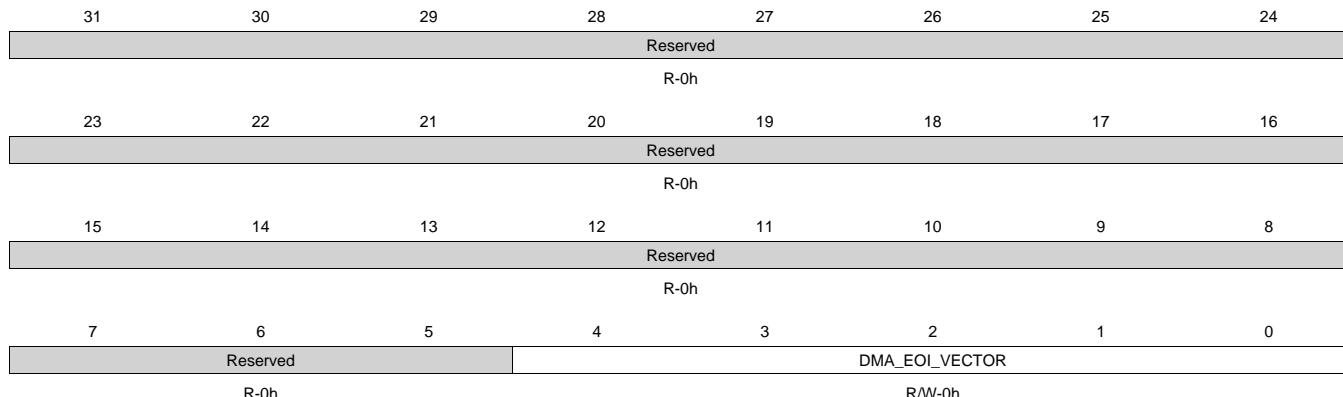
| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-0 | DMA_IN_VECTOR | R    | 0h    | DMA Input Vector - The value of DMA_In_Vector is reset to zero, but will change to the IN_VECTOR bus value one clock after reset is deasserted.<br>Thereafter, this value will change to a new IN_VECTOR value one clock after the IN_VECTOR value changes. |

#### 14.5.2.25 CPDMA\_EOI\_VECTOR Register (offset = 94h) [reset = 0h]

CPDMA\_EOI\_VECTOR is shown in [Figure 14-53](#) and described in [Table 14-64](#).

CPDMA\_INT END OF INTERRUPT VECTOR

**Figure 14-53. CPDMA\_EOI\_VECTOR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-64. CPDMA\_EOI\_VECTOR Register Field Descriptions**

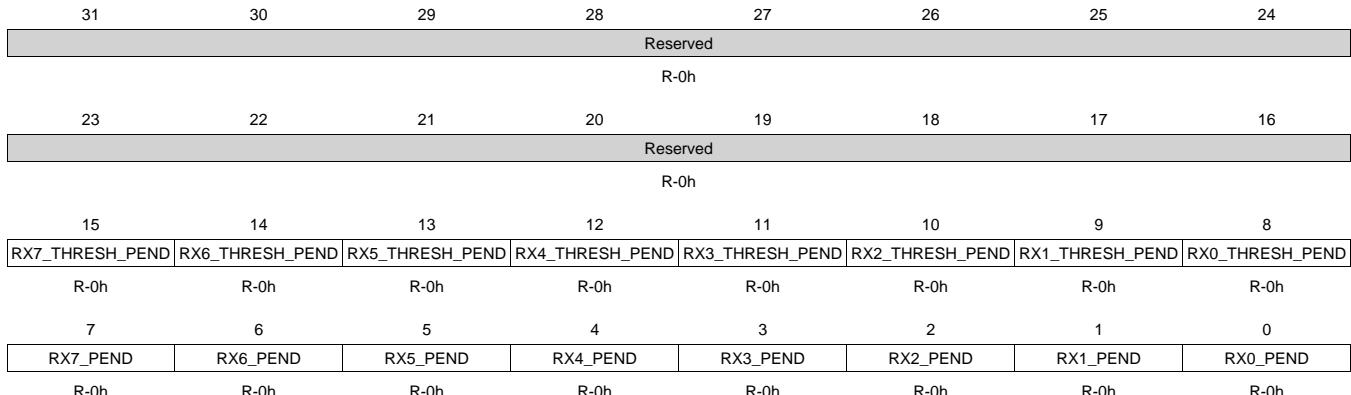
| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-5 | Reserved       | R    | 0h    |   |
| 4-0  | DMA_EOI_VECTOR | R/W  | 0h    | DMA End of Interrupt Vector - The EOI_VECTOR(4:0) pins reflect the value written to this location one CLK cycle after a write to this location.<br>The EOI_WR signal is asserted for a single clock cycle after a latency of two CLK cycles when a write is performed to this location. |

#### 14.5.2.26 RX\_INTSTAT\_RAW Register (offset = A0h) [reset = 0h]

RX\_INTSTAT\_RAW is shown in [Figure 14-54](#) and described in [Table 14-65](#).

CPDMA\_INT RX INTERRUPT STATUS REGISTER (RAW VALUE)

**Figure 14-54. RX\_INTSTAT\_RAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

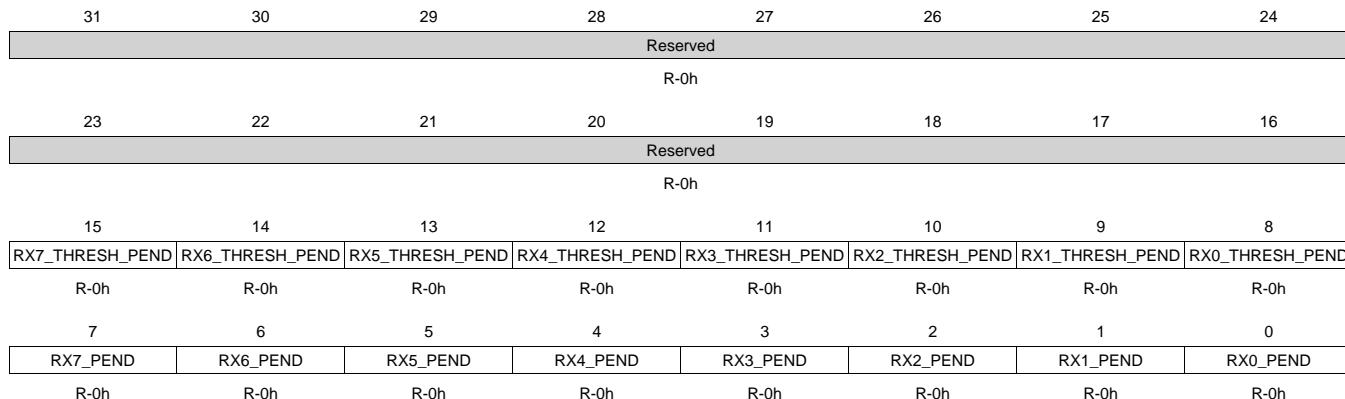
**Table 14-65. RX\_INTSTAT\_RAW Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description                                 |
|-------|-----------------|------|-------|---|
| 31-16 | Reserved        | R    | 0h    |   |
| 15    | RX7_THRESH_PEND | R    | 0h    | RX7_THRESH_PEND raw int read (before mask). |
| 14    | RX6_THRESH_PEND | R    | 0h    | RX6_THRESH_PEND raw int read (before mask). |
| 13    | RX5_THRESH_PEND | R    | 0h    | RX5_THRESH_PEND raw int read (before mask). |
| 12    | RX4_THRESH_PEND | R    | 0h    | RX4_THRESH_PEND raw int read (before mask). |
| 11    | RX3_THRESH_PEND | R    | 0h    | RX3_THRESH_PEND raw int read (before mask). |
| 10    | RX2_THRESH_PEND | R    | 0h    | RX2_THRESH_PEND raw int read (before mask). |
| 9     | RX1_THRESH_PEND | R    | 0h    | RX1_THRESH_PEND raw int read (before mask). |
| 8     | RX0_THRESH_PEND | R    | 0h    | RX0_THRESH_PEND raw int read (before mask). |
| 7     | RX7_PEND        | R    | 0h    | RX7_PEND raw int read (before mask).        |
| 6     | RX6_PEND        | R    | 0h    | RX6_PEND raw int read (before mask).        |
| 5     | RX5_PEND        | R    | 0h    | RX5_PEND raw int read (before mask).        |
| 4     | RX4_PEND        | R    | 0h    | RX4_PEND raw int read (before mask).        |
| 3     | RX3_PEND        | R    | 0h    | RX3_PEND raw int read (before mask).        |
| 2     | RX2_PEND        | R    | 0h    | RX2_PEND raw int read (before mask).        |
| 1     | RX1_PEND        | R    | 0h    | RX1_PEND raw int read (before mask).        |
| 0     | RX0_PEND        | R    | 0h    | RX0_PEND raw int read (before mask).        |

**14.5.2.27 RX\_INTSTAT\_MASKED Register (offset = A4h) [reset = 0h]**

 RX\_INTSTAT\_MASKED is shown in [Figure 14-55](#) and described in [Table 14-66](#).

CPDMA\_INT RX INTERRUPT STATUS REGISTER (MASKED VALUE)

**Figure 14-55. RX\_INTSTAT\_MASKED Register**


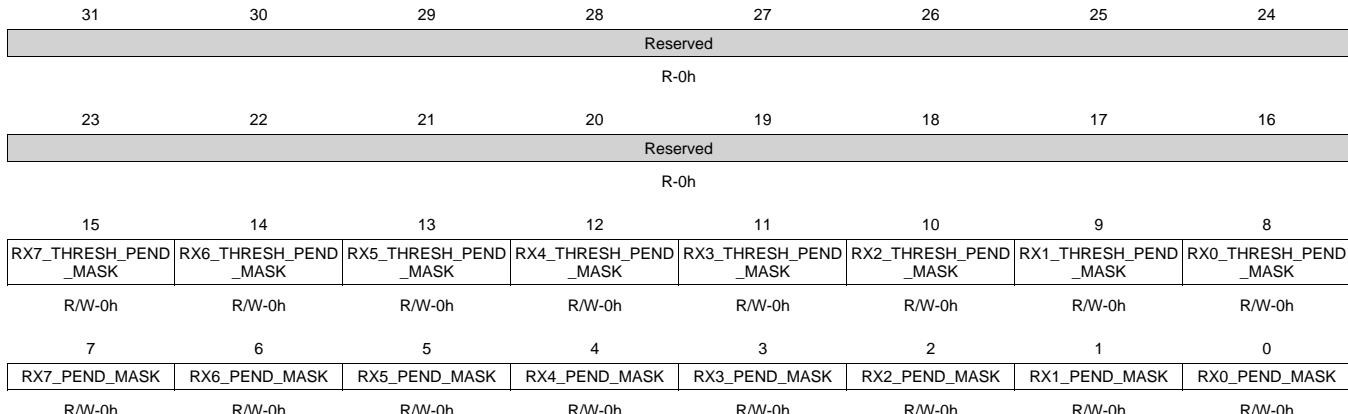
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-66. RX\_INTSTAT\_MASKED Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description                      |
|-------|-----------------|------|-------|----------------------------------|
| 31-16 | Reserved        | R    | 0h    |                                  |
| 15    | RX7_THRESH_PEND | R    | 0h    | RX7_THRESH_PEND masked int read. |
| 14    | RX6_THRESH_PEND | R    | 0h    | RX6_THRESH_PEND masked int read. |
| 13    | RX5_THRESH_PEND | R    | 0h    | RX5_THRESH_PEND masked int read. |
| 12    | RX4_THRESH_PEND | R    | 0h    | RX4_THRESH_PEND masked int read. |
| 11    | RX3_THRESH_PEND | R    | 0h    | RX3_THRESH_PEND masked int read. |
| 10    | RX2_THRESH_PEND | R    | 0h    | RX2_THRESH_PEND masked int read. |
| 9     | RX1_THRESH_PEND | R    | 0h    | RX1_THRESH_PEND masked int read. |
| 8     | RX0_THRESH_PEND | R    | 0h    | RX0_THRESH_PEND masked int read. |
| 7     | RX7_PEND        | R    | 0h    | RX7_PEND masked int read.        |
| 6     | RX6_PEND        | R    | 0h    | RX6_PEND masked int read.        |
| 5     | RX5_PEND        | R    | 0h    | RX5_PEND masked int read.        |
| 4     | RX4_PEND        | R    | 0h    | RX4_PEND masked int read.        |
| 3     | RX3_PEND        | R    | 0h    | RX3_PEND masked int read.        |
| 2     | RX2_PEND        | R    | 0h    | RX2_PEND masked int read.        |
| 1     | RX1_PEND        | R    | 0h    | RX1_PEND masked int read.        |
| 0     | RX0_PEND        | R    | 0h    | RX0_PEND masked int read.        |

**14.5.2.28 RX\_INTMASK\_SET Register (offset = A8h) [reset = 0h]**

RX\_INTMASK\_SET is shown in [Figure 14-56](#) and described in [Table 14-67](#).

**CPDMA\_INT RX INTERRUPT MASK SET REGISTER**
**Figure 14-56. RX\_INTMASK\_SET Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-67. RX\_INTMASK\_SET Register Field Descriptions**

| Bit   | Field                | Type | Reset | Description   |
|-------|----------------------|------|-------|---|
| 31-16 | Reserved             | R    | 0h    |   |
| 15    | RX7_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 7 Threshold Pending Int. Mask - Write one to enable Int. |
| 14    | RX6_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 6 Threshold Pending Int. Mask - Write one to enable Int. |
| 13    | RX5_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 5 Threshold Pending Int. Mask - Write one to enable Int. |
| 12    | RX4_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 4 Threshold Pending Int. Mask - Write one to enable Int. |
| 11    | RX3_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 3 Threshold Pending Int. Mask - Write one to enable Int. |
| 10    | RX2_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 2 Threshold Pending Int. Mask - Write one to enable Int. |
| 9     | RX1_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 1 Threshold Pending Int. Mask - Write one to enable Int. |
| 8     | RX0_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 0 Threshold Pending Int. Mask - Write one to enable Int. |
| 7     | RX7_PEND_MASK        | R/W  | 0h    | RX Channel 7 Pending Int. Mask - Write one to enable Int.           |
| 6     | RX6_PEND_MASK        | R/W  | 0h    | RX Channel 6 Pending Int. Mask - Write one to enable Int.           |
| 5     | RX5_PEND_MASK        | R/W  | 0h    | RX Channel 5 Pending Int. Mask - Write one to enable Int.           |
| 4     | RX4_PEND_MASK        | R/W  | 0h    | RX Channel 4 Pending Int. Mask - Write one to enable Int.           |
| 3     | RX3_PEND_MASK        | R/W  | 0h    | RX Channel 3 Pending Int. Mask - Write one to enable Int.           |
| 2     | RX2_PEND_MASK        | R/W  | 0h    | RX Channel 2 Pending Int. Mask - Write one to enable Int.           |
| 1     | RX1_PEND_MASK        | R/W  | 0h    | RX Channel 1 Pending Int. Mask - Write one to enable Int.           |
| 0     | RX0_PEND_MASK        | R/W  | 0h    | RX Channel 0 Pending Int. Mask - Write one to enable Int.           |

**14.5.2.29 RX\_INTMASK\_CLEAR Register (offset = ACh) [reset = 0h]**

 RX\_INTMASK\_CLEAR is shown in [Figure 14-57](#) and described in [Table 14-68](#).

**CPDMA\_INT RX INTERRUPT MASK CLEAR REGISTER**
**Figure 14-57. RX\_INTMASK\_CLEAR Register**

| 31                   | 30                   | 29                   | 28                   | 27                   | 26                   | 25                   | 24                   |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Reserved             |                      |                      |                      |                      |                      |                      |                      |
| R-0h                 |                      |                      |                      |                      |                      |                      |                      |
| 23                   | 22                   | 21                   | 20                   | 19                   | 18                   | 17                   | 16                   |
| Reserved             |                      |                      |                      |                      |                      |                      |                      |
| R-0h                 |                      |                      |                      |                      |                      |                      |                      |
| 15                   | 14                   | 13                   | 12                   | 11                   | 10                   | 9                    | 8                    |
| RX7_THRESH_PEND_MASK | RX6_THRESH_PEND_MASK | RX5_THRESH_PEND_MASK | RX4_THRESH_PEND_MASK | RX3_THRESH_PEND_MASK | RX2_THRESH_PEND_MASK | RX1_THRESH_PEND_MASK | RX0_THRESH_PEND_MASK |
| R/W-0h               |
| 7                    | 6                    | 5                    | 4                    | 3                    | 2                    | 1                    | 0                    |
| RX7_PEND_MASK        | RX6_PEND_MASK        | RX5_PEND_MASK        | RX4_PEND_MASK        | RX3_PEND_MASK        | RX2_PEND_MASK        | RX1_PEND_MASK        | RX0_PEND_MASK        |
| R/W-0h               |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-68. RX\_INTMASK\_CLEAR Register Field Descriptions**

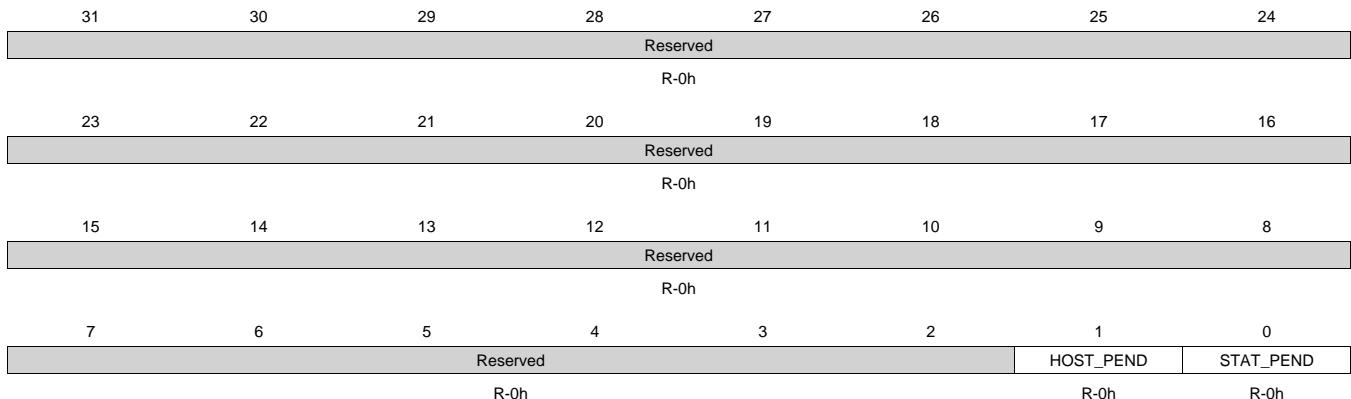
| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31-16 | Reserved             | R    | 0h    |  |
| 15    | RX7_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 7 Threshold Pending Int. Mask - Write one to disable Int. |
| 14    | RX6_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 6 Threshold Pending Int. Mask - Write one to disable Int. |
| 13    | RX5_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 5 Threshold Pending Int. Mask - Write one to disable Int. |
| 12    | RX4_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 4 Threshold Pending Int. Mask - Write one to disable Int. |
| 11    | RX3_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 3 Threshold Pending Int. Mask - Write one to disable Int. |
| 10    | RX2_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 2 Threshold Pending Int. Mask - Write one to disable Int. |
| 9     | RX1_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 1 Threshold Pending Int. Mask - Write one to disable Int. |
| 8     | RX0_THRESH_PEND_MASK | R/W  | 0h    | RX Channel 0 Threshold Pending Int. Mask - Write one to disable Int. |
| 7     | RX7_PEND_MASK        | R/W  | 0h    | RX Channel 7 Pending Int. Mask - Write one to disable Int.           |
| 6     | RX6_PEND_MASK        | R/W  | 0h    | RX Channel 6 Pending Int. Mask - Write one to disable Int.           |
| 5     | RX5_PEND_MASK        | R/W  | 0h    | RX Channel 5 Pending Int. Mask - Write one to disable Int.           |
| 4     | RX4_PEND_MASK        | R/W  | 0h    | RX Channel 4 Pending Int. Mask - Write one to disable Int.           |
| 3     | RX3_PEND_MASK        | R/W  | 0h    | RX Channel 3 Pending Int. Mask - Write one to disable Int.           |
| 2     | RX2_PEND_MASK        | R/W  | 0h    | RX Channel 2 Pending Int. Mask - Write one to disable Int.           |
| 1     | RX1_PEND_MASK        | R/W  | 0h    | RX Channel 1 Pending Int. Mask - Write one to disable Int.           |
| 0     | RX0_PEND_MASK        | R/W  | 0h    | RX Channel 0 Pending Int. Mask - Write one to disable Int.           |

**14.5.2.30 DMA\_INTSTAT\_RAW Register (offset = B0h) [reset = 0h]**

DMA\_INTSTAT\_RAW is shown in [Figure 14-58](#) and described in [Table 14-69](#).

CPDMA\_INT DMA INTERRUPT STATUS REGISTER (RAW VALUE)

**Figure 14-58. DMA\_INTSTAT\_RAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-69. DMA\_INTSTAT\_RAW Register Field Descriptions**

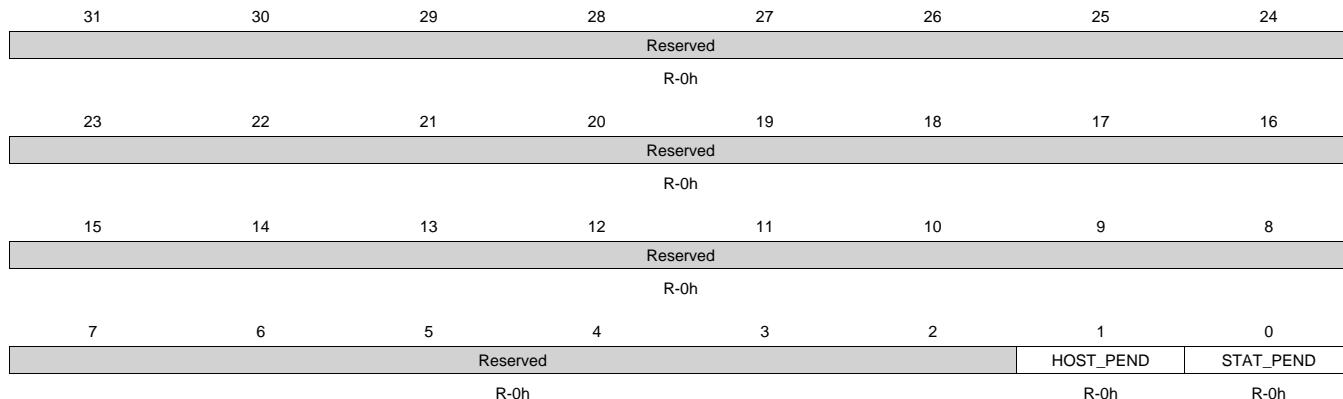
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-2 | Reserved  | R    | 0h    |  |
| 1    | HOST_PEND | R    | 0h    | Host Pending Interrupt - raw int read (before mask).       |
| 0    | STAT_PEND | R    | 0h    | Statistics Pending Interrupt - raw int read (before mask). |

#### 14.5.2.31 DMA\_INTSTAT\_MASKED Register (offset = B4h) [reset = 0h]

DMA\_INTSTAT\_MASKED is shown in [Figure 14-59](#) and described in [Table 14-70](#).

CPDMA\_INT DMA INTERRUPT STATUS REGISTER (MASKED VALUE)

**Figure 14-59. DMA\_INTSTAT\_MASKED Register**



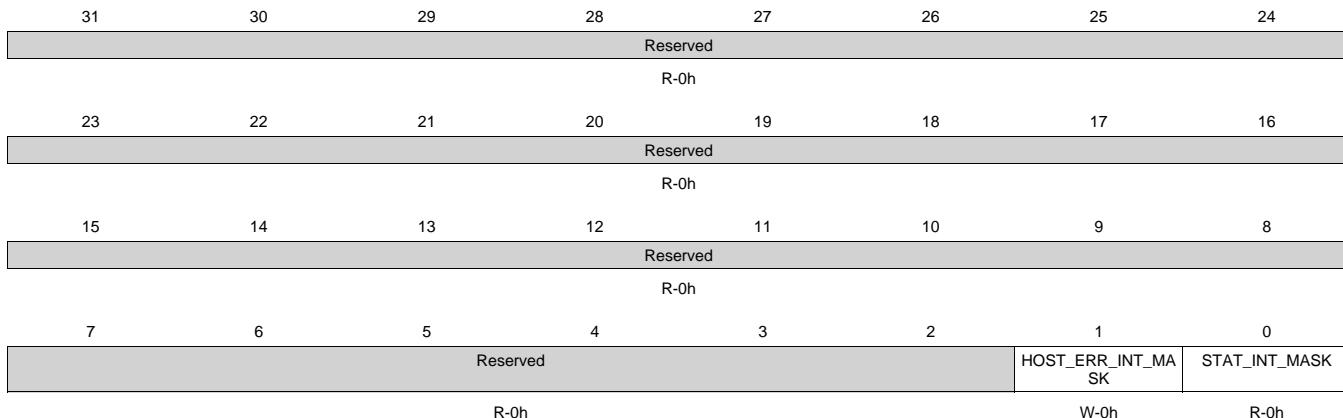
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-70. DMA\_INTSTAT\_MASKED Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 31-2 | Reserved  | R    | 0h    |   |
| 1    | HOST_PEND | R    | 0h    | Host Pending Interrupt - masked interrupt read.       |
| 0    | STAT_PEND | R    | 0h    | Statistics Pending Interrupt - masked interrupt read. |

**14.5.2.32 DMA\_INTMASK\_SET Register (offset = B8h) [reset = 0h]**

DMA\_INTMASK\_SET is shown in [Figure 14-60](#) and described in [Table 14-71](#).

**CPDMA\_INT DMA INTERRUPT MASK SET REGISTER**
**Figure 14-60. DMA\_INTMASK\_SET Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-71. DMA\_INTMASK\_SET Register Field Descriptions**

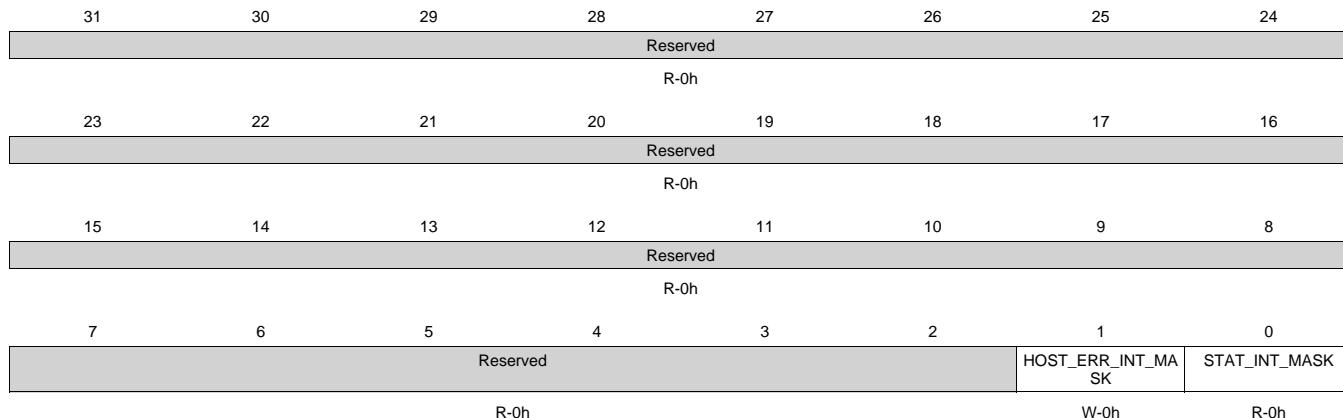
| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 31-2 | Reserved          | R    | 0h    |  |
| 1    | HOST_ERR_INT_MASK | W    | 0h    | Host Error Interrupt Mask - Write one to enable interrupt. |
| 0    | STAT_INT_MASK     | R    | 0h    | Statistics Interrupt Mask - Write one to enable interrupt. |

#### 14.5.2.33 DMA\_INTMASK\_CLEAR Register (offset = BCh) [reset = 0h]

DMA\_INTMASK\_CLEAR is shown in [Figure 14-61](#) and described in [Table 14-72](#).

##### CPDMA\_INT DMA INTERRUPT MASK CLEAR REGISTER

**Figure 14-61. DMA\_INTMASK\_CLEAR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-72. DMA\_INTMASK\_CLEAR Register Field Descriptions**

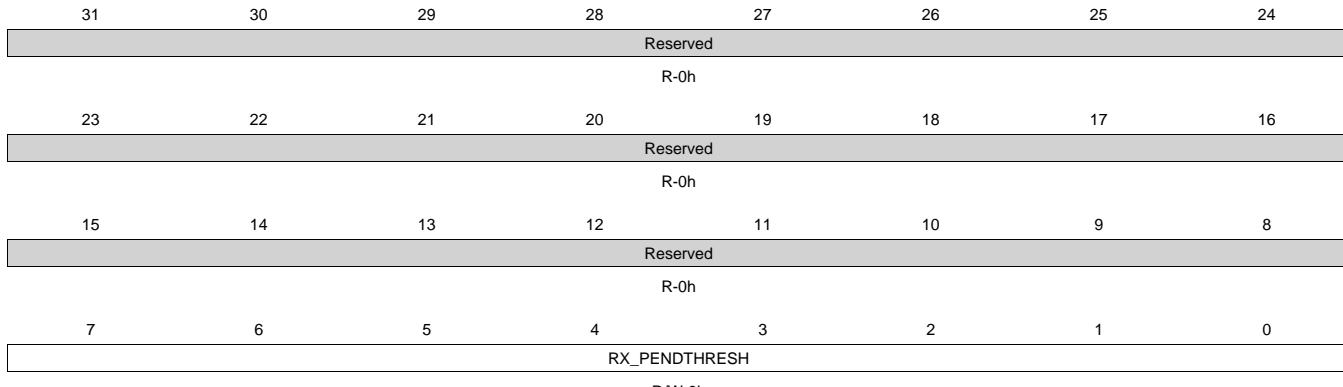
| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-2 | Reserved          | R    | 0h    |   |
| 1    | HOST_ERR_INT_MASK | W    | 0h    | Host Error Interrupt Mask - Write one to disable interrupt. |
| 0    | STAT_INT_MASK     | R    | 0h    | Statistics Interrupt Mask - Write one to disable interrupt. |

#### 14.5.2.34 RX0\_PENDTHRESH Register (offset = C0h) [reset = 0h]

RX0\_PENDTHRESH is shown in [Figure 14-62](#) and described in [Table 14-73](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 0

**Figure 14-62. RX0\_PENDTHRESH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

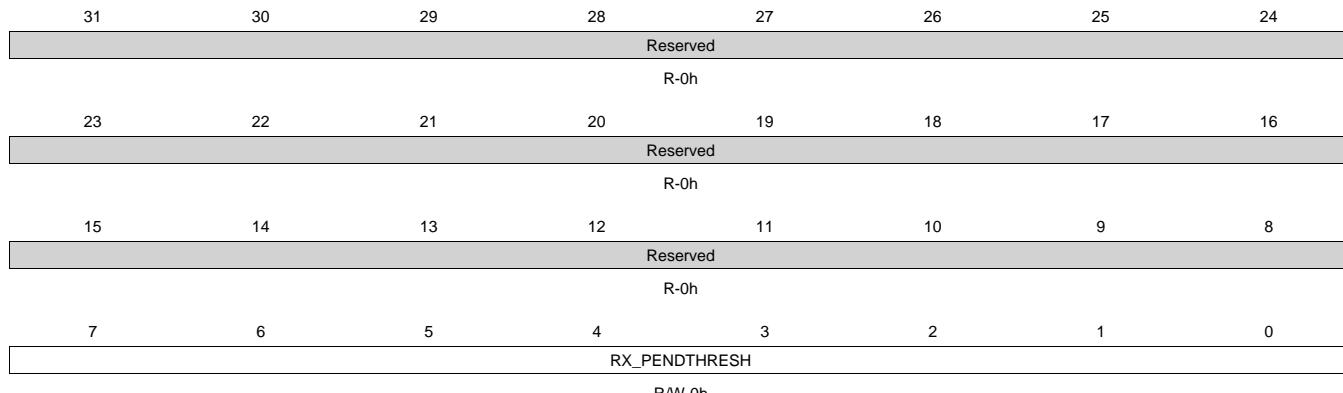
**Table 14-73. RX0\_PENDTHRESH Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.35 RX1\_PENDTHRESH Register (offset = C4h) [reset = 0h]**

 RX1\_PENDTHRESH is shown in [Figure 14-63](#) and described in [Table 14-74](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 1

**Figure 14-63. RX1\_PENDTHRESH Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-74. RX1\_PENDTHRESH Register Field Descriptions**

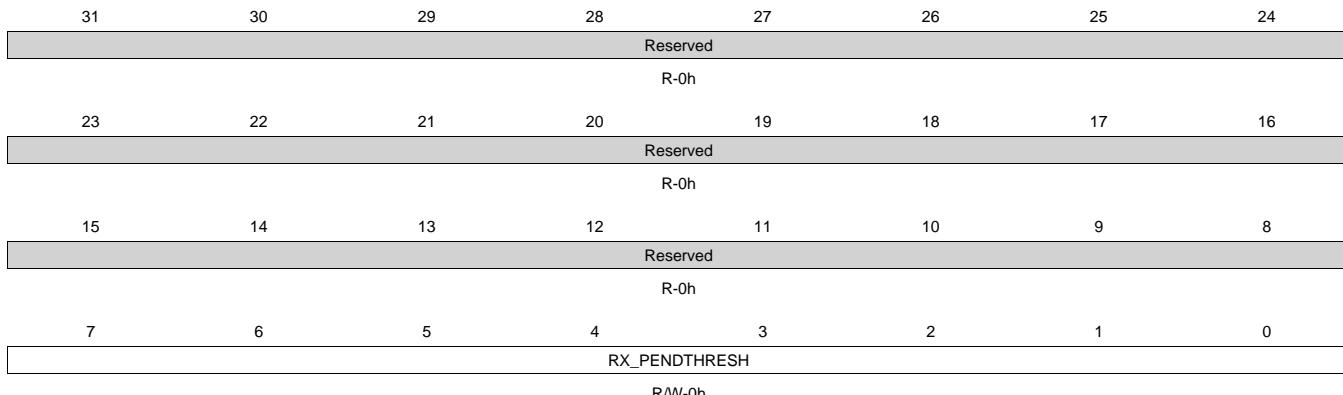
| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.36 RX2\_PENDTHRESH Register (offset = C8h) [reset = 0h]**

RX2\_PENDTHRESH is shown in [Figure 14-64](#) and described in [Table 14-75](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 2

**Figure 14-64. RX2\_PENDTHRESH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

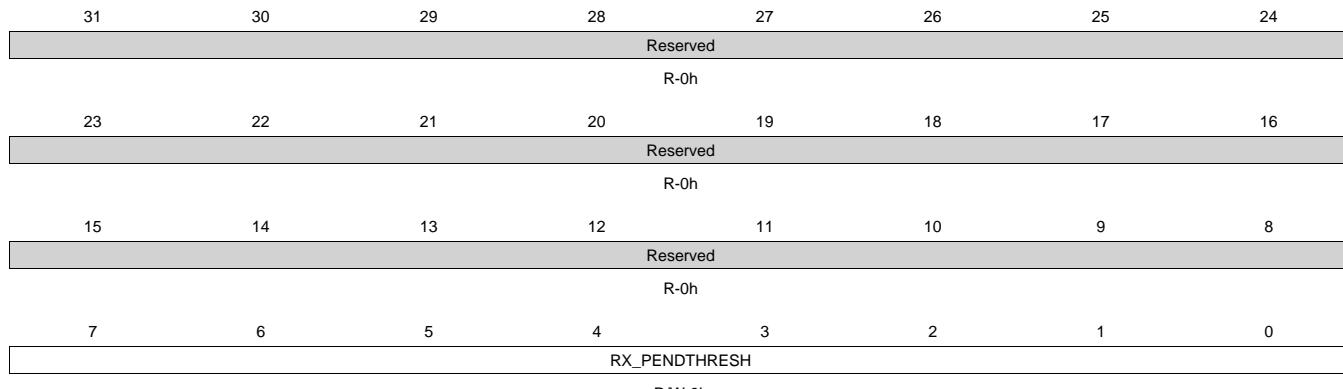
**Table 14-75. RX2\_PENDTHRESH Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.37 RX3\_PENDTHRESH Register (offset = CCh) [reset = 0h]**

 RX3\_PENDTHRESH is shown in [Figure 14-65](#) and described in [Table 14-76](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 3

**Figure 14-65. RX3\_PENDTHRESH Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-76. RX3\_PENDTHRESH Register Field Descriptions**

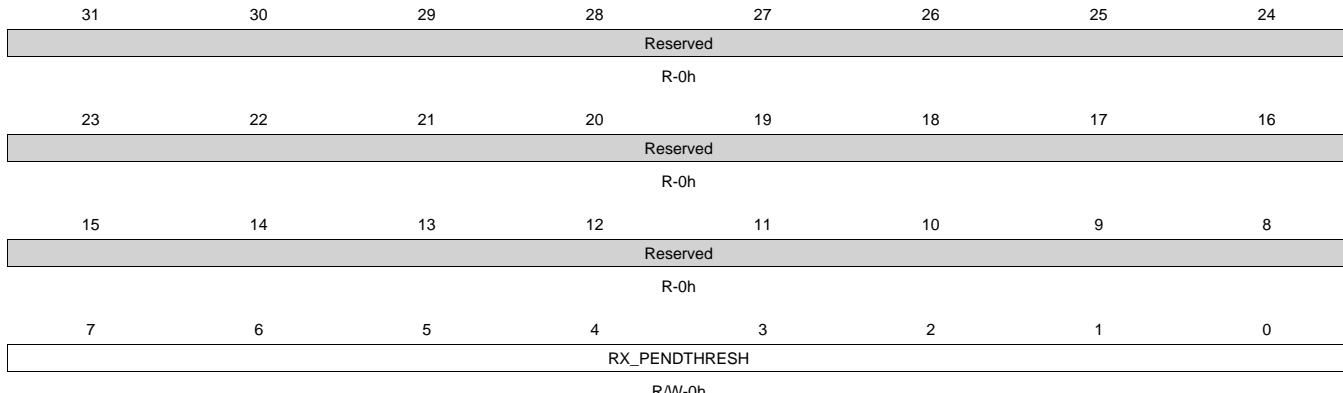
| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.38 RX4\_PENDTHRESH Register (offset = D0h) [reset = 0h]**

RX4\_PENDTHRESH is shown in [Figure 14-66](#) and described in [Table 14-77](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 4

**Figure 14-66. RX4\_PENDTHRESH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

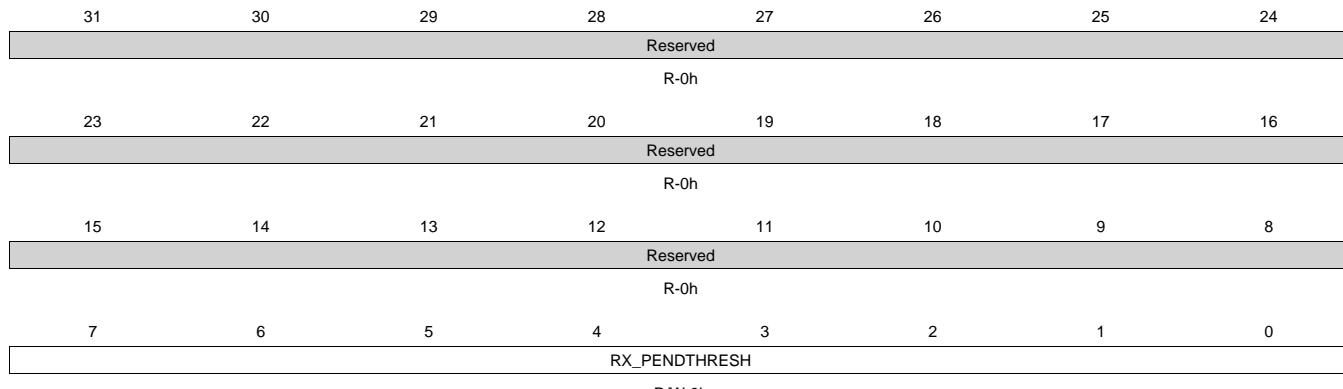
**Table 14-77. RX4\_PENDTHRESH Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.39 RX5\_PENDTHRESH Register (offset = D4h) [reset = 0h]**

 RX5\_PENDTHRESH is shown in [Figure 14-67](#) and described in [Table 14-78](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 5

**Figure 14-67. RX5\_PENDTHRESH Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-78. RX5\_PENDTHRESH Register Field Descriptions**

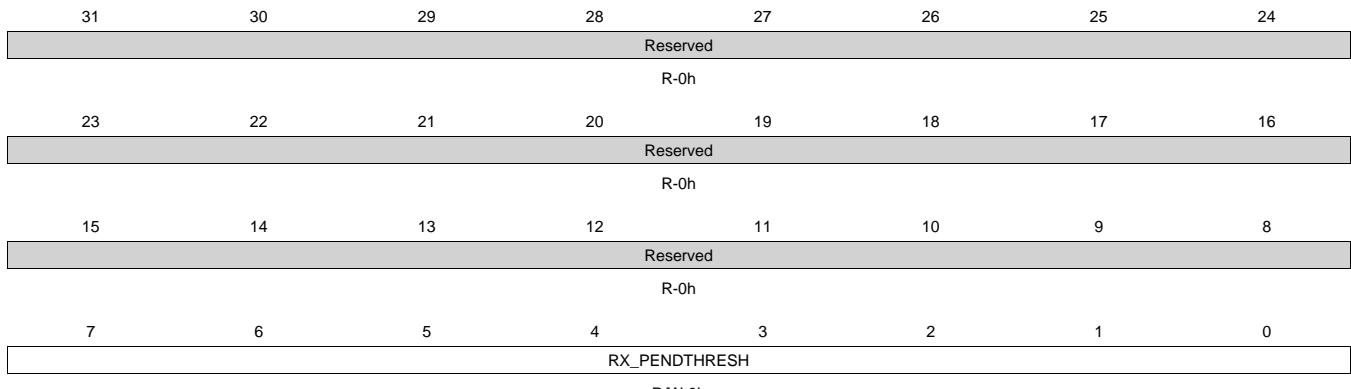
| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

**14.5.2.40 RX6\_PENDTHRESH Register (offset = D8h) [reset = 0h]**

RX6\_PENDTHRESH is shown in [Figure 14-68](#) and described in [Table 14-79](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 6

**Figure 14-68. RX6\_PENDTHRESH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-79. RX6\_PENDTHRESH Register Field Descriptions**

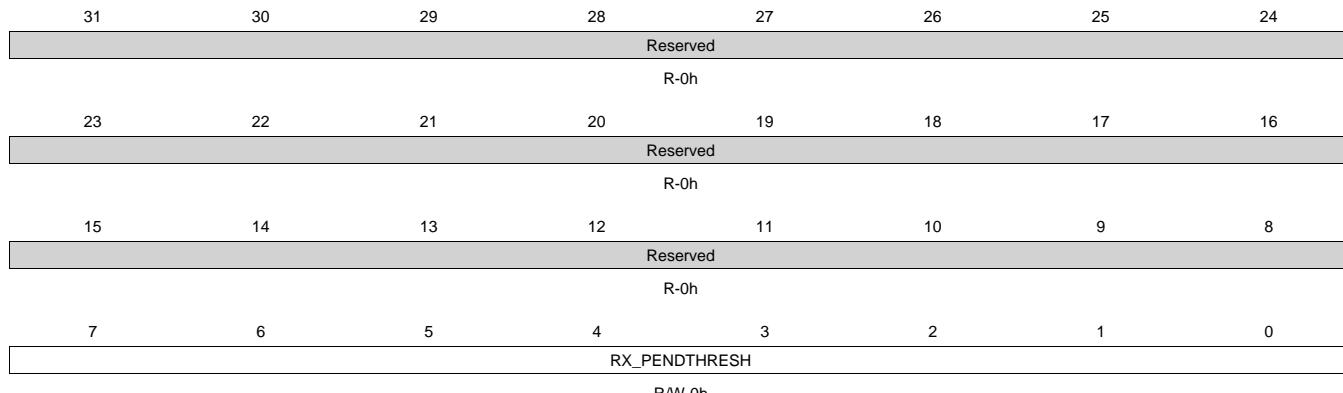
| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

#### 14.5.2.41 RX7\_PENDTHRESH Register (offset = DCh) [reset = 0h]

RX7\_PENDTHRESH is shown in [Figure 14-69](#) and described in [Table 14-80](#).

CPDMA\_INT RECEIVE THRESHOLD PENDING REGISTER CHANNEL 7

**Figure 14-69. RX7\_PENDTHRESH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-80. RX7\_PENDTHRESH Register Field Descriptions**

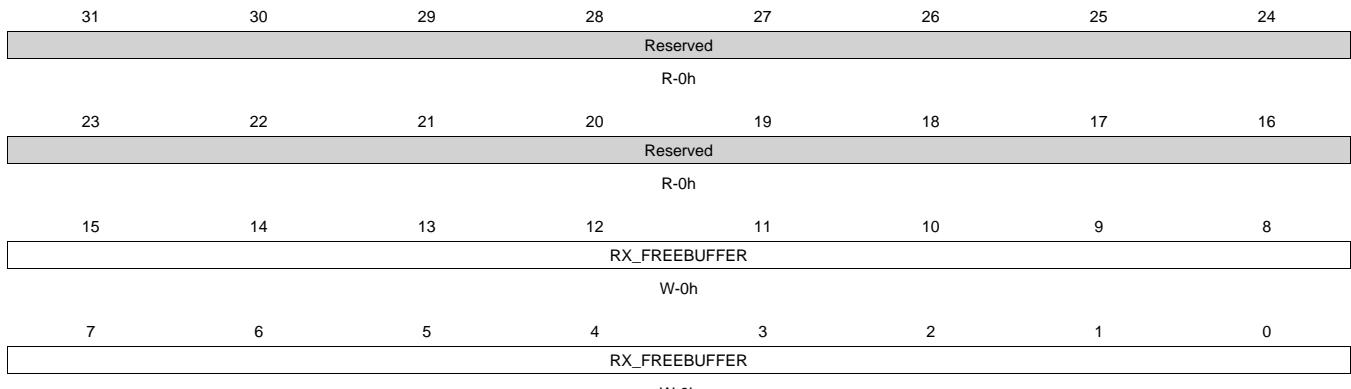
| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-8 | Reserved      | R    | 0h    |  |
| 7-0  | RX_PENDTHRESH | R/W  | 0h    | Rx Flow Threshold - This field contains the threshold value for issuing receive threshold pending interrupts (when enabled). |

#### 14.5.2.42 RX0\_FREEBUFFER Register (offset = E0h) [reset = 0h]

RX0\_FREEBUFFER is shown in [Figure 14-70](#) and described in [Table 14-81](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 0

**Figure 14-70. RX0\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-81. RX0\_FREEBUFFER Register Field Descriptions**

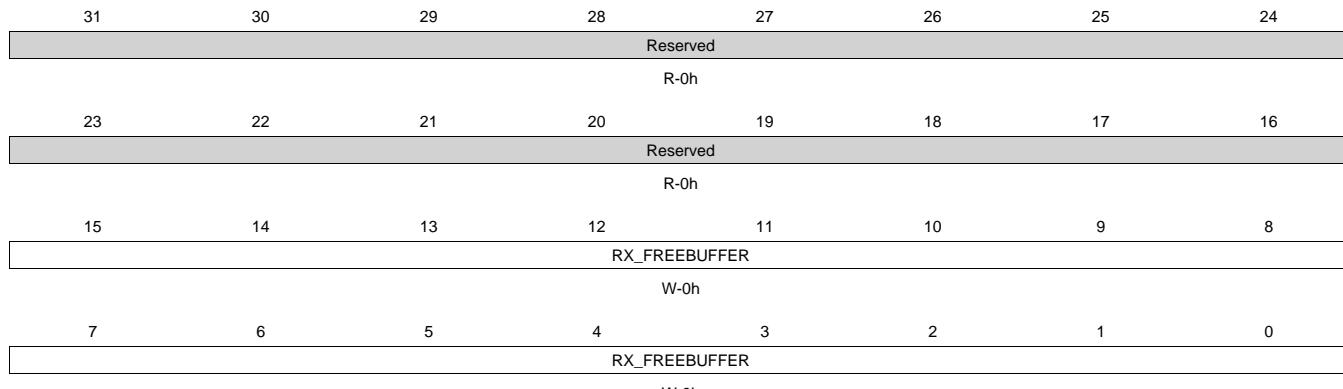
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.43 RX1\_FREEBUFFER Register (offset = E4h) [reset = 0h]

RX1\_FREEBUFFER is shown in [Figure 14-71](#) and described in [Table 14-82](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 1

**Figure 14-71. RX1\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-82. RX1\_FREEBUFFER Register Field Descriptions**

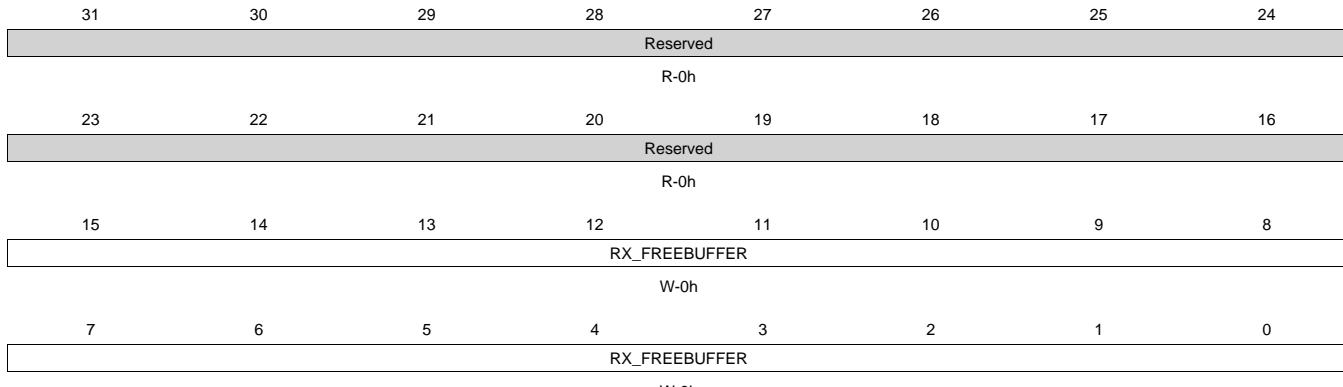
| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31-16 | Reserved      | R    | 0h    |  |
| 15-0  | RX_FREEBUFFER | W    | 0h    | Rx Free Buffer Count - This field contains the count of free buffers available.<br>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).<br>This is a write to increment field.<br>This field rolls over to zero on overflow.<br>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).<br>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.<br>This is a write to increment field.<br>The host must write this field with the number of buffers that have been freed due to host processing. |

#### 14.5.2.44 RX2\_FREEBUFFER Register (offset = E8h) [reset = 0h]

RX2\_FREEBUFFER is shown in [Figure 14-72](#) and described in [Table 14-83](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 2

**Figure 14-72. RX2\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-83. RX2\_FREEBUFFER Register Field Descriptions**

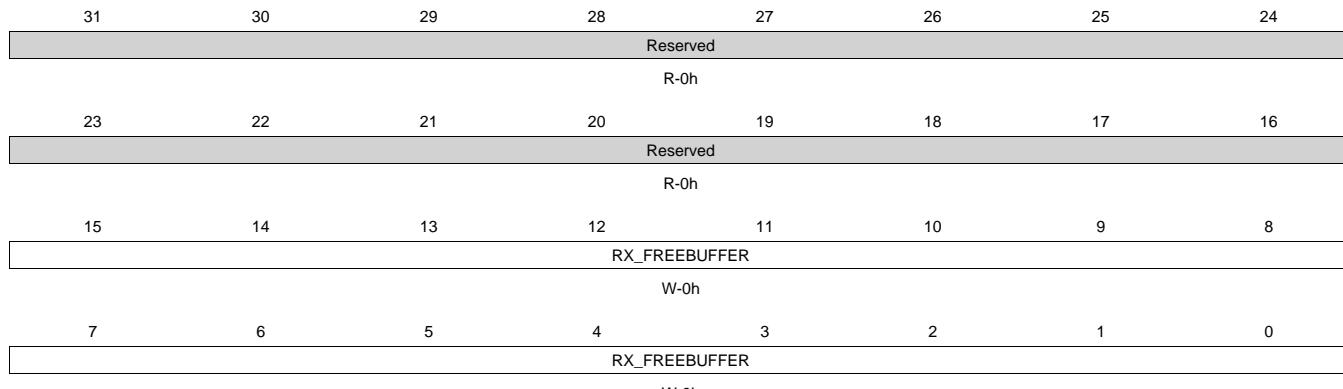
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.45 RX3\_FREEBUFFER Register (offset = ECh) [reset = 0h]

RX3\_FREEBUFFER is shown in [Figure 14-73](#) and described in [Table 14-84](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 3

**Figure 14-73. RX3\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-84. RX3\_FREEBUFFER Register Field Descriptions**

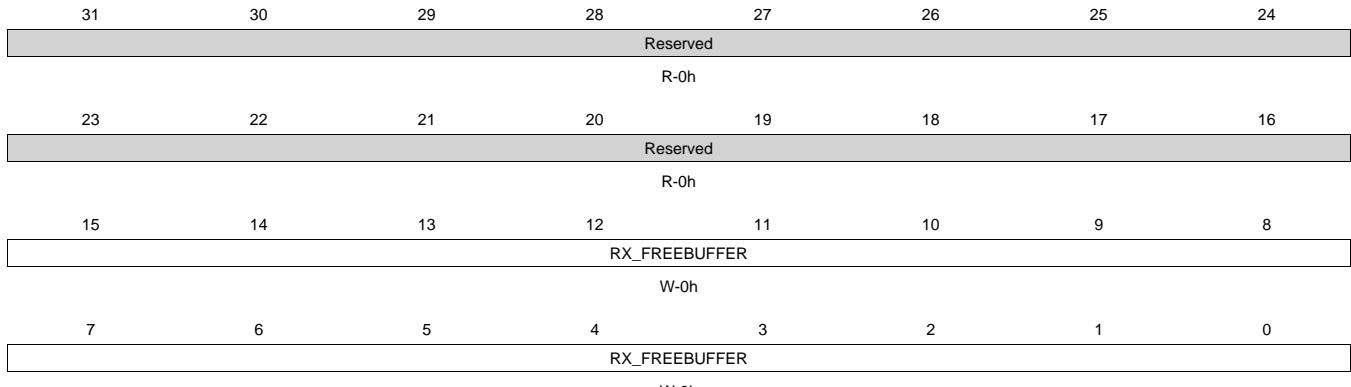
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.46 RX4\_FREEBUFFER Register (offset = F0h) [reset = 0h]

RX4\_FREEBUFFER is shown in [Figure 14-74](#) and described in [Table 14-85](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 4

**Figure 14-74. RX4\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-85. RX4\_FREEBUFFER Register Field Descriptions**

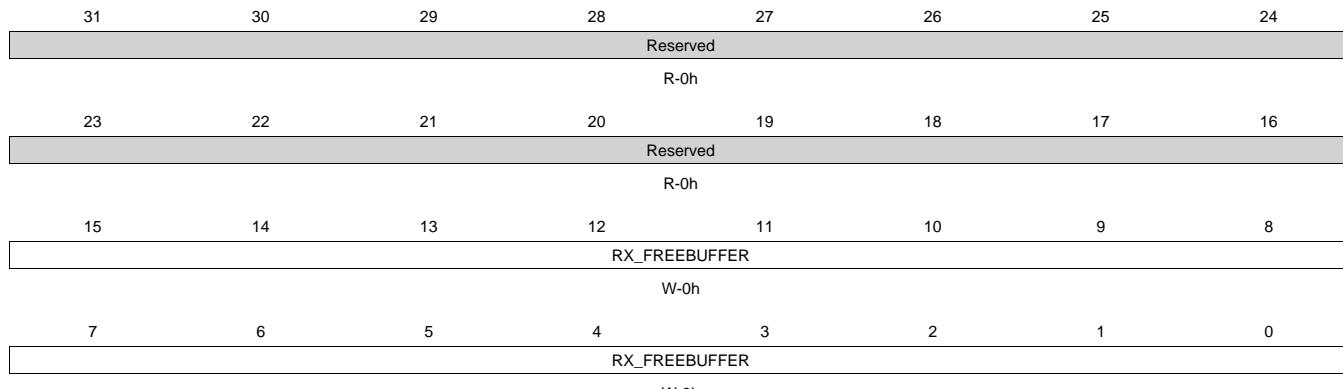
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.47 RX5\_FREEBUFFER Register (offset = F4h) [reset = 0h]

RX5\_FREEBUFFER is shown in [Figure 14-75](#) and described in [Table 14-86](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 5

**Figure 14-75. RX5\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-86. RX5\_FREEBUFFER Register Field Descriptions**

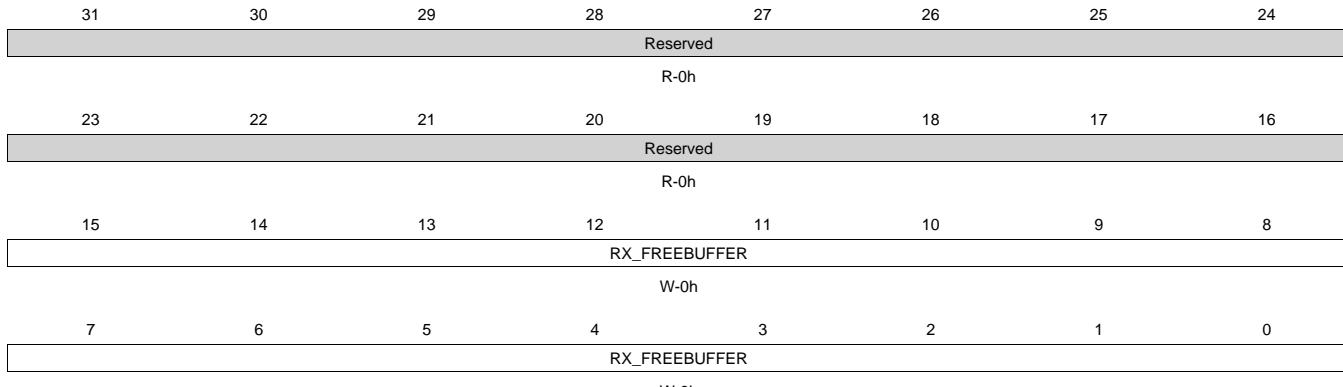
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.48 RX6\_FREEBUFFER Register (offset = F8h) [reset = 0h]

RX6\_FREEBUFFER is shown in [Figure 14-76](#) and described in [Table 14-87](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 6

**Figure 14-76. RX6\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-87. RX6\_FREEBUFFER Register Field Descriptions**

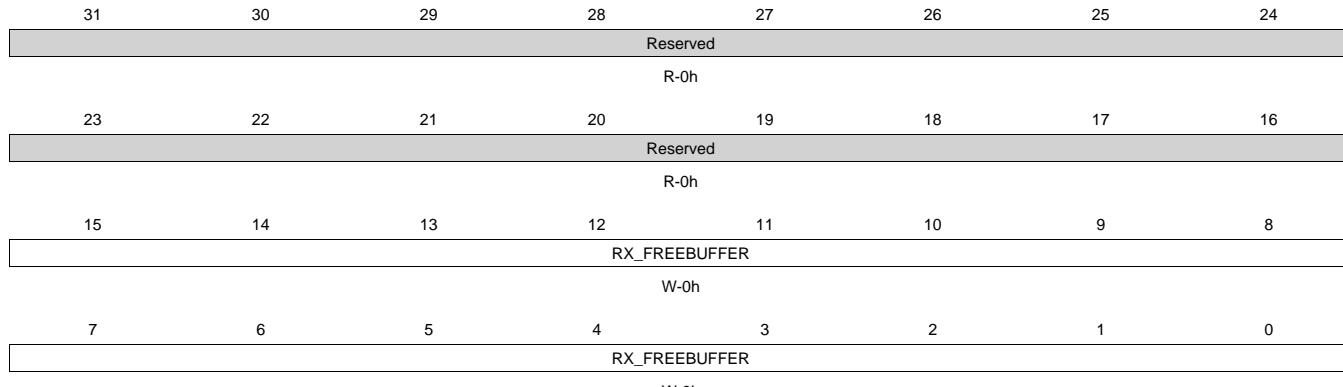
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.2.49 RX7\_FREEBUFFER Register (offset = FCh) [reset = 0h]

RX7\_FREEBUFFER is shown in [Figure 14-77](#) and described in [Table 14-88](#).

CPDMA\_INT RECEIVE FREE BUFFER REGISTER CHANNEL 7

**Figure 14-77. RX7\_FREEBUFFER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-88. RX7\_FREEBUFFER Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-16 | Reserved      | R    | 0h    |   |
| 15-0  | RX_FREEBUFFER | W    | 0h    | <p>Rx Free Buffer Count - This field contains the count of free buffers available.</p> <p>The rx_pendthresh value is compared with this field to determine if the receive threshold pending interrupt should be asserted (if enabled).</p> <p>This is a write to increment field.</p> <p>This field rolls over to zero on overflow.</p> <p>If receive threshold pending interrupts are used, the host must initialize this field to the number of available buffers (one register per channel).</p> <p>The port decrements (by the number of buffers in the received frame) the associated channel register for each received frame.</p> <p>This is a write to increment field.</p> <p>The host must write this field with the number of buffers that have been freed due to host processing.</p> |

#### 14.5.3 CPSW\_CPTS Registers

[Table 14-89](#) lists the memory-mapped registers for the CPSW\_CPTS. All register offset addresses not listed in [Table 14-89](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-89. CPSW\_CPTS REGISTERS**

| Offset | Acronym             | Register Name                              | Section                          |
|--------|---------------------|--|----------------------------------|
| 0h     | CPTS_IDVER          | IDENTIFICATION AND VERSION REGISTER        | <a href="#">Section 14.5.3.1</a> |
| 4h     | CPTS_CONTROL        | TIME SYNC CONTROL REGISTER                 | <a href="#">Section 14.5.3.2</a> |
| Ch     | CPTS_TS_PUSH        | TIME STAMP EVENT PUSH REGISTER             | <a href="#">Section 14.5.3.3</a> |
| 10h    | CPTS_TS_LOAD_VAL    | TIME STAMP LOAD VALUE REGISTER             | <a href="#">Section 14.5.3.4</a> |
| 14h    | CPTS_TS_LOAD_EN     | TIME STAMP LOAD ENABLE REGISTER            | <a href="#">Section 14.5.3.5</a> |
| 20h    | CPTS_INTSTAT_RAW    | TIME SYNC INTERRUPT STATUS RAW REGISTER    | <a href="#">Section 14.5.3.6</a> |
| 24h    | CPTS_INTSTAT_MASKED | TIME SYNC INTERRUPT STATUS MASKED REGISTER | <a href="#">Section 14.5.3.7</a> |
| 28h    | CPTS_INT_ENABLE     | TIME SYNC INTERRUPT ENABLE REGISTER        | <a href="#">Section 14.5.3.8</a> |

**Table 14-89. CPSW\_CPTS REGISTERS (continued)**

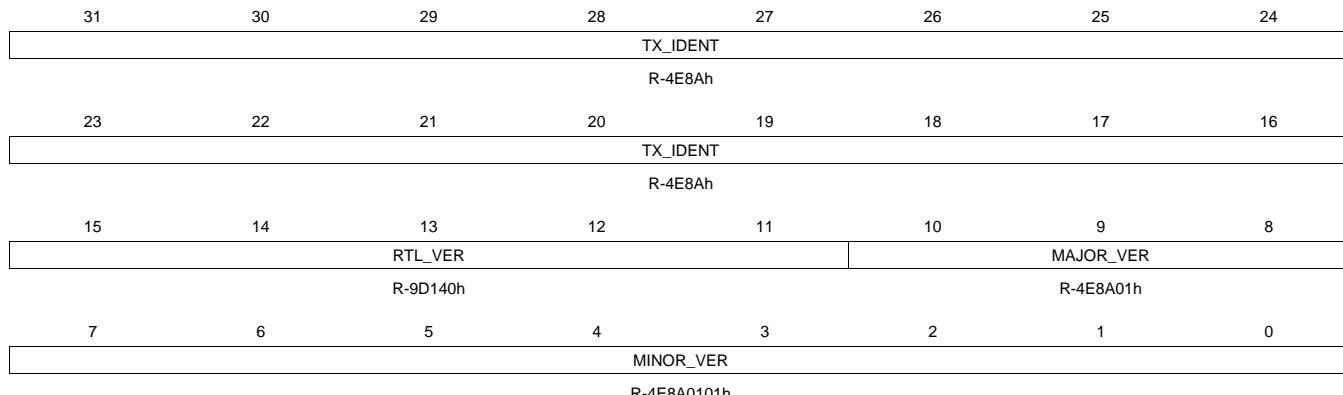
| <b>Offset</b> | <b>Acronym</b>  | <b>Register Name</b>             | <b>Section</b>                    |
|---------------|-----------------|----------------------------------|-----------------------------------|
| 30h           | CPTS_EVENT_POP  | EVENT INTERRUPT POP REGISTER     | <a href="#">Section 14.5.3.9</a>  |
| 34h           | CPTS_EVENT_LOW  | LOWER 32-BITS OF THE EVENT VALUE | <a href="#">Section 14.5.3.10</a> |
| 38h           | CPTS_EVENT_HIGH | UPPER 32-BITS OF THE EVENT VALUE | <a href="#">Section 14.5.3.11</a> |

#### 14.5.3.1 CPTS\_IDVER Register (offset = 0h) [reset = 4E8A0101h]

CPTS\_IDVER is shown in [Figure 14-78](#) and described in [Table 14-90](#).

##### IDENTIFICATION AND VERSION REGISTER

**Figure 14-78. CPTS\_IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-90. CPTS\_IDVER Register Field Descriptions**

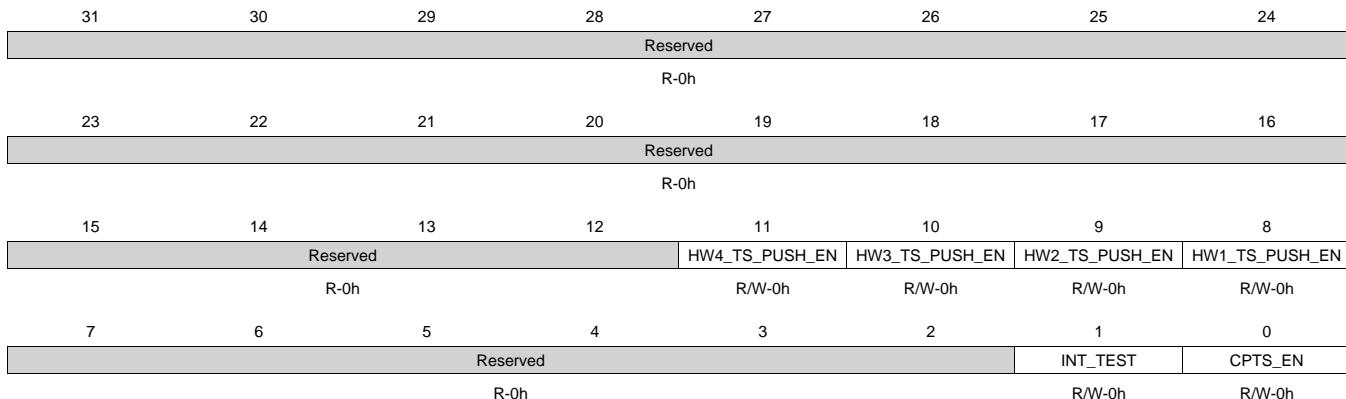
| Bit   | Field     | Type | Reset     | Description             |
|-------|-----------|------|-----------|-------------------------|
| 31-16 | TX_IDENT  | R    | 4E8Ah     | TX Identification Value |
| 15-11 | RTL_VER   | R    | 9D140h    | RTL Version Value       |
| 10-8  | MAJOR_VER | R    | 4E8A01h   | Major Version Value     |
| 7-0   | MINOR_VER | R    | 4E8A0101h | Minor Version Value     |

#### 14.5.3.2 CPTS\_CONTROL Register (offset = 4h) [reset = 0h]

CPTS\_CONTROL is shown in [Figure 14-79](#) and described in [Table 14-91](#).

##### TIME SYNC CONTROL REGISTER

**Figure 14-79. CPTS\_CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-91. CPTS\_CONTROL Register Field Descriptions**

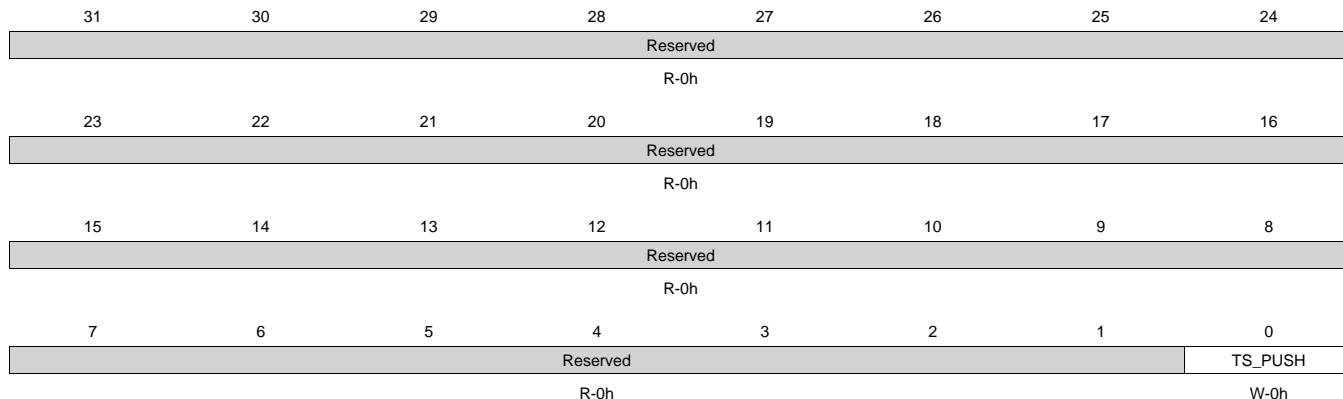
| Bit   | Field          | Type | Reset | Description  |
|-------|----------------|------|-------|--|
| 31-12 | Reserved       | R    | 0h    |  |
| 11    | HW4_TS_PUSH_EN | R/W  | 0h    | Hardware push 4 enable   |
| 10    | HW3_TS_PUSH_EN | R/W  | 0h    | Hardware push 3 enable   |
| 9     | HW2_TS_PUSH_EN | R/W  | 0h    | Hardware push 2 enable   |
| 8     | HW1_TS_PUSH_EN | R/W  | 0h    | Hardware push 1 enable   |
| 7-2   | Reserved       | R    | 0h    |  |
| 1     | INT_TEST       | R/W  | 0h    | Interrupt Test - When set, this bit allows the raw interrupt to be written to facilitate interrupt test.                                 |
| 0     | CPTS_EN        | R/W  | 0h    | Time Sync Enable - When disabled (cleared to zero), the RCLK domain is held in reset.<br>0 - Time Sync Disabled<br>1 - Time Sync Enabled |

#### 14.5.3.3 CPTS\_TS\_PUSH Register (offset = Ch) [reset = 0h]

CPTS\_TS\_PUSH is shown in [Figure 14-80](#) and described in [Table 14-92](#).

##### TIME STAMP EVENT PUSH REGISTER

**Figure 14-80. CPTS\_TS\_PUSH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-92. CPTS\_TS\_PUSH Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-1 | Reserved | R    | 0h    |  |
| 0    | TS_PUSH  | W    | 0h    | <p>Time stamp event push - When a logic high is written to this bit a time stamp event is pushed onto the event FIFO.</p> <p>The time stamp value is the time of the write of this register, not the time of the event read.</p> <p>The time stamp value can then be read on interrupt via the event registers.</p> <p>Software should not push a second time stamp event onto the event FIFO until the first time stamp value has been read from the event FIFO (there should be only one time stamp event in the event FIFO at any given time).</p> <p>This bit is write only and always reads zero.</p> |

#### 14.5.3.4 CPTS\_TS\_LOAD\_VAL Register (offset = 10h) [reset = 0h]

CPTS\_TS\_LOAD\_VAL is shown in [Figure 14-81](#) and described in [Table 14-93](#).

##### TIME STAMP LOAD VALUE REGISTER

**Figure 14-81. CPTS\_TS\_LOAD\_VAL Register**

| 31          | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TS_LOAD_VAL |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-93. CPTS\_TS\_LOAD\_VAL Register Field Descriptions**

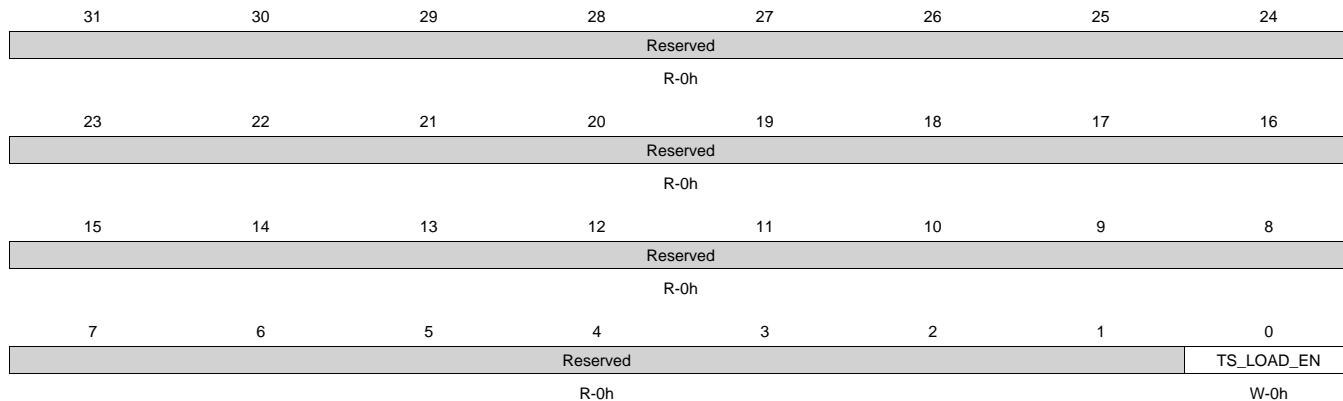
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-0 | TS_LOAD_VAL | R/W  | 0h    | Time Stamp Load Value - Writing the ts_load_en bit causes the value contained in this register to be written into the time stamp. The time stamp value is read by initiating a time stamp push event, not by reading this register. When reading this register, the value read is not the time stamp, but is the value that was last written to this register. |

#### 14.5.3.5 CPTS\_TS\_LOAD\_EN Register (offset = 14h) [reset = 0h]

CPTS\_TS\_LOAD\_EN is shown in [Figure 14-82](#) and described in [Table 14-94](#).

##### TIME STAMP LOAD ENABLE REGISTER

**Figure 14-82. CPTS\_TS\_LOAD\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-94. CPTS\_TS\_LOAD\_EN Register Field Descriptions**

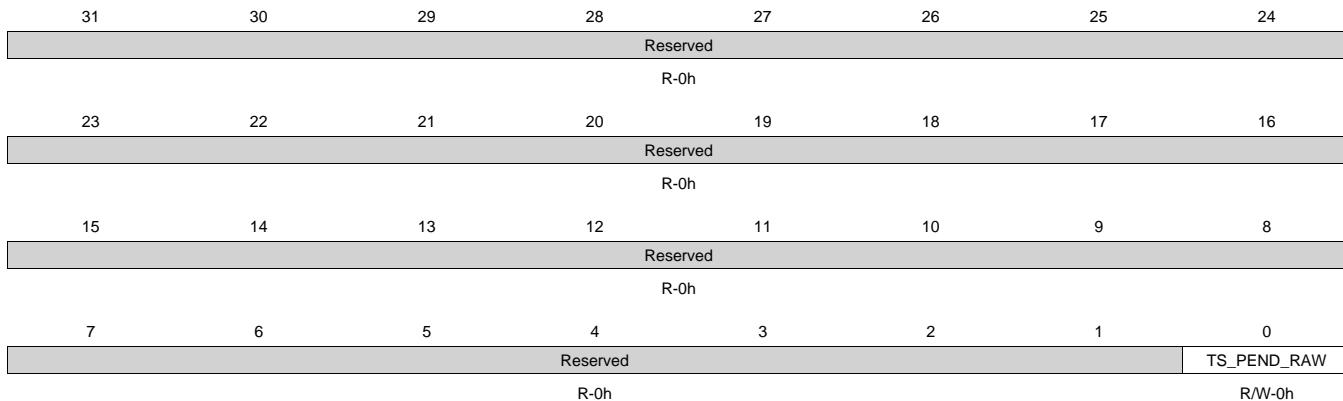
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-1 | Reserved   | R    | 0h    |  |
| 0    | TS_LOAD_EN | W    | 0h    | Time Stamp Load - Writing a one to this bit enables the time stamp value to be written via the ts_load_val[31:0] register.<br>This feature is included for test purposes.<br>This bit is write only. |

#### 14.5.3.6 CPTS\_INTSTAT\_RAW Register (offset = 20h) [reset = 0h]

CPTS\_INTSTAT\_RAW is shown in [Figure 14-83](#) and described in [Table 14-95](#).

##### TIME SYNC INTERRUPT STATUS RAW REGISTER

**Figure 14-83. CPTS\_INTSTAT\_RAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-95. CPTS\_INTSTAT\_RAW Register Field Descriptions**

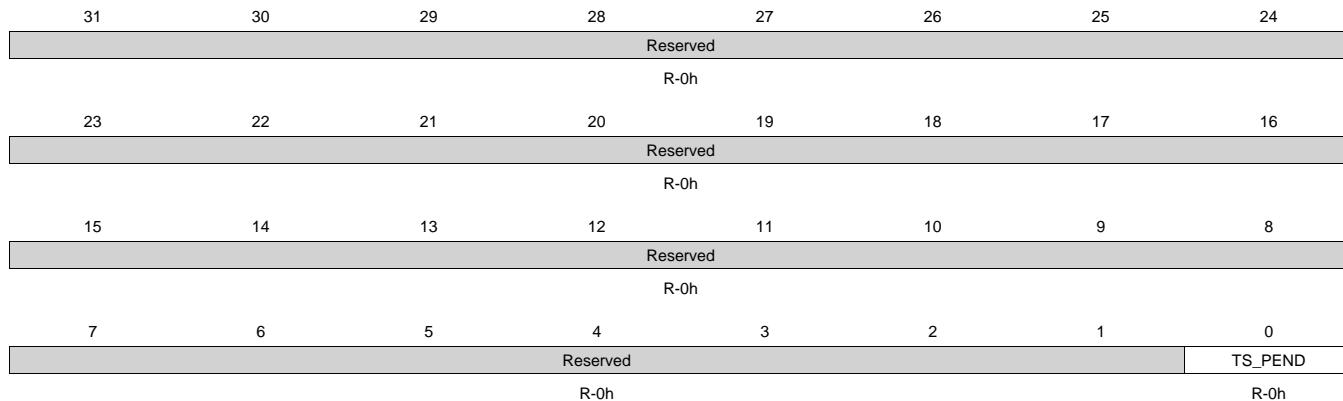
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-1 | Reserved    | R    | 0h    |  |
| 0    | TS_PEND_RAW | R/W  | 0h    | TS_PEND_RAW int read (before enable). Writable when int_test = 1 A one in this bit indicates that there is one or more events in the event FIFO. |

#### 14.5.3.7 CPTS\_INTSTAT\_MASKED Register (offset = 24h) [reset = 0h]

CPTS\_INTSTAT\_MASKED is shown in [Figure 14-84](#) and described in [Table 14-96](#).

##### TIME SYNC INTERRUPT STATUS MASKED REGISTER

**Figure 14-84. CPTS\_INTSTAT\_MASKED Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-96. CPTS\_INTSTAT\_MASKED Register Field Descriptions**

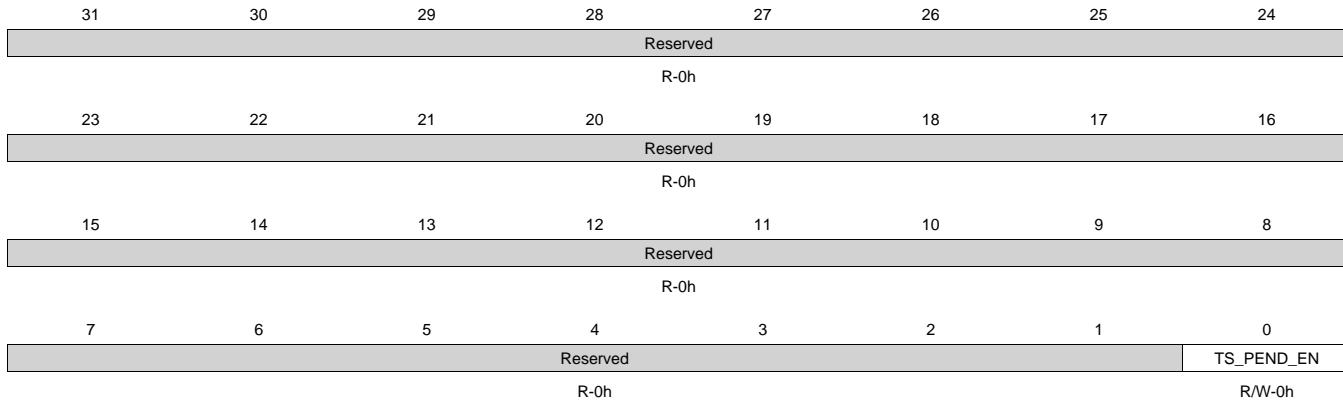
| Bit  | Field    | Type | Reset | Description                                   |
|------|----------|------|-------|---|
| 31-1 | Reserved | R    | 0h    |   |
| 0    | TS_PEND  | R    | 0h    | TS_PEND masked interrupt read (after enable). |

#### 14.5.3.8 CPTS\_INT\_ENABLE Register (offset = 28h) [reset = 0h]

CPTS\_INT\_ENABLE is shown in [Figure 14-85](#) and described in [Table 14-97](#).

##### TIME SYNC INTERRUPT ENABLE REGISTER

**Figure 14-85. CPTS\_INT\_ENABLE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-97. CPTS\_INT\_ENABLE Register Field Descriptions**

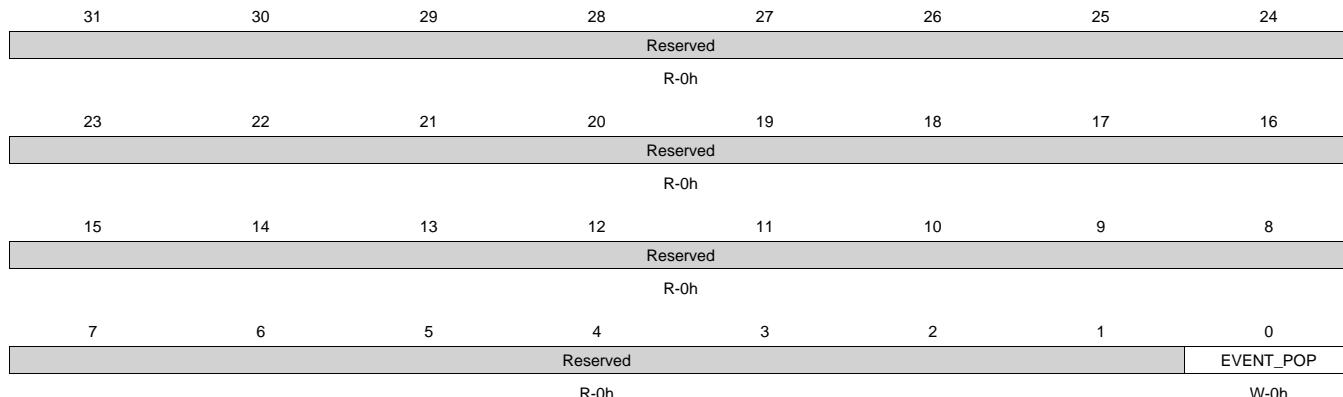
| Bit  | Field      | Type | Reset | Description                      |
|------|------------|------|-------|----------------------------------|
| 31-1 | Reserved   | R    | 0h    |                                  |
| 0    | TS_PEND_EN | R/W  | 0h    | TS_PEND masked interrupt enable. |

#### 14.5.3.9 CPTS\_EVENT\_POP Register (offset = 30h) [reset = 0h]

CPTS\_EVENT\_POP is shown in [Figure 14-86](#) and described in [Table 14-98](#).

##### EVENT INTERRUPT POP REGISTER

**Figure 14-86. CPTS\_EVENT\_POP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-98. CPTS\_EVENT\_POP Register Field Descriptions**

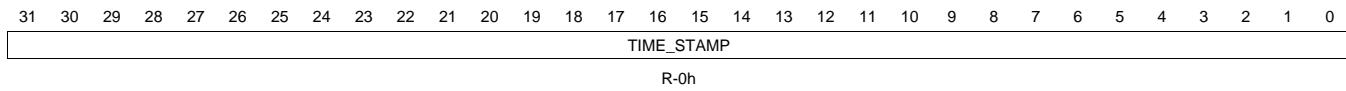
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-1 | Reserved  | R    | 0h    |  |
| 0    | EVENT_POP | W    | 0h    | Event Pop - When a logic high is written to this bit an event is popped off the event FIFO.<br>The event FIFO pop occurs as part of the interrupt process after the event has been read in the Event_Low and Event_High registers.<br>Popping an event discards the event and causes the next event, if any, to be moved to the top of the FIFO ready to be read by software on interrupt. |

#### 14.5.3.10 CPTS\_EVENT\_LOW Register (offset = 34h) [reset = 0h]

CPTS\_EVENT\_LOW is shown in [Figure 14-87](#) and described in [Table 14-99](#).

LOWER 32-BITS OF THE EVENT VALUE

**Figure 14-87. CPTS\_EVENT\_LOW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-99. CPTS\_EVENT\_LOW Register Field Descriptions**

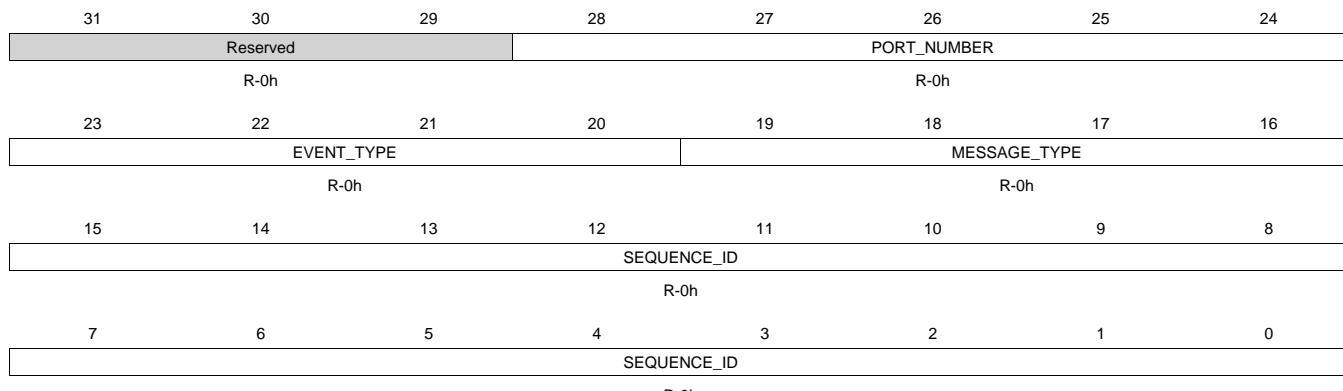
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-0 | TIME_STAMP | R    | 0h    | Time Stamp - The timestamp is valid for transmit, receive, and time stamp push event types.<br>The timestamp value is not valid for counter roll event types. |

#### 14.5.3.11 CPTS\_EVENT\_HIGH Register (offset = 38h) [reset = 0h]

CPTS\_EVENT\_HIGH is shown in [Figure 14-88](#) and described in [Table 14-100](#).

UPPER 32-BITS OF THE EVENT VALUE

**Figure 14-88. CPTS\_EVENT\_HIGH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-100. CPTS\_EVENT\_HIGH Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31-29 | Reserved     | R    | 0h    |   |
| 28-24 | PORT_NUMBER  | R    | 0h    | Port Number - indicates the port number of an ethernet event or the hardware push pin number (1 to 4).  |
| 23-20 | EVENT_TYPE   | R    | 0h    | Time Sync Event Type<br>0000 - Time Stamp Push Event<br>0001 - Time Stamp Rollover Event<br>0010 - Time Stamp Half Rollover Event<br>0011 - Hardware Time Stamp Push Event<br>0100 - Ethernet Receive Event<br>0101 - Ethernet Transmit Event |
| 19-16 | MESSAGE_TYPE | R    | 0h    | Message type - The message type value that was contained in an ethernet transmit or receive time sync packet.<br>This field is valid only for ethernet transmit or receive events.  |
| 15-0  | SEQUENCE_ID  | R    | 0h    | Sequence ID - The 16-bit sequence id is the value that was contained in an ethernet transmit or receive time sync packet.<br>This field is valid only for ethernet transmit or receive events.  |

#### 14.5.4 CPSW\_STATS Registers

For a full description of the CPSW\_STATS registers, see [Section 14.3.2.20, CPSW\\_3G Network Statistics](#). The registers are summarized in [Table 14-101](#).

**Table 14-101. CPSW\_STATS REGISTERS**

| Offset | Acronym | Register Name                 | Section                                |
|--------|---------|-------------------------------|--|
| 00h    |         | Good Rx Frames                | <a href="#">Section 14.3.2.20.1.1</a>  |
| 04h    |         | Broadcast Rx Frames           | <a href="#">Section 14.3.2.20.1.2</a>  |
| 08h    |         | Multicast Rx Frames           | <a href="#">Section 14.3.2.20.1.3</a>  |
| 0Ch    |         | Pause Rx Frames               | <a href="#">Section 14.3.2.20.1.4</a>  |
| 10h    |         | Rx CRC Errors                 | <a href="#">Section 14.3.2.20.1.5</a>  |
| 14h    |         | Rx Align/Code Errors          | <a href="#">Section 14.3.2.20.1.6</a>  |
| 18h    |         | Oversize Rx Frames            | <a href="#">Section 14.3.2.20.1.7</a>  |
| 1Ch    |         | Rx Jabbers                    | <a href="#">Section 14.3.2.20.1.8</a>  |
| 20h    |         | Undersize (Short) Rx Frames   | <a href="#">Section 14.3.2.20.1.9</a>  |
| 24h    |         | Rx Fragments                  | <a href="#">Section 14.3.2.20.1.10</a> |
| 30h    |         | Rx Octets                     | <a href="#">Section 14.3.2.20.1.14</a> |
| 34h    |         | Good Tx Frames                | <a href="#">Section 14.3.2.20.2.1</a>  |
| 38h    |         | Broadcast Tx Frames           | <a href="#">Section 14.3.2.20.2.2</a>  |
| 3Ch    |         | Multicast Tx Frames           | <a href="#">Section 14.3.2.20.2.3</a>  |
| 40h    |         | Pause Tx Frames               | <a href="#">Section 14.3.2.20.2.4</a>  |
| 44h    |         | Deferred Tx Frames            | <a href="#">Section 14.3.2.20.2.11</a> |
| 48h    |         | Collisions                    | <a href="#">Section 14.3.2.20.2.5</a>  |
| 4Ch    |         | Single Collision Tx Frames    | <a href="#">Section 14.3.2.20.2.6</a>  |
| 50h    |         | Multiple Collision Tx Frames  | <a href="#">Section 14.3.2.20.2.7</a>  |
| 54h    |         | Excessive Collisions          | <a href="#">Section 14.3.2.20.2.8</a>  |
| 58h    |         | Late Collisions               | <a href="#">Section 14.3.2.20.2.9</a>  |
| 5Ch    |         | Tx Underrun                   | <a href="#">Section 14.3.2.20.2.10</a> |
| 60h    |         | Carrier Sense Errors          | <a href="#">Section 14.3.2.20.2.12</a> |
| 64h    |         | Tx Octets                     | <a href="#">Section 14.3.2.20.2.13</a> |
| 68h    |         | Rx + Tx 64 Octet Frames       | <a href="#">Section 14.3.2.20.3.1</a>  |
| 6Ch    |         | Rx + Tx 65–127 Octet Frames   | <a href="#">Section 14.3.2.20.3.2</a>  |
| 70h    |         | Rx + Tx 128–255 Octet Frames  | <a href="#">Section 14.3.2.20.3.3</a>  |
| 74h    |         | Rx + Tx 256–511 Octet Frames  | <a href="#">Section 14.3.2.20.3.4</a>  |
| 78h    |         | Rx + Tx 512–1023 Octet Frames | <a href="#">Section 14.3.2.20.3.5</a>  |
| 7Ch    |         | Rx + Tx 1024_Up Octet Frames  | <a href="#">Section 14.3.2.20.3.6</a>  |
| 80h    |         | Net Octets                    | <a href="#">Section 14.3.2.20.1.15</a> |
| 84h    |         | Rx Start of Frame Overruns    | <a href="#">Section 14.3.2.20.1.11</a> |
| 88h    |         | Rx Middle of Frame Overruns   | <a href="#">Section 14.3.2.20.1.12</a> |
| 8Ch    |         | Rx DMA Overruns               | <a href="#">Section 14.3.2.20.1.13</a> |

#### 14.5.5 CPDMA\_STATERAM Registers

[Table 14-102](#) lists the memory-mapped registers for the CPSW\_CPDMA. All register offset addresses not listed in [Table 14-102](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-102. CPDMA\_STATERAM REGISTERS**

| Offset | Acronym | Register Name                                      | Section                          |
|--------|---------|--|----------------------------------|
| 00h    | TX0_HDP | CPDMA_STATERAM TX CHANNEL 0 HEAD DESC<br>POINTER * | <a href="#">Section 14.5.5.1</a> |

**Table 14-102. CPDMA\_STATERAM REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b>   | <b>Section</b>                    |
|---------------|----------------|--|-----------------------------------|
| 04h           | TX1_HDP        | CPDMA_STATERAM TX CHANNEL 1 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.2</a>  |
| 08h           | TX2_HDP        | CPDMA_STATERAM TX CHANNEL 2 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.3</a>  |
| 0Ch           | TX3_HDP        | CPDMA_STATERAM TX CHANNEL 3 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.4</a>  |
| 10h           | TX4_HDP        | CPDMA_STATERAM TX CHANNEL 4 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.5</a>  |
| 14h           | TX5_HDP        | CPDMA_STATERAM TX CHANNEL 5 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.6</a>  |
| 18h           | TX6_HDP        | CPDMA_STATERAM TX CHANNEL 6 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.7</a>  |
| 1Ch           | TX7_HDP        | CPDMA_STATERAM TX CHANNEL 7 HEAD DESC<br>POINTER *           | <a href="#">Section 14.5.5.8</a>  |
| 20h           | RX0_HDP        | CPDMA_STATERAM RX 0 CHANNEL 0 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.9</a>  |
| 24h           | RX1_HDP        | CPDMA_STATERAM RX 1 CHANNEL 1 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.10</a> |
| 28h           | RX2_HDP        | CPDMA_STATERAM RX 2 CHANNEL 2 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.11</a> |
| 2Ch           | RX3_HDP        | CPDMA_STATERAM RX 3 CHANNEL 3 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.12</a> |
| 30h           | RX4_HDP        | CPDMA_STATERAM RX 4 CHANNEL 4 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.13</a> |
| 34h           | RX5_HDP        | CPDMA_STATERAM RX 5 CHANNEL 5 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.14</a> |
| 38h           | RX6_HDP        | CPDMA_STATERAM RX 6 CHANNEL 6 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.15</a> |
| 3Ch           | RX7_HDP        | CPDMA_STATERAM RX 7 CHANNEL 7 HEAD DESC<br>POINTER *         | <a href="#">Section 14.5.5.16</a> |
| 40h           | TX0_CP         | CPDMA_STATERAM TX CHANNEL 0 COMPLETION<br>POINTER REGISTER   | <a href="#">Section 14.5.5.17</a> |
| 44h           | TX1_CP         | CPDMA_STATERAM TX CHANNEL 1 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.18</a> |
| 48h           | TX2_CP         | CPDMA_STATERAM TX CHANNEL 2 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.19</a> |
| 4Ch           | TX3_CP         | CPDMA_STATERAM TX CHANNEL 3 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.20</a> |
| 50h           | TX4_CP         | CPDMA_STATERAM TX CHANNEL 4 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.21</a> |
| 54h           | TX5_CP         | CPDMA_STATERAM TX CHANNEL 5 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.22</a> |
| 58h           | TX6_CP         | CPDMA_STATERAM TX CHANNEL 6 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.23</a> |
| 5Ch           | TX7_CP         | CPDMA_STATERAM TX CHANNEL 7 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.24</a> |
| 60h           | RX0_CP         | CPDMA_STATERAM RX CHANNEL 0 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.25</a> |
| 64h           | RX1_CP         | CPDMA_STATERAM RX CHANNEL 1 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.26</a> |
| 68h           | RX2_CP         | CPDMA_STATERAM RX CHANNEL 2 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.27</a> |
| 6Ch           | RX3_CP         | CPDMA_STATERAM RX CHANNEL 3 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.28</a> |
| 70h           | RX4_CP         | CPDMA_STATERAM RX CHANNEL 4 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.29</a> |

**Table 14-102. CPDMA\_STATERAM REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b>   | <b>Section</b>                    |
|---------------|----------------|--|-----------------------------------|
| 74h           | RX5_CP         | CPDMA_STATERAM RX CHANNEL 5 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.30</a> |
| 78h           | RX6_CP         | CPDMA_STATERAM RX CHANNEL 6 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.31</a> |
| 7Ch           | RX7_CP         | CPDMA_STATERAM RX CHANNEL 7 COMPLETION<br>POINTER REGISTER * | <a href="#">Section 14.5.5.32</a> |

#### 14.5.5.1 TX0\_HDP Register (offset = A00h) [reset = 0h]

TX0\_HDP is shown in [Figure 14-89](#) and described in [Table 14-103](#).

CPDMA\_STATERAM TX CHANNEL 0 HEAD DESC POINTER \*

**Figure 14-89. TX0\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-103. TX0\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.2 TX1\_HDP Register (offset = A04h) [reset = 0h]

TX1\_HDP is shown in [Figure 14-90](#) and described in [Table 14-104](#).

CPDMA\_STATERAM TX CHANNEL 1 HEAD DESC POINTER \*

**Figure 14-90. TX1\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-104. TX1\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.3 TX2\_HDP Register (offset = A08h) [reset = 0h]

TX2\_HDP is shown in [Figure 14-91](#) and described in [Table 14-105](#).

CPDMA\_STATERAM TX CHANNEL 2 HEAD DESC POINTER \*

**Figure 14-91. TX2\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-105. TX2\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.4 TX3\_HDP Register (offset = A0Ch) [reset = 0h]

TX3\_HDP is shown in [Figure 14-92](#) and described in [Table 14-106](#).

CPDMA\_STATERAM TX CHANNEL 3 HEAD DESC POINTER \*

**Figure 14-92. TX3\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-106. TX3\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.5 TX4\_HDP Register (offset = A10h) [reset = 0h]

TX4\_HDP is shown in [Figure 14-93](#) and described in [Table 14-107](#).

CPDMA\_STATERAM TX CHANNEL 4 HEAD DESC POINTER \*

**Figure 14-93. TX4\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-107. TX4\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.6 TX5\_HDP Register (offset = A14h) [reset = 0h]

TX5\_HDP is shown in [Figure 14-94](#) and described in [Table 14-108](#).

CPDMA\_STATERAM TX CHANNEL 5 HEAD DESC POINTER \*

**Figure 14-94. TX5\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-108. TX5\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.7 TX6\_HDP Register (offset = A18h) [reset = 0h]

TX6\_HDP is shown in [Figure 14-95](#) and described in [Table 14-109](#).

CPDMA\_STATERAM TX CHANNEL 6 HEAD DESC POINTER \*

**Figure 14-95. TX6\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-109. TX6\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.8 TX7\_HDP Register (offset = A1Ch) [reset = 0h]

TX7\_HDP is shown in [Figure 14-96](#) and described in [Table 14-110](#).

CPDMA\_STATERAM TX CHANNEL 7 HEAD DESC POINTER \*

**Figure 14-96. TX7\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-110. TX7\_HDP Register Field Descriptions**

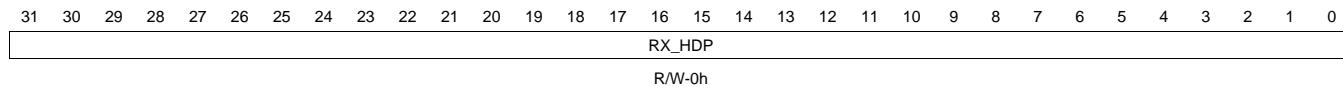
| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | TX_HDP | R/W  | 0h    | TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.9 RX0\_HDP Register (offset = A20h) [reset = 0h]

RX0\_HDP is shown in [Figure 14-97](#) and described in [Table 14-111](#).

CPDMA\_STATERAM RX 0 CHANNEL 0 HEAD DESC POINTER \*

**Figure 14-97. RX0\_HDP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-111. RX0\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.10 RX1\_HDP Register (offset = A24h) [reset = 0h]

RX1\_HDP is shown in [Figure 14-98](#) and described in [Table 14-112](#).

CPDMA\_STATERAM RX 1 CHANNEL 1 HEAD DESC POINTER \*

**Figure 14-98. RX1\_HDP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-112. RX1\_HDP Register Field Descriptions**

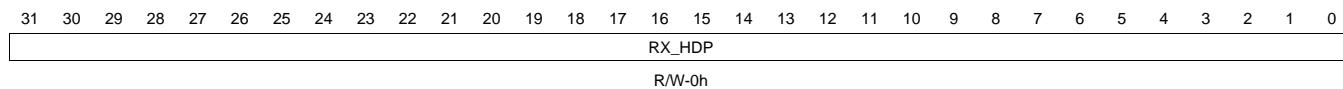
| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.11 RX2\_HDP Register (offset = A28h) [reset = 0h]

RX2\_HDP is shown in [Figure 14-99](#) and described in [Table 14-113](#).

CPDMA\_STATERAM RX 2 CHANNEL 2 HEAD DESC POINTER \*

**Figure 14-99. RX2\_HDP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-113. RX2\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.12 RX3\_HDP Register (offset = A2Ch) [reset = 0h]

RX3\_HDP is shown in [Figure 14-100](#) and described in [Table 14-114](#).

CPDMA\_STATERAM RX 3 CHANNEL 3 HEAD DESC POINTER \*

**Figure 14-100. RX3\_HDP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-114. RX3\_HDP Register Field Descriptions**

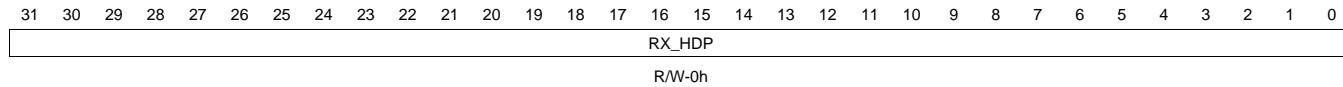
| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.13 RX4\_HDP Register (offset = A30h) [reset = 0h]

RX4\_HDP is shown in [Figure 14-101](#) and described in [Table 14-115](#).

CPDMA\_STATERAM RX 4 CHANNEL 4 HEAD DESC POINTER \*

**Figure 14-101. RX4\_HDP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-115. RX4\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.14 RX5\_HDP Register (offset = A34h) [reset = 0h]

RX5\_HDP is shown in [Figure 14-102](#) and described in [Table 14-116](#).

CPDMA\_STATERAM RX 5 CHANNEL 5 HEAD DESC POINTER \*

**Figure 14-102. RX5\_HDP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| RX_HDP |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-116. RX5\_HDP Register Field Descriptions**

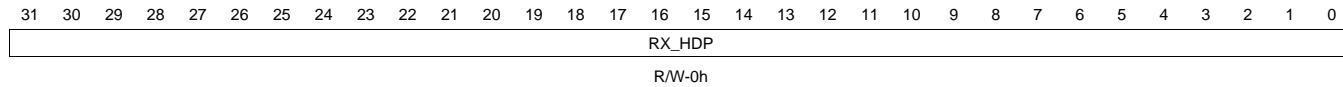
| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.15 RX6\_HDP Register (offset = A38h) [reset = 0h]

RX6\_HDP is shown in [Figure 14-103](#) and described in [Table 14-117](#).

CPDMA\_STATERAM RX 6 CHANNEL 6 HEAD DESC POINTER \*

**Figure 14-103. RX6\_HDP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-117. RX6\_HDP Register Field Descriptions**

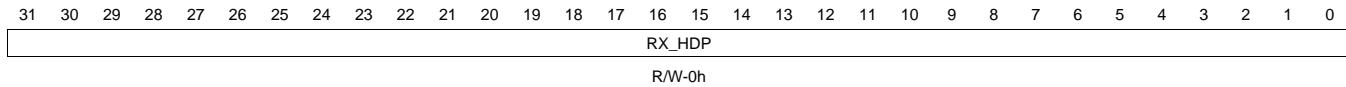
| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.16 RX7\_HDP Register (offset = A3Ch) [reset = 0h]

RX7\_HDP is shown in [Figure 14-104](#) and described in [Table 14-118](#).

CPDMA\_STATERAM RX 7 CHANNEL 7 HEAD DESC POINTER \*

**Figure 14-104. RX7\_HDP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-118. RX7\_HDP Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | RX_HDP | R/W  | 0h    | RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received.<br>Writing to these locations when they are non-zero is an error (except at reset).<br>Host software must initialize these locations to zero on reset. |

#### 14.5.5.17 TX0\_CP Register (offset = A40h) [reset = 0h]

TX0\_CP is shown in [Figure 14-105](#) and described in [Table 14-119](#).

CPDMA\_STATERAM TX CHANNEL 0 COMPLETION POINTER REGISTER

**Figure 14-105. TX0\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-119. TX0\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.18 TX1\_CP Register (offset = A44h) [reset = 0h]

TX1\_CP is shown in [Figure 14-106](#) and described in [Table 14-120](#).

CPDMA\_STATERAM TX CHANNEL 1 COMPLETION POINTER REGISTER \*

**Figure 14-106. TX1\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-120. TX1\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.19 TX2\_CP Register (offset = A48h) [reset = 0h]

TX2\_CP is shown in [Figure 14-107](#) and described in [Table 14-121](#).

CPDMA\_STATERAM TX CHANNEL 2 COMPLETION POINTER REGISTER \*

**Figure 14-107. TX2\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-121. TX2\_CP Register Field Descriptions**

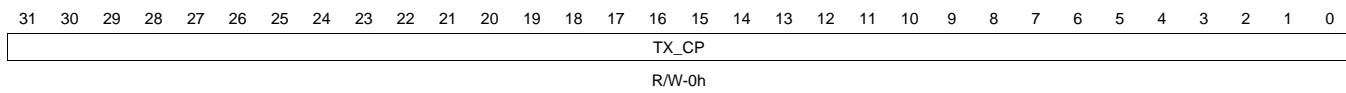
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.20 TX3\_CP Register (offset = A4Ch) [reset = 0h]

TX3\_CP is shown in [Figure 14-108](#) and described in [Table 14-122](#).

CPDMA\_STATERAM TX CHANNEL 3 COMPLETION POINTER REGISTER \*

**Figure 14-108. TX3\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-122. TX3\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.21 TX4\_CP Register (offset = A50h) [reset = 0h]

TX4\_CP is shown in [Figure 14-109](#) and described in [Table 14-123](#).

CPDMA\_STATERAM TX CHANNEL 4 COMPLETION POINTER REGISTER \*

**Figure 14-109. TX4\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-123. TX4\_CP Register Field Descriptions**

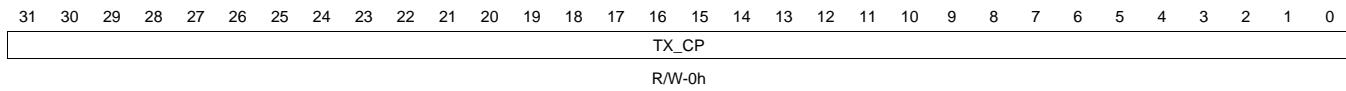
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.22 TX5\_CP Register (offset = A54h) [reset = 0h]

TX5\_CP is shown in [Figure 14-110](#) and described in [Table 14-124](#).

CPDMA\_STATERAM TX CHANNEL 5 COMPLETION POINTER REGISTER \*

**Figure 14-110. TX5\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-124. TX5\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.23 TX6\_CP Register (offset = A58h) [reset = 0h]

TX6\_CP is shown in [Figure 14-111](#) and described in [Table 14-125](#).

CPDMA\_STATERAM TX CHANNEL 6 COMPLETION POINTER REGISTER \*

**Figure 14-111. TX6\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-125. TX6\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.24 TX7\_CP Register (offset = A5Ch) [reset = 0h]

TX7\_CP is shown in [Figure 14-112](#) and described in [Table 14-126](#).

CPDMA\_STATERAM TX CHANNEL 7 COMPLETION POINTER REGISTER \*

**Figure 14-112. TX7\_CP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-126. TX7\_CP Register Field Descriptions**

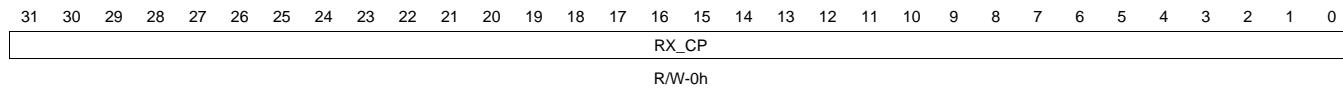
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | TX_CP | R/W  | 0h    | Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.<br>The port uses the value written to determine if the interrupt should be deasserted. |

#### 14.5.5.25 RX0\_CP Register (offset = A60h) [reset = 0h]

RX0\_CP is shown in [Figure 14-113](#) and described in [Table 14-127](#).

CPDMA\_STATERAM RX CHANNEL 0 COMPLETION POINTER REGISTER \*

**Figure 14-113. RX0\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-127. RX0\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.26 RX1\_CP Register (offset = A64h) [reset = 0h]

RX1\_CP is shown in [Figure 14-114](#) and described in [Table 14-128](#).

CPDMA\_STATERAM RX CHANNEL 1 COMPLETION POINTER REGISTER \*

**Figure 14-114. RX1\_CP Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| RX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-128. RX1\_CP Register Field Descriptions**

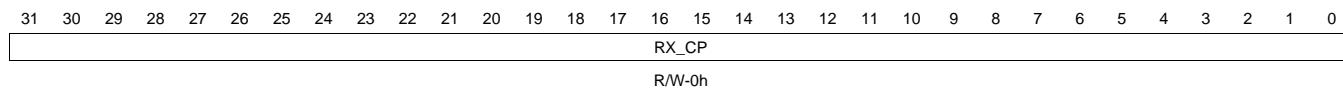
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.27 RX2\_CP Register (offset = A68h) [reset = 0h]

RX2\_CP is shown in [Figure 14-115](#) and described in [Table 14-129](#).

CPDMA\_STATERAM RX CHANNEL 2 COMPLETION POINTER REGISTER \*

**Figure 14-115. RX2\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-129. RX2\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.28 RX3\_CP Register (offset = A6Ch) [reset = 0h]

RX3\_CP is shown in [Figure 14-116](#) and described in [Table 14-130](#).

CPDMA\_STATERAM RX CHANNEL 3 COMPLETION POINTER REGISTER \*

**Figure 14-116. RX3\_CP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-130. RX3\_CP Register Field Descriptions**

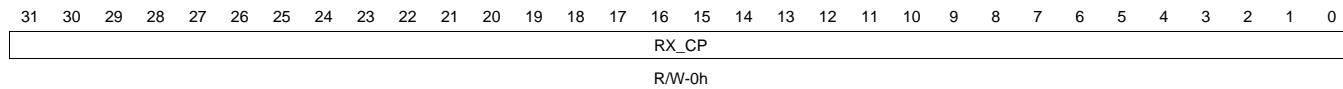
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.29 RX4\_CP Register (offset = A70h) [reset = 0h]

RX4\_CP is shown in [Figure 14-117](#) and described in [Table 14-131](#).

CPDMA\_STATERAM RX CHANNEL 4 COMPLETION POINTER REGISTER \*

**Figure 14-117. RX4\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-131. RX4\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.30 RX5\_CP Register (offset = A74h) [reset = 0h]

RX5\_CP is shown in [Figure 14-118](#) and described in [Table 14-132](#).

CPDMA\_STATERAM RX CHANNEL 5 COMPLETION POINTER REGISTER \*

**Figure 14-118. RX5\_CP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-132. RX5\_CP Register Field Descriptions**

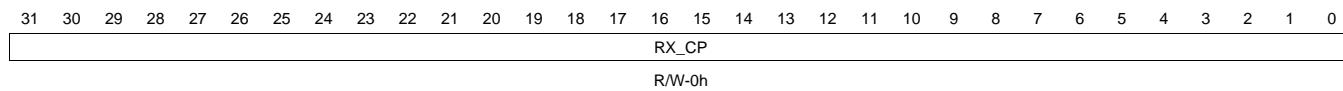
| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.31 RX6\_CP Register (offset = A78h) [reset = 0h]

RX6\_CP is shown in [Figure 14-119](#) and described in [Table 14-133](#).

CPDMA\_STATERAM RX CHANNEL 6 COMPLETION POINTER REGISTER \*

**Figure 14-119. RX6\_CP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-133. RX6\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.5.32 RX7\_CP Register (offset = A7Ch) [reset = 0h]

RX7\_CP is shown in [Figure 14-120](#) and described in [Table 14-134](#).

CPDMA\_STATERAM RX CHANNEL 7 COMPLETION POINTER REGISTER \*

**Figure 14-120. RX7\_CP Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_CP  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-134. RX7\_CP Register Field Descriptions**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 31-0 | RX_CP | R/W  | 0h    | <p>Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing.</p> <p>The port uses the value written to determine if the interrupt should be deasserted.</p> <p>Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port).</p> <p>The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted.</p> <p>The value written is not actually stored in the location.</p> <p>The interrupt is deasserted if the two values are equal.</p> |

#### 14.5.6 CPSW\_PORT Registers

[Table 14-135](#) lists the memory-mapped registers for the CPSW\_PORT. All register offset addresses not listed in [Table 14-135](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-135. CPSW\_PORT REGISTERS**

| Offset | Acronym             | Register Name | Section                           |
|--------|---------------------|---------------|-----------------------------------|
| 0h     | P0_CONTROL          |               | <a href="#">Section 14.5.6.1</a>  |
| 8h     | P0_MAX_BLKS         |               | <a href="#">Section 14.5.6.2</a>  |
| Ch     | P0_BLK_CNT          |               | <a href="#">Section 14.5.6.3</a>  |
| 10h    | P0_TX_IN_CTL        |               | <a href="#">Section 14.5.6.4</a>  |
| 14h    | P0_PORT_VLAN        |               | <a href="#">Section 14.5.6.5</a>  |
| 18h    | P0_TX_PRI_MAP       |               | <a href="#">Section 14.5.6.6</a>  |
| 1Ch    | P0_CPDMA_TX_PRI_MAP |               | <a href="#">Section 14.5.6.7</a>  |
| 20h    | P0_CPDMA_RX_CH_MAP  |               | <a href="#">Section 14.5.6.8</a>  |
| 30h    | P0_RX_DSCP_PRI_MAP0 |               | <a href="#">Section 14.5.6.9</a>  |
| 34h    | P0_RX_DSCP_PRI_MAP1 |               | <a href="#">Section 14.5.6.10</a> |
| 38h    | P0_RX_DSCP_PRI_MAP2 |               | <a href="#">Section 14.5.6.11</a> |
| 3Ch    | P0_RX_DSCP_PRI_MAP3 |               | <a href="#">Section 14.5.6.12</a> |
| 40h    | P0_RX_DSCP_PRI_MAP4 |               | <a href="#">Section 14.5.6.13</a> |
| 44h    | P0_RX_DSCP_PRI_MAP5 |               | <a href="#">Section 14.5.6.14</a> |
| 48h    | P0_RX_DSCP_PRI_MAP6 |               | <a href="#">Section 14.5.6.15</a> |
| 4Ch    | P0_RX_DSCP_PRI_MAP7 |               | <a href="#">Section 14.5.6.16</a> |
| 100h   | P1_CONTROL          |               | <a href="#">Section 14.5.6.17</a> |
| 108h   | P1_MAX_BLKS         |               | <a href="#">Section 14.5.6.18</a> |
| 10Ch   | P1_BLK_CNT          |               | <a href="#">Section 14.5.6.19</a> |
| 110h   | P1_TX_IN_CTL        |               | <a href="#">Section 14.5.6.20</a> |

**Table 14-135. CPSW\_PORT REGISTERS (continued)**

| Offset | Acronym             | Register Name | Section                           |
|--------|---------------------|---------------|-----------------------------------|
| 114h   | P1_PORT_VLAN        |               | <a href="#">Section 14.5.6.21</a> |
| 118h   | P1_TX_PRI_MAP       |               | <a href="#">Section 14.5.6.22</a> |
| 11Ch   | P1_TS_SEQ_MTYPE     |               | <a href="#">Section 14.5.6.23</a> |
| 120h   | P1_SA_LO            |               | <a href="#">Section 14.5.6.24</a> |
| 124h   | P1_SA_HI            |               | <a href="#">Section 14.5.6.25</a> |
| 128h   | P1_SEND_PERCENT     |               | <a href="#">Section 14.5.6.26</a> |
| 130h   | P1_RX_DSCP_PRI_MAP0 |               | <a href="#">Section 14.5.6.27</a> |
| 134h   | P1_RX_DSCP_PRI_MAP1 |               | <a href="#">Section 14.5.6.28</a> |
| 138h   | P1_RX_DSCP_PRI_MAP2 |               | <a href="#">Section 14.5.6.29</a> |
| 13Ch   | P1_RX_DSCP_PRI_MAP3 |               | <a href="#">Section 14.5.6.30</a> |
| 140h   | P1_RX_DSCP_PRI_MAP4 |               | <a href="#">Section 14.5.6.31</a> |
| 144h   | P1_RX_DSCP_PRI_MAP5 |               | <a href="#">Section 14.5.6.32</a> |
| 148h   | P1_RX_DSCP_PRI_MAP6 |               | <a href="#">Section 14.5.6.33</a> |
| 14Ch   | P1_RX_DSCP_PRI_MAP7 |               | <a href="#">Section 14.5.6.34</a> |
| 200h   | P2_CONTROL          |               | <a href="#">Section 14.5.6.35</a> |
| 208h   | P2_MAX_BLKS         |               | <a href="#">Section 14.5.6.36</a> |
| 20Ch   | P2_BLK_CNT          |               | <a href="#">Section 14.5.6.37</a> |
| 210h   | P2_TX_IN_CTL        |               | <a href="#">Section 14.5.6.38</a> |
| 214h   | P2_PORT_VLAN        |               | <a href="#">Section 14.5.6.39</a> |
| 218h   | P2_TX_PRI_MAP       |               | <a href="#">Section 14.5.6.40</a> |
| 21Ch   | P2_TS_SEQ_MTYPE     |               | <a href="#">Section 14.5.6.41</a> |
| 220h   | P2_SA_LO            |               | <a href="#">Section 14.5.6.42</a> |
| 224h   | P2_SA_HI            |               | <a href="#">Section 14.5.6.43</a> |
| 228h   | P2_SEND_PERCENT     |               | <a href="#">Section 14.5.6.44</a> |
| 230h   | P2_RX_DSCP_PRI_MAP0 |               | <a href="#">Section 14.5.6.45</a> |
| 234h   | P2_RX_DSCP_PRI_MAP1 |               | <a href="#">Section 14.5.6.46</a> |
| 238h   | P2_RX_DSCP_PRI_MAP2 |               | <a href="#">Section 14.5.6.47</a> |
| 23Ch   | P2_RX_DSCP_PRI_MAP3 |               | <a href="#">Section 14.5.6.48</a> |
| 240h   | P2_RX_DSCP_PRI_MAP4 |               | <a href="#">Section 14.5.6.49</a> |
| 244h   | P2_RX_DSCP_PRI_MAP5 |               | <a href="#">Section 14.5.6.50</a> |
| 248h   | P2_RX_DSCP_PRI_MAP6 |               | <a href="#">Section 14.5.6.51</a> |
| 24Ch   | P2_RX_DSCP_PRI_MAP7 |               | <a href="#">Section 14.5.6.52</a> |

#### 14.5.6.1 P0\_CONTROL Register (offset = 0h) [reset = 0h]

P0\_CONTROL is shown in [Figure 14-121](#) and described in [Table 14-136](#).

##### CPSW PORT 0 CONTROL REGISTER

**Figure 14-121. P0\_CONTROL Register**

| 31       | 30 | 29                | 28                | 27       | 26       | 25 | 24                 |
|----------|----|-------------------|-------------------|----------|----------|----|--------------------|
| Reserved |    | P0_DLR_CPDMA_CH   |                   |          | Reserved |    | P0_PASS_PRI_TAGGED |
| R/W-0h   |    |                   |                   |          |          |    | R/W-0h             |
| 23       | 22 | 21                | 20                | 19       | 18       | 17 | 16                 |
| Reserved |    | P0_VLAN_LTYPE2_EN | P0_VLAN_LTYPE1_EN | Reserved |          |    | P0_DSCP_PRI_EN     |
| R/W-0h   |    |                   | R/W-0h            |          |          |    |                    |
| 15       | 14 | 13                | 12                | 11       | 10       | 9  | 8                  |
| Reserved |    |                   |                   |          |          |    |                    |
| 7        | 6  | 5                 | 4                 | 3        | 2        | 1  | 0                  |
| Reserved |    |                   |                   |          |          |    |                    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-136. P0\_CONTROL Register Field Descriptions**

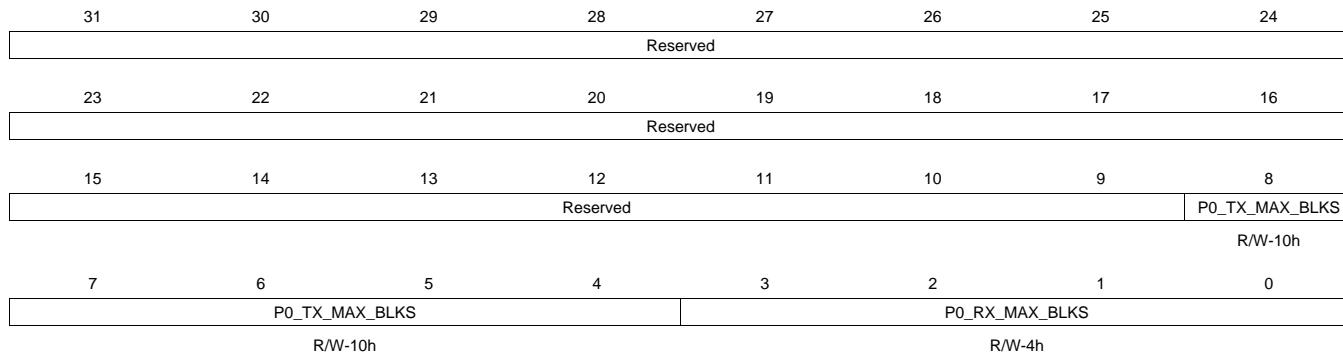
| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 30-28 | P0_DLR_CPDMA_CH    | R/W  | 0h    | Port 0 DLR CPDMA Channel This field indicates the CPDMA channel that DLR packets will be received on.   |
| 24    | P0_PASS_PRI_TAGGED | R/W  | 0h    | Port 0 Pass Priority Tagged<br>0 - Priority tagged packets have the zero VID replaced with the input port P0_PORT_VLAN [11:0]<br>1 - Priority tagged packets are processed unchanged.   |
| 21    | P0_VLAN_LTYPE2_EN  | R/W  | 0h    | Port 0 VLAN LTYPE 2 enable<br>0 - disabled<br>1 - enabled   |
| 20    | P0_VLAN_LTYPE1_EN  | R/W  | 0h    | Port 0 VLAN LTYPE 1 enable<br>0 - disabled<br>1 - enabled   |
| 16    | P0_DSCP_PRI_EN     | R/W  | 0h    | Port 0 DSCP Priority Enable<br>0 - DSCP priority disabled<br>1 - DSCP priority enabled.<br>All non-tagged IPV4 packets have their received packet priority determined by mapping the 6 TOS bits through the port DSCP priority mapping registers. |

#### 14.5.6.2 P0\_MAX\_BLKS Register (offset = 8h) [reset = 104h]

P0\_MAX\_BLKS is shown in [Figure 14-122](#) and described in [Table 14-137](#).

##### CPSW PORT 0 MAXIMUM FIFO BLOCKS REGISTER

**Figure 14-122. P0\_MAX\_BLKS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-137. P0\_MAX\_BLKS Register Field Descriptions**

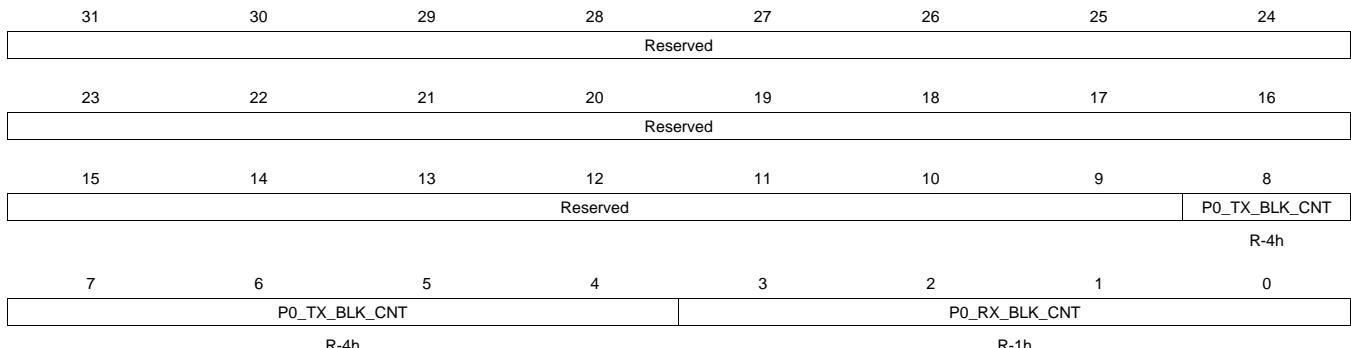
| Bit | Field          | Type | Reset | Description  |
|-----|----------------|------|-------|--|
| 8-4 | P0_TX_MAX_BLKS | R/W  | 10h   | Transmit FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical transmit priority queues.<br>0x10 is the recommended value of p0_tx_max_blks.<br>Port 0 should remain in flow control mode.<br>0xe is the minimum value tx max blks. |
| 3-0 | P0_RX_MAX_BLKS | R/W  | 4h    | Receive FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical receive queue.<br>0x4 is the recommended value.<br>0x3 is the minimum value rx max blks and 0x6 is the maximum value.   |

#### 14.5.6.3 P0\_BLK\_CNT Register (offset = Ch) [reset = 41h]

P0\_BLK\_CNT is shown in [Figure 14-123](#) and described in [Table 14-138](#).

CPSW PORT 0 FIFO BLOCK USAGE COUNT (READ ONLY)

**Figure 14-123. P0\_BLK\_CNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-138. P0\_BLK\_CNT Register Field Descriptions**

| Bit | Field         | Type | Reset | Description   |
|-----|---------------|------|-------|---|
| 8-4 | P0_TX_BLK_CNT | R    | 4h    | Port 0 Transmit Block Count Usage - This value is the number of blocks allocated to the FIFO logical transmit queues. |
| 3-0 | P0_RX_BLK_CNT | R    | 1h    | Port 0 Receive Block Count Usage - This value is the number of blocks allocated to the FIFO logical receive queues.   |

#### 14.5.6.4 P0\_TX\_IN\_CTL Register (offset = 10h) [reset = 40C0h]

P0\_TX\_IN\_CTL is shown in Figure 14-124 and described in Table 14-139.

CPSW PORT 0 TRANSMIT FIFO CONTROL

**Figure 14-124. P0\_TX\_IN\_CTL Register**

| 31          | 30 | 29 | 28 | 27       | 26 | 25         | 24 |
|-------------|----|----|----|----------|----|------------|----|
| Reserved    |    |    |    |          |    |            |    |
| 23          | 22 | 21 | 20 | 19       | 18 | 17         | 16 |
| TX_RATE_EN  |    |    |    | Reserved |    | TX_IN_SEL  |    |
| R/W-0h      |    |    |    | R/W-0h   |    | R/W-0h     |    |
| 15          | 14 | 13 | 12 | 11       | 10 | 9          | 8  |
| TX_BLKS_Rem |    |    |    | Reserved |    | TX_PRI_WDS |    |
| R/W-4h      |    |    |    | R/W-C0h  |    | R/W-C0h    |    |
| 7           | 6  | 5  | 4  | 3        | 2  | 1          | 0  |
| TX_PRI_WDS  |    |    |    |          |    |            |    |
| R/W-C0h     |    |    |    |          |    |            |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-139. P0\_TX\_IN\_CTL Register Field Descriptions**

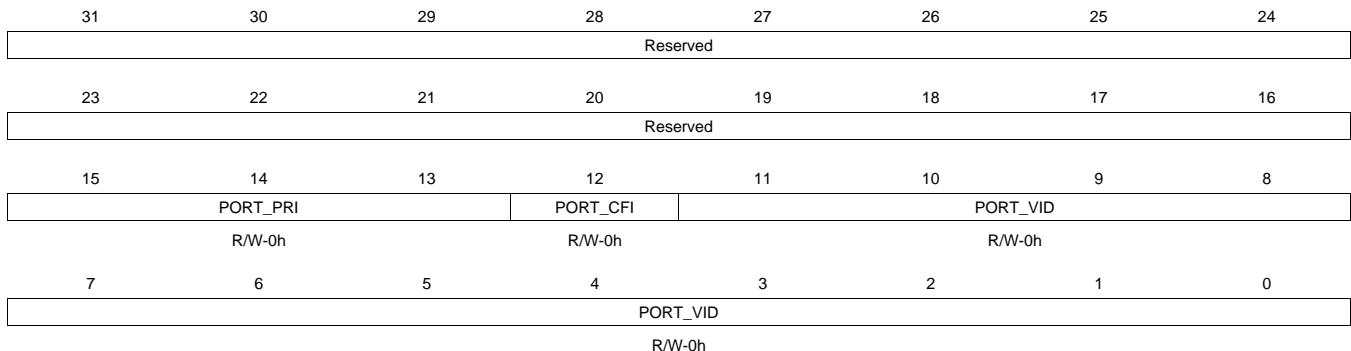
| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 23-20 | TX_RATE_EN  | R/W  | 0h    | Transmit FIFO Input Rate Enable  |
| 17-16 | TX_IN_SEL   | R/W  | 0h    | Transmit FIFO Input Queue Type Select<br>00 - Normal priority mode<br>01 - Dual MAC mode<br>10 - Rate Limit mode<br>11 - reserved Note that Dual MAC mode is not compatible with escalation or shaping because dual mac mode forces round robin priority on FIFO egress. |
| 15-12 | TX_BLKS_Rem | R/W  | 4h    | Transmit FIFO Input Blocks to subtract in dual mac mode  |
| 9-0   | TX_PRI_WDS  | R/W  | C0h   | Transmit FIFO Words in queue   |

#### 14.5.6.5 P0\_PORT\_VLAN Register (offset = 14h) [reset = 0h]

P0\_PORT\_VLAN is shown in [Figure 14-125](#) and described in [Table 14-140](#).

##### CPSW PORT 0 VLAN REGISTER

**Figure 14-125. P0\_PORT\_VLAN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-140. P0\_PORT\_VLAN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                                |
|-------|----------|------|-------|--|
| 15-13 | PORT_PRI | R/W  | 0h    | Port VLAN Priority (7 is highest priority) |
| 12    | PORT_CFI | R/W  | 0h    | Port CFI bit                               |
| 11-0  | PORT_VID | R/W  | 0h    | Port VLAN ID                               |

#### 14.5.6.6 P0\_TX\_PRI\_MAP Register (offset = 18h) [reset = 33221001h]

P0\_TX\_PRI\_MAP is shown in [Figure 14-126](#) and described in [Table 14-141](#).

CPSW PORT 0 TX HEADER PRI TO SWITCH PRI MAPPING REGISTER

**Figure 14-126. P0\_TX\_PRI\_MAP Register**

| 31       | 30 | 29   | 28     | 27       | 26 | 25   | 24     |
|----------|----|------|--------|----------|----|------|--------|
| Reserved |    | PRI7 |        | Reserved |    | PRI6 |        |
|          |    |      | R/W-3h |          |    |      | R/W-3h |
| 23       | 22 | 21   | 20     | 19       | 18 | 17   | 16     |
| Reserved |    | PRI5 |        | Reserved |    | PRI4 |        |
|          |    |      | R/W-2h |          |    |      | R/W-2h |
| 15       | 14 | 13   | 12     | 11       | 10 | 9    | 8      |
| Reserved |    | PRI3 |        | Reserved |    | PRI2 |        |
|          |    |      | R/W-1h |          |    |      | R/W-0h |
| 7        | 6  | 5    | 4      | 3        | 2  | 1    | 0      |
| Reserved |    | PRI1 |        | Reserved |    | PRI0 |        |
|          |    |      | R/W-0h |          |    |      | R/W-1h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-141. P0\_TX\_PRI\_MAP Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 29-28 | PRI7  | R/W  | 3h    | Priority<br>7 - A packet header priority of 0x7 is given this switch queue pri. |
| 25-24 | PRI6  | R/W  | 3h    | Priority<br>6 - A packet header priority of 0x6 is given this switch queue pri. |
| 21-20 | PRI5  | R/W  | 2h    | Priority<br>5 - A packet header priority of 0x5 is given this switch queue pri. |
| 17-16 | PRI4  | R/W  | 2h    | Priority<br>4 - A packet header priority of 0x4 is given this switch queue pri. |
| 13-12 | PRI3  | R/W  | 1h    | Priority<br>3 - A packet header priority of 0x3 is given this switch queue pri. |
| 9-8   | PRI2  | R/W  | 0h    | Priority<br>2 - A packet header priority of 0x2 is given this switch queue pri. |
| 5-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet header priority of 0x1 is given this switch queue pri. |
| 1-0   | PRI0  | R/W  | 1h    | Priority<br>0 - A packet header priority of 0x0 is given this switch queue pri. |

#### 14.5.6.7 P0\_CPDMA\_TX\_PRI\_MAP Register (offset = 1Ch) [reset = 76543210h]

P0\_CPDMA\_TX\_PRI\_MAP is shown in [Figure 14-127](#) and described in [Table 14-142](#).

CPSW CPDMA TX (PORT 0 RX) PKT PRIORITY TO HEADER PRIORITY

**Figure 14-127. P0\_CPDMA\_TX\_PRI\_MAP Register**

|          |    |        |    |          |    |        |    |
|----------|----|--------|----|----------|----|--------|----|
| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
| Reserved |    | PRI7   |    | Reserved |    | PRI6   |    |
|          |    | R/W-7h |    |          |    | R/W-6h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI5   |    | Reserved |    | PRI4   |    |
|          |    | R/W-5h |    |          |    | R/W-4h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI3   |    | Reserved |    | PRI2   |    |
|          |    | R/W-3h |    |          |    | R/W-2h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI1   |    | Reserved |    | PRI0   |    |
|          |    | R/W-1h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-142. P0\_CPDMA\_TX\_PRI\_MAP Register Field Descriptions**

| Bit   | Field | Type | Reset | Description  |
|-------|-------|------|-------|--|
| 30-28 | PRI7  | R/W  | 7h    | Priority<br>7 - A packet pri of 0x7 is mapped (changed) to this header packet pri. |
| 26-24 | PRI6  | R/W  | 6h    | Priority<br>6 - A packet pri of 0x6 is mapped (changed) to this header packet pri. |
| 22-20 | PRI5  | R/W  | 5h    | Priority<br>5 - A packet pri of 0x5 is mapped (changed) to this header packet pri. |
| 18-16 | PRI4  | R/W  | 4h    | Priority<br>4 - A packet pri of 0x4 is mapped (changed) to this header packet pri. |
| 14-12 | PRI3  | R/W  | 3h    | Priority<br>3 - A packet pri of 0x3 is mapped (changed) to this header packet pri. |
| 10-8  | PRI2  | R/W  | 2h    | Priority<br>2 - A packet pri of 0x2 is mapped (changed) to this header packet pri. |
| 6-4   | PRI1  | R/W  | 1h    | Priority<br>1 - A packet pri of 0x1 is mapped (changed) to this header packet pri. |
| 2-0   | PRI0  | R/W  | 0h    | Priority<br>0 - A packet pri of 0x0 is mapped (changed) to this header packet pri. |

#### 14.5.6.8 P0\_CPDMA\_RX\_CH\_MAP Register (offset = 20h) [reset = 0h]

P0\_CPDMA\_RX\_CH\_MAP is shown in [Figure 14-128](#) and described in [Table 14-143](#).

CPSW CPDMA RX (PORT 0 TX) SWITCH PRIORITY TO DMA CHANNEL

**Figure 14-128. P0\_CPDMA\_RX\_CH\_MAP Register**

| 31       | 30 | 29      | 28 | 27       | 26 | 25      | 24 |
|----------|----|---------|----|----------|----|---------|----|
| Reserved |    | P2_PRI3 |    | Reserved |    | P2_PRI2 |    |
|          |    | R/W-0h  |    |          |    | R/W-0h  |    |
| 23       | 22 | 21      | 20 | 19       | 18 | 17      | 16 |
| Reserved |    | P2_PRI1 |    | Reserved |    | P2_PRI0 |    |
|          |    | R/W-0h  |    |          |    | R/W-0h  |    |
| 15       | 14 | 13      | 12 | 11       | 10 | 9       | 8  |
| Reserved |    | P1_PRI3 |    | Reserved |    | P1_PRI2 |    |
|          |    | R/W-0h  |    |          |    | R/W-0h  |    |
| 7        | 6  | 5       | 4  | 3        | 2  | 1       | 0  |
| Reserved |    | P1_PRI1 |    | Reserved |    | P1_PRI0 |    |
|          |    | R/W-0h  |    |          |    | R/W-0h  |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-143. P0\_CPDMA\_RX\_CH\_MAP Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description   |
|-------|---------|------|-------|---|
| 30-28 | P2_PRI3 | R/W  | 0h    | Port 2 Priority 3 packets go to this CPDMA Rx Channel |
| 26-24 | P2_PRI2 | R/W  | 0h    | Port 2 Priority 2 packets go to this CPDMA Rx Channel |
| 22-20 | P2_PRI1 | R/W  | 0h    | Port 2 Priority 1 packets go to this CPDMA Rx Channel |
| 18-16 | P2_PRI0 | R/W  | 0h    | Port 2 Priority 0 packets go to this CPDMA Rx Channel |
| 14-12 | P1_PRI3 | R/W  | 0h    | Port 1 Priority 3 packets go to this CPDMA Rx Channel |
| 10-8  | P1_PRI2 | R/W  | 0h    | Port 1 Priority 2 packets go to this CPDMA Rx Channel |
| 6-4   | P1_PRI1 | R/W  | 0h    | Port 1 Priority 1 packets go to this CPDMA Rx Channel |
| 2-0   | P1_PRI0 | R/W  | 0h    | Port 1 Priority 0 packets go to this CPDMA Rx Channel |

**14.5.6.9 P0\_RX\_DSCP\_PRI\_MAP0 Register (offset = 30h) [reset = 0h]**

P0\_RX\_DSCP\_PRI\_MAP0 is shown in [Figure 14-129](#) and described in [Table 14-144](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0

**Figure 14-129. P0\_RX\_DSCP\_PRI\_MAP0 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI7   |    | Reserved |    | PRI6   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI5   |    | Reserved |    | PRI4   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI3   |    | Reserved |    | PRI2   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI1   |    | Reserved |    | PRI0   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-144. P0\_RX\_DSCP\_PRI\_MAP0 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI7  | R/W  | 0h    | Priority<br>7 - A packet TOS of 0d7 is mapped to this received packet priority. |
| 26-24 | PRI6  | R/W  | 0h    | Priority<br>6 - A packet TOS of 0d6 is mapped to this received packet priority. |
| 22-20 | PRI5  | R/W  | 0h    | Priority<br>5 - A packet TOS of 0d5 is mapped to this received packet priority. |
| 18-16 | PRI4  | R/W  | 0h    | Priority<br>4 - A packet TOS of 0d4 is mapped to this received packet priority. |
| 14-12 | PRI3  | R/W  | 0h    | Priority<br>3 - A packet TOS of 0d3 is mapped to this received packet priority. |
| 10-8  | PRI2  | R/W  | 0h    | Priority<br>2 - A packet TOS of 0d2 is mapped to this received packet priority. |
| 6-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet TOS of 0d1 is mapped to this received packet priority. |
| 2-0   | PRI0  | R/W  | 0h    | Priority<br>0 - A packet TOS of 0d0 is mapped to this received packet priority. |

#### 14.5.6.10 P0\_RX\_DSCP\_PRI\_MAP1 Register (offset = 34h) [reset = 0h]

P0\_RX\_DSCP\_PRI\_MAP1 is shown in [Figure 14-130](#) and described in [Table 14-145](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1

**Figure 14-130. P0\_RX\_DSCP\_PRI\_MAP1 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI15  |    | Reserved |    | PRI14  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI13  |    | Reserved |    | PRI12  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI11  |    | Reserved |    | PRI10  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI9   |    | Reserved |    | PRI8   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-145. P0\_RX\_DSCP\_PRI\_MAP1 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI15 | R/W  | 0h    | Priority<br>15 - A packet TOS of 0d15 is mapped to this received packet priority. |
| 26-24 | PRI14 | R/W  | 0h    | Priority<br>14 - A packet TOS of 0d14 is mapped to this received packet priority. |
| 22-20 | PRI13 | R/W  | 0h    | Priority<br>13 - A packet TOS of 0d13 is mapped to this received packet priority. |
| 18-16 | PRI12 | R/W  | 0h    | Priority<br>12 - A packet TOS of 0d12 is mapped to this received packet priority. |
| 14-12 | PRI11 | R/W  | 0h    | Priority<br>11 - A packet TOS of 0d11 is mapped to this received packet priority. |
| 10-8  | PRI10 | R/W  | 0h    | Priority<br>10 - A packet TOS of 0d10 is mapped to this received packet priority. |
| 6-4   | PRI9  | R/W  | 0h    | Priority<br>9 - A packet TOS of 0d9 is mapped to this received packet priority.   |
| 2-0   | PRI8  | R/W  | 0h    | Priority<br>8 - A packet TOS of 0d8 is mapped to this received packet priority.   |

**14.5.6.11 P0\_RX\_DSCP\_PRI\_MAP2 Register (offset = 38h) [reset = 0h]**

P0\_RX\_DSCP\_PRI\_MAP2 is shown in [Figure 14-131](#) and described in [Table 14-146](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2

**Figure 14-131. P0\_RX\_DSCP\_PRI\_MAP2 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI23  |    | Reserved |    | PRI22  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI21  |    | Reserved |    | PRI20  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI19  |    | Reserved |    | PRI18  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI17  |    | Reserved |    | PRI16  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-146. P0\_RX\_DSCP\_PRI\_MAP2 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI23 | R/W  | 0h    | Priority<br>23 - A packet TOS of 0d23 is mapped to this received packet priority. |
| 26-24 | PRI22 | R/W  | 0h    | Priority<br>22 - A packet TOS of 0d22 is mapped to this received packet priority. |
| 22-20 | PRI21 | R/W  | 0h    | Priority<br>21 - A packet TOS of 0d21 is mapped to this received packet priority. |
| 18-16 | PRI20 | R/W  | 0h    | Priority<br>20 - A packet TOS of 0d20 is mapped to this received packet priority. |
| 14-12 | PRI19 | R/W  | 0h    | Priority<br>19 - A packet TOS of 0d19 is mapped to this received packet priority. |
| 10-8  | PRI18 | R/W  | 0h    | Priority<br>18 - A packet TOS of 0d18 is mapped to this received packet priority. |
| 6-4   | PRI17 | R/W  | 0h    | Priority<br>17 - A packet TOS of 0d17 is mapped to this received packet priority. |
| 2-0   | PRI16 | R/W  | 0h    | Priority<br>16 - A packet TOS of 0d16 is mapped to this received packet priority. |

#### 14.5.6.12 P0\_RX\_DSCP\_PRI\_MAP3 Register (offset = 3Ch) [reset = 0h]

P0\_RX\_DSCP\_PRI\_MAP3 is shown in [Figure 14-132](#) and described in [Table 14-147](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3

**Figure 14-132. P0\_RX\_DSCP\_PRI\_MAP3 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI31  |    | Reserved |    | PRI30  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI29  |    | Reserved |    | PRI28  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI27  |    | Reserved |    | PRI26  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI25  |    | Reserved |    | PRI24  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-147. P0\_RX\_DSCP\_PRI\_MAP3 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI31 | R/W  | 0h    | Priority<br>31 - A packet TOS of 0d31 is mapped to this received packet priority. |
| 26-24 | PRI30 | R/W  | 0h    | Priority<br>30 - A packet TOS of 0d30 is mapped to this received packet priority. |
| 22-20 | PRI29 | R/W  | 0h    | Priority<br>29 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 18-16 | PRI28 | R/W  | 0h    | Priority<br>28 - A packet TOS of 0d28 is mapped to this received packet priority. |
| 14-12 | PRI27 | R/W  | 0h    | Priority<br>27 - A packet TOS of 0d27 is mapped to this received packet priority. |
| 10-8  | PRI26 | R/W  | 0h    | Priority<br>26 - A packet TOS of 0d26 is mapped to this received packet priority. |
| 6-4   | PRI25 | R/W  | 0h    | Priority<br>25 - A packet TOS of 0d25 is mapped to this received packet priority. |
| 2-0   | PRI24 | R/W  | 0h    | Priority<br>24 - A packet TOS of 0d24 is mapped to this received packet priority. |

**14.5.6.13 P0\_RX\_DSCP\_PRI\_MAP4 Register (offset = 40h) [reset = 0h]**

P0\_RX\_DSCP\_PRI\_MAP4 is shown in [Figure 14-133](#) and described in [Table 14-148](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4

**Figure 14-133. P0\_RX\_DSCP\_PRI\_MAP4 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI39  |    | Reserved |    | PRI38  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI37  |    | Reserved |    | PRI36  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI35  |    | Reserved |    | PRI34  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI33  |    | Reserved |    | PRI32  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-148. P0\_RX\_DSCP\_PRI\_MAP4 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI39 | R/W  | 0h    | Priority<br>39 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 26-24 | PRI38 | R/W  | 0h    | Priority<br>38 - A packet TOS of 0d38 is mapped to this received packet priority. |
| 22-20 | PRI37 | R/W  | 0h    | Priority<br>37 - A packet TOS of 0d37 is mapped to this received packet priority. |
| 18-16 | PRI36 | R/W  | 0h    | Priority<br>36 - A packet TOS of 0d36 is mapped to this received packet priority. |
| 14-12 | PRI35 | R/W  | 0h    | Priority<br>35 - A packet TOS of 0d35 is mapped to this received packet priority. |
| 10-8  | PRI34 | R/W  | 0h    | Priority<br>34 - A packet TOS of 0d34 is mapped to this received packet priority. |
| 6-4   | PRI33 | R/W  | 0h    | Priority<br>33 - A packet TOS of 0d33 is mapped to this received packet priority. |
| 2-0   | PRI32 | R/W  | 0h    | Priority<br>32 - A packet TOS of 0d32 is mapped to this received packet priority. |

#### 14.5.6.14 P0\_RX\_DSCP\_PRI\_MAP5 Register (offset = 44h) [reset = 0h]

P0\_RX\_DSCP\_PRI\_MAP5 is shown in [Figure 14-134](#) and described in [Table 14-149](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5

**Figure 14-134. P0\_RX\_DSCP\_PRI\_MAP5 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI47  |    | Reserved |    | PRI46  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI45  |    | Reserved |    | PRI44  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI43  |    | Reserved |    | PRI42  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI41  |    | Reserved |    | PRI40  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-149. P0\_RX\_DSCP\_PRI\_MAP5 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI47 | R/W  | 0h    | Priority<br>47 - A packet TOS of 0d47 is mapped to this received packet priority. |
| 26-24 | PRI46 | R/W  | 0h    | Priority<br>46 - A packet TOS of 0d46 is mapped to this received packet priority. |
| 22-20 | PRI45 | R/W  | 0h    | Priority<br>45 - A packet TOS of 0d45 is mapped to this received packet priority. |
| 18-16 | PRI44 | R/W  | 0h    | Priority<br>44 - A packet TOS of 0d44 is mapped to this received packet priority. |
| 14-12 | PRI43 | R/W  | 0h    | Priority<br>43 - A packet TOS of 0d43 is mapped to this received packet priority. |
| 10-8  | PRI42 | R/W  | 0h    | Priority<br>42 - A packet TOS of 0d42 is mapped to this received packet priority. |
| 6-4   | PRI41 | R/W  | 0h    | Priority<br>41 - A packet TOS of 0d41 is mapped to this received packet priority. |
| 2-0   | PRI40 | R/W  | 0h    | Priority<br>40 - A packet TOS of 0d40 is mapped to this received packet priority. |

**14.5.6.15 P0\_RX\_DSCP\_PRI\_MAP6 Register (offset = 48h) [reset = 0h]**

P0\_RX\_DSCP\_PRI\_MAP6 is shown in [Figure 14-135](#) and described in [Table 14-150](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6

**Figure 14-135. P0\_RX\_DSCP\_PRI\_MAP6 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI55  |    | Reserved |    | PRI54  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI53  |    | Reserved |    | PRI52  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI51  |    | Reserved |    | PRI50  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI49  |    | Reserved |    | PRI48  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-150. P0\_RX\_DSCP\_PRI\_MAP6 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI55 | R/W  | 0h    | Priority<br>55 - A packet TOS of 0d55 is mapped to this received packet priority. |
| 26-24 | PRI54 | R/W  | 0h    | Priority<br>54 - A packet TOS of 0d54 is mapped to this received packet priority. |
| 22-20 | PRI53 | R/W  | 0h    | Priority<br>53 - A packet TOS of 0d53 is mapped to this received packet priority. |
| 18-16 | PRI52 | R/W  | 0h    | Priority<br>52 - A packet TOS of 0d52 is mapped to this received packet priority. |
| 14-12 | PRI51 | R/W  | 0h    | Priority<br>51 - A packet TOS of 0d51 is mapped to this received packet priority. |
| 10-8  | PRI50 | R/W  | 0h    | Priority<br>50 - A packet TOS of 0d50 is mapped to this received packet priority. |
| 6-4   | PRI49 | R/W  | 0h    | Priority<br>49 - A packet TOS of 0d49 is mapped to this received packet priority. |
| 2-0   | PRI48 | R/W  | 0h    | Priority<br>48 - A packet TOS of 0d48 is mapped to this received packet priority. |

#### 14.5.6.16 P0\_RX\_DSCP\_PRI\_MAP7 Register (offset = 4Ch) [reset = 0h]

P0\_RX\_DSCP\_PRI\_MAP7 is shown in [Figure 14-136](#) and described in [Table 14-151](#).

CPSW PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7

**Figure 14-136. P0\_RX\_DSCP\_PRI\_MAP7 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI63  |    | Reserved |    | PRI62  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI61  |    | Reserved |    | PRI60  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI59  |    | Reserved |    | PRI58  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI57  |    | Reserved |    | PRI56  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-151. P0\_RX\_DSCP\_PRI\_MAP7 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI63 | R/W  | 0h    | Priority<br>63 - A packet TOS of 0d63 is mapped to this received packet priority. |
| 26-24 | PRI62 | R/W  | 0h    | Priority<br>62 - A packet TOS of 0d62 is mapped to this received packet priority. |
| 22-20 | PRI61 | R/W  | 0h    | Priority<br>61 - A packet TOS of 0d61 is mapped to this received packet priority. |
| 18-16 | PRI60 | R/W  | 0h    | Priority<br>60 - A packet TOS of 0d60 is mapped to this received packet priority. |
| 14-12 | PRI59 | R/W  | 0h    | Priority<br>59 - A packet TOS of 0d59 is mapped to this received packet priority. |
| 10-8  | PRI58 | R/W  | 0h    | Priority<br>58 - A packet TOS of 0d58 is mapped to this received packet priority. |
| 6-4   | PRI57 | R/W  | 0h    | Priority<br>57 - A packet TOS of 0d57 is mapped to this received packet priority. |
| 2-0   | PRI56 | R/W  | 0h    | Priority<br>56 - A packet TOS of 0d56 is mapped to this received packet priority. |

#### 14.5.6.17 P1\_CONTROL Register (offset = 100h) [reset = 0h]

P1\_CONTROL is shown in [Figure 14-137](#) and described in [Table 14-152](#).

##### CPSW PORT 1 CONTROL REGISTER

**Figure 14-137. P1\_CONTROL Register**

|                                |           |                   |                   |                 |                 |             |                    |  |
|--------------------------------|-----------|-------------------|-------------------|-----------------|-----------------|-------------|--------------------|--|
| 31                             | 30        | 29                | 28                | 27              | 26              | 25          | 24                 |  |
| Reserved                       |           |                   |                   |                 |                 |             | P1_PASS_PRI_TAGGED |  |
| R/W-0h                         |           |                   |                   |                 |                 |             |                    |  |
| 23                             | 22        | 21                | 20                | 19              | 18              | 17          | 16                 |  |
| Reserved                       |           | P1_VLAN_LTYPE2_EN | P1_VLAN_LTYPE1_EN | Reserved        |                 |             | P1_DSCP_PRI_EN     |  |
| R/W-0h      R/W-0h      R/W-0h |           |                   |                   |                 |                 |             |                    |  |
| 15                             | 14        | 13                | 12                | 11              | 10              | 9           | 8                  |  |
| Reserved                       | P1_TS_320 | P1_TS_319         | P1_TS_132         | P1_TS_131       | P1_TS_130       | P1_TS_129   | P1_TS_TTL_NONZERO  |  |
| R/W-0h                         |           | R/W-0h            |                   | R/W-0h          |                 | R/W-0h      |                    |  |
| 7                              | 6         | 5                 | 4                 | 3               | 2               | 1           | 0                  |  |
| Reserved                       |           |                   | P1_TS_ANNEC_D_EN  | P1_TS_LTYPE2_EN | P1_TS_LTYPE1_EN | P1_TS_TX_EN | P1_TS_RX_EN        |  |
| R/W-0h                         |           |                   | R-0h              |                 | R/W-0h          |             | R/W-0h             |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-152. P1\_CONTROL Register Field Descriptions**

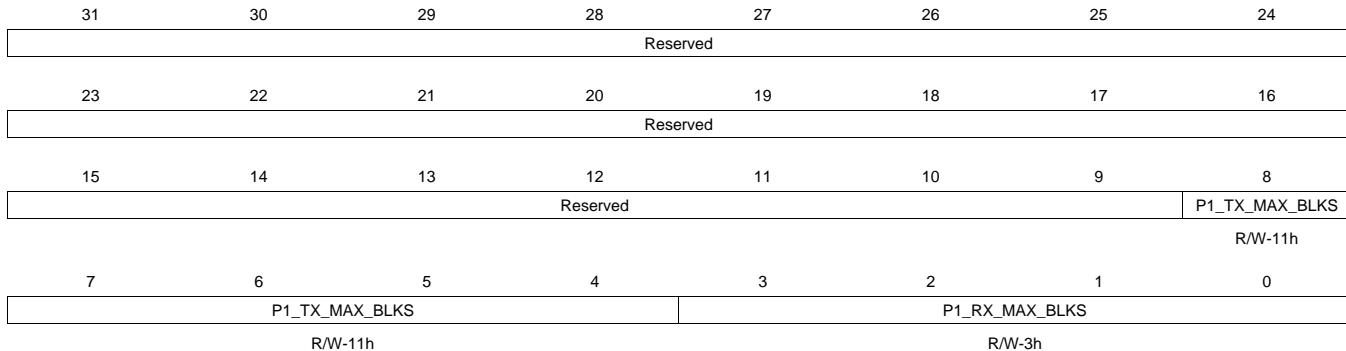
| Bit | Field              | Type | Reset | Description   |
|-----|--------------------|------|-------|---|
| 24  | P1_PASS_PRI_TAGGED | R/W  | 0h    | Port 1 Pass Priority Tagged<br>0 - Priority tagged packets have the zero VID replaced with the input port P1_PORT_VLAN [11:0]<br>1 - Priority tagged packets are processed unchanged.   |
| 21  | P1_VLAN_LTYPE2_EN  | R/W  | 0h    | Port 1 VLAN LTYPE 2 enable<br>0 - disabled<br>1 - VLAN LTYPE2 enabled on transmit and receive   |
| 20  | P1_VLAN_LTYPE1_EN  | R/W  | 0h    | Port 1 VLAN LTYPE 1 enable<br>0 - disabled<br>1 - VLAN LTYPE1 enabled on transmit and receive   |
| 16  | P1_DSCP_PRI_EN     | R/W  | 0h    | Port 1 DSCP Priority Enable<br>0 - DSCP priority disabled<br>1 - DSCP priority enabled.<br>All non-tagged IPv4 packets have their received packet priority determined by mapping the 6 TOS bits through the port DSCP priority mapping registers. |
| 14  | P1_TS_320          | R/W  | 0h    | Port 1 Time Sync Destination Port Number 320 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination port number 320 (decimal) is enabled.  |
| 13  | P1_TS_319          | R/W  | 0h    | Port 1 Time Sync Destination Port Number 319 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination port number 319 (decimal) is enabled.  |
| 12  | P1_TS_132          | R/W  | 0h    | Port 1 Time Sync Destination IP Address 132 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 132 (decimal) is enabled.   |
| 11  | P1_TS_131          | R/W  | 0h    | Port 1 Time Sync Destination IP Address 131 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 131 (decimal) is enabled.   |

**Table 14-152. P1\_CONTROL Register Field Descriptions (continued)**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 10  | P1_TS_130         | R/W  | 0h    | Port 1 Time Sync Destination IP Address 130 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 130 (decimal) is enabled. |
| 9   | P1_TS_129         | R/W  | 0h    | Port 1 Time Sync Destination IP Address 129 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 129 (decimal) is enabled. |
| 8   | P1_TS_TTL_NONZERO | R/W  | 0h    | Port 1 Time Sync Time To Live Non-zero enable.<br>0 = TTL must be zero.<br>1 = TTL may be any value.  |
| 4   | P1_TS_ANNEX_D_EN  | R/W  | 0h    | Port 1 Time Sync Annex D enable<br>0 - Annex D disabled<br>1 - Annex D enabled  |
| 3   | P1_TS_LTYPE2_EN   | R    | 0h    | Port 1 Time Sync LTYPE 2 enable<br>0 - disabled<br>1 - enabled  |
| 2   | P1_TS_LTYPE1_EN   | R/W  | 0h    | Port 1 Time Sync LTYPE 1 enable<br>0 - disabled<br>1 - enabled  |
| 1   | P1_TS_TX_EN       | R/W  | 0h    | Port 1 Time Sync Transmit Enable<br>0 - disabled<br>1 - enabled   |
| 0   | P1_TS_RX_EN       | R/W  | 0h    | Port 1 Time Sync Receive Enable<br>0 - Port 1 Receive Time Sync disabled<br>1 - Port 1 Receive Time Sync enabled  |

**14.5.6.18 P1\_MAX\_BLKS Register (offset = 108h) [reset = 113h]**

P1\_MAX\_BLKS is shown in [Figure 14-138](#) and described in [Table 14-153](#).

**CPSW PORT 1 MAXIMUM FIFO BLOCKS REGISTER**
**Figure 14-138. P1\_MAX\_BLKS Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-153. P1\_MAX\_BLKS Register Field Descriptions**

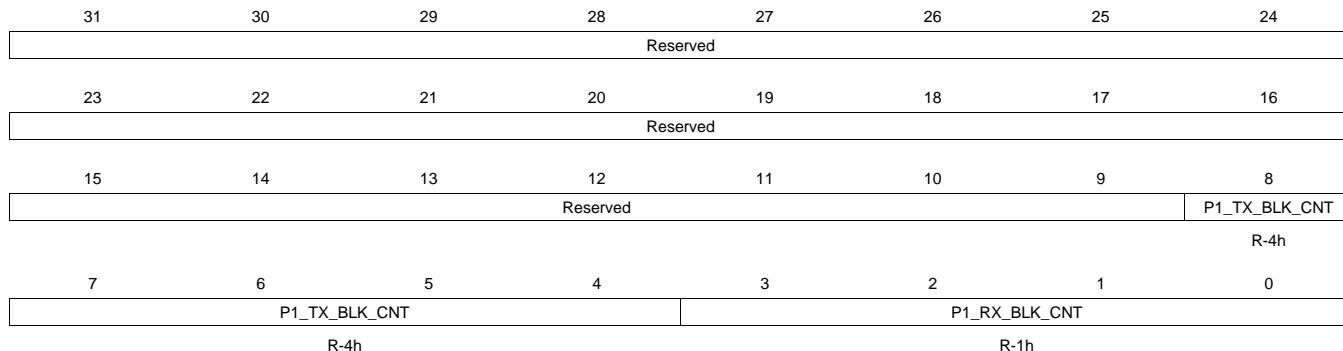
| Bit | Field          | Type | Reset | Description   |
|-----|----------------|------|-------|---|
| 8-4 | P1_TX_MAX_BLKS | R/W  | 11h   | <p>Transmit FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical transmit priority queues.</p> <p>0x11 is the recommended value of p1_tx_max_blks unless the port is in fullduplex flow control mode.</p> <p>In flow control mode, the p1_rx_max_blks will need to increase in order to accept the required run out in fullduplex mode.</p> <p>This value will need to decrease by the amount of increase in p1_rx_max_blks.</p> <p>0xe is the minimum value tx max blks.</p> |
| 3-0 | P1_RX_MAX_BLKS | R/W  | 3h    | <p>Receive FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical receive queue.</p> <p>This value must be greater than or equal to 0x3.</p> <p>It should be increased in fullduplex flow control mode to 0x5 or 0x6 depending on the required runout space.</p> <p>The p1_tx_max_blks value must be decreased by the amount of increase in p1_rx_max_blks.</p> <p>0x3 is the minimum value rx max blks and 0x6 is the maximum value.</p>                                       |

#### 14.5.6.19 P1\_BLK\_CNT Register (offset = 10Ch) [reset = 41h]

P1\_BLK\_CNT is shown in [Figure 14-139](#) and described in [Table 14-154](#).

CPSW PORT 1 FIFO BLOCK USAGE COUNT (READ ONLY)

**Figure 14-139. P1\_BLK\_CNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-154. P1\_BLK\_CNT Register Field Descriptions**

| Bit | Field         | Type | Reset | Description   |
|-----|---------------|------|-------|---|
| 8-4 | P1_TX_BLK_CNT | R    | 4h    | Port 1 Transmit Block Count Usage - This value is the number of blocks allocated to the FIFO logical transmit queues. |
| 3-0 | P1_RX_BLK_CNT | R    | 1h    | Port 1 Receive Block Count Usage - This value is the number of blocks allocated to the FIFO logical receive queues.   |

**14.5.6.20 P1\_TX\_IN\_CTL Register (offset = 110h) [reset = 80040C0h]**

P1\_TX\_IN\_CTL is shown in [Figure 14-140](#) and described in [Table 14-155](#).

CPSW PORT 1 TRANSMIT FIFO CONTROL

**Figure 14-140. P1\_TX\_IN\_CTL Register**

|             |    |    |               |          |    |            |    |
|-------------|----|----|---------------|----------|----|------------|----|
| 31          | 30 | 29 | 28            | 27       | 26 | 25         | 24 |
| Reserved    |    |    | HOST_BLKS_Rem |          |    |            |    |
| R/W-8h      |    |    |               |          |    |            |    |
| 23          | 22 | 21 | 20            | 19       | 18 | 17         | 16 |
| TX_RATE_EN  |    |    |               | Reserved |    | TX_IN_SEL  |    |
| R/W-0h      |    |    |               |          |    |            |    |
| 15          | 14 | 13 | 12            | 11       | 10 | 9          | 8  |
| TX_BLKS_Rem |    |    |               | Reserved |    | TX_PRI_WDS |    |
| R/W-4h      |    |    |               |          |    |            |    |
| 7           | 6  | 5  | 4             | 3        | 2  | 1          | 0  |
| TX_PRI_WDS  |    |    |               |          |    |            |    |
| R/W-C0h     |    |    |               |          |    |            |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-155. P1\_TX\_IN\_CTL Register Field Descriptions**

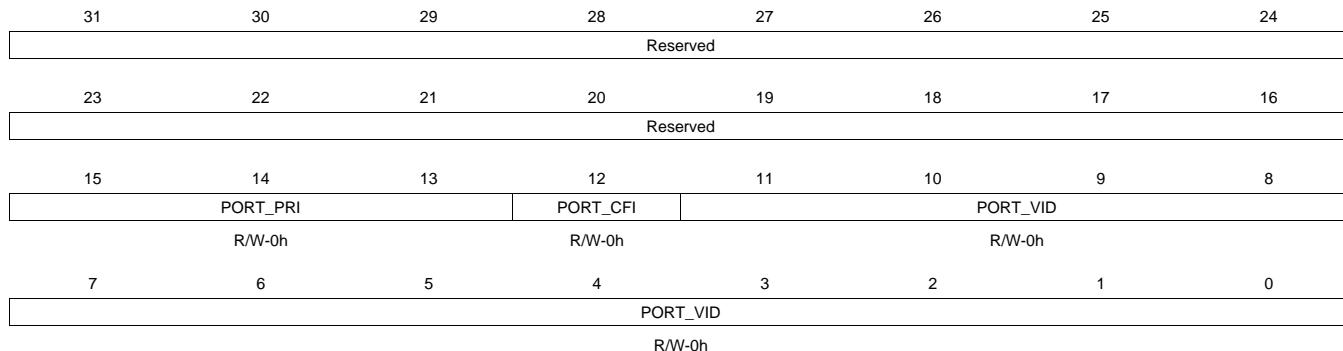
| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 27-24 | HOST_BLKS_Rem | R/W  | 8h    | Transmit FIFO Blocks that must be free before a non rate-limited CPDMA channel can begin sending a packet to the FIFO.         |
| 23-20 | TX_RATE_EN    | R/W  | 0h    | Transmit FIFO Input Rate Enable  |
| 17-16 | TX_IN_SEL     | R/W  | 0h    | Transmit FIFO Input Queue Type Select<br>00 - Normal priority mode<br>01 - reserved<br>10 - Rate Limit mode<br>11 - reserved   |
| 15-12 | TX_BLKS_Rem   | R/W  | 4h    | Transmit FIFO Input Blocks to subtract in dual mac mode and blocks to subtract on non rate-limited traffic in rate-limit mode. |
| 9-0   | TX_PRI_WDS    | R/W  | C0h   | Transmit FIFO Words in queue   |

#### 14.5.6.21 P1\_PORT\_VLAN Register (offset = 114h) [reset = 0h]

P1\_PORT\_VLAN is shown in [Figure 14-141](#) and described in [Table 14-156](#).

##### CPSW PORT 1 VLAN REGISTER

**Figure 14-141. P1\_PORT\_VLAN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-156. P1\_PORT\_VLAN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                                |
|-------|----------|------|-------|--|
| 15-13 | PORT_PRI | R/W  | 0h    | Port VLAN Priority (7 is highest priority) |
| 12    | PORT_CFI | R/W  | 0h    | Port CFI bit                               |
| 11-0  | PORT_VID | R/W  | 0h    | Port VLAN ID                               |

**14.5.6.22 P1\_TX\_PRI\_MAP Register (offset = 118h) [reset = 33221001h]**

P1\_TX\_PRI\_MAP is shown in [Figure 14-142](#) and described in [Table 14-157](#).

CPSW PORT 1 TX HEADER PRIORITY TO SWITCH PRI MAPPING REGISTER

**Figure 14-142. P1\_TX\_PRI\_MAP Register**

| 31       | 30 | 29   | 28     | 27       | 26 | 25   | 24     |
|----------|----|------|--------|----------|----|------|--------|
| Reserved |    | PRI7 |        | Reserved |    | PRI6 |        |
|          |    |      | R/W-3h |          |    |      | R/W-3h |
| 23       | 22 | 21   | 20     | 19       | 18 | 17   | 16     |
| Reserved |    | PRI5 |        | Reserved |    | PRI4 |        |
|          |    |      | R/W-2h |          |    |      | R/W-2h |
| 15       | 14 | 13   | 12     | 11       | 10 | 9    | 8      |
| Reserved |    | PRI3 |        | Reserved |    | PRI2 |        |
|          |    |      | R/W-1h |          |    |      | R/W-0h |
| 7        | 6  | 5    | 4      | 3        | 2  | 1    | 0      |
| Reserved |    | PRI1 |        | Reserved |    | PRI0 |        |
|          |    |      | R/W-0h |          |    |      | R/W-1h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-157. P1\_TX\_PRI\_MAP Register Field Descriptions**

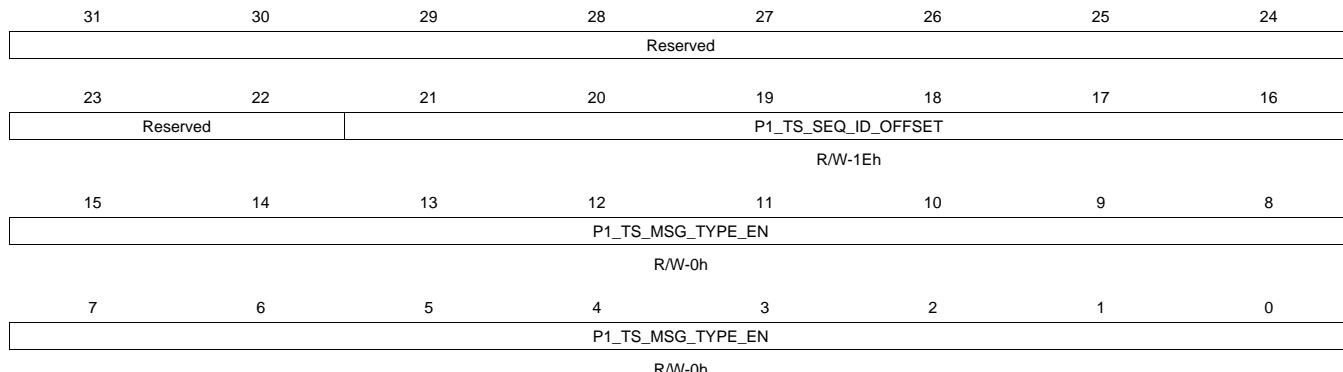
| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 29-28 | PRI7  | R/W  | 3h    | Priority<br>7 - A packet header priority of 0x7 is given this switch queue pri. |
| 25-24 | PRI6  | R/W  | 3h    | Priority<br>6 - A packet header priority of 0x6 is given this switch queue pri. |
| 21-20 | PRI5  | R/W  | 2h    | Priority<br>5 - A packet header priority of 0x5 is given this switch queue pri. |
| 17-16 | PRI4  | R/W  | 2h    | Priority<br>4 - A packet header priority of 0x4 is given this switch queue pri. |
| 13-12 | PRI3  | R/W  | 1h    | Priority<br>3 - A packet header priority of 0x3 is given this switch queue pri. |
| 9-8   | PRI2  | R/W  | 0h    | Priority<br>2 - A packet header priority of 0x2 is given this switch queue pri. |
| 5-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet header priority of 0x1 is given this switch queue pri. |
| 1-0   | PRI0  | R/W  | 1h    | Priority<br>0 - A packet header priority of 0x0 is given this switch queue pri. |

#### 14.5.6.23 P1\_TS\_SEQ\_MTYPE Register (offset = 11Ch) [reset = 1E0000h]

P1\_TS\_SEQ\_MTYPE is shown in [Figure 14-143](#) and described in [Table 14-158](#).

CPSW PORT 1 TIME SYNC SEQUENCE ID OFFSET AND MSG TYPE.

**Figure 14-143. P1\_TS\_SEQ\_MTYPE Register**



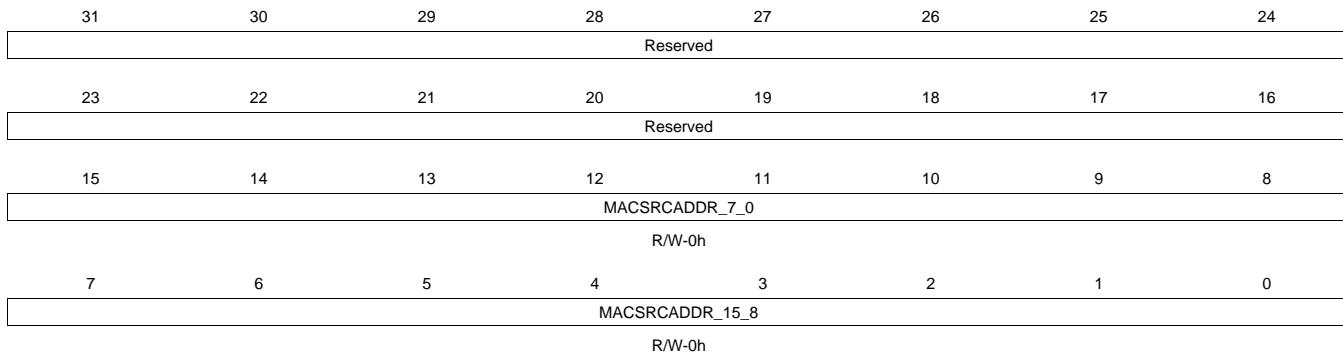
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-158. P1\_TS\_SEQ\_MTYPE Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description  |
|-------|---------------------|------|-------|--|
| 21-16 | P1_TS_SEQ_ID_OFFSET | R/W  | 1Eh   | Port 1 Time Sync Sequence ID Offset This is the number of octets that the sequence ID is offset in the tx and rx time sync message header.<br>The minimum value is 6.                |
| 15-0  | P1_TS_MSG_TYPE_EN   | R/W  | 0h    | Port 1 Time Sync Message Type Enable - Each bit in this field enables the corresponding message type in receive and transmit time sync messages (Bit 0 enables message type 0 etc.). |

**14.5.6.24 P1\_SA\_LO Register (offset = 120h) [reset = 0h]**

P1\_SA\_LO is shown in [Figure 14-144](#) and described in [Table 14-159](#).

**CPSW CPGMAC\_SL1 SOURCE ADDRESS LOW REGISTER**
**Figure 14-144. P1\_SA\_LO Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-159. P1\_SA\_LO Register Field Descriptions**

| Bit  | Field           | Type | Reset | Description                          |
|------|-----------------|------|-------|--------------------------------------|
| 15-8 | MACSRCADDR_7_0  | R/W  | 0h    | Source Address Lower 8 bits (byte 0) |
| 7-0  | MACSRCADDR_15_8 | R/W  | 0h    | Source Address bits 15:8 (byte 1)    |

#### 14.5.6.25 P1\_SA\_HI Register (offset = 124h) [reset = 0h]

P1\_SA\_HI is shown in [Figure 14-145](#) and described in [Table 14-160](#).

##### CPSW CPGMAC\_SL1 SOURCE ADDRESS HIGH REGISTER

**Figure 14-145. P1\_SA\_HI Register**

| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|------------------|----|----|----|----|----|----|----|
| MACSRCADDR_23_16 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MACSRCADDR_31_24 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| MACSRCADDR_39_32 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| MACSRCADDR_47_40 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-160. P1\_SA\_HI Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description                        |
|-------|------------------|------|-------|------------------------------------|
| 31-24 | MACSRCADDR_23_16 | R/W  | 0h    | Source Address bits 23:16 (byte 2) |
| 23-16 | MACSRCADDR_31_24 | R/W  | 0h    | Source Address bits 31:24 (byte 3) |
| 15-8  | MACSRCADDR_39_32 | R/W  | 0h    | Source Address bits 39:32 (byte 4) |
| 7-0   | MACSRCADDR_47_40 | R/W  | 0h    | Source Address bits 47:40 (byte 5) |

#### 14.5.6.26 P1\_SEND\_PERCENT Register (offset = 128h) [reset = 0h]

P1\_SEND\_PERCENT is shown in [Figure 14-146](#) and described in [Table 14-161](#).

CPSW PORT 1 TRANSMIT QUEUE SEND PERCENTAGES

**Figure 14-146. P1\_SEND\_PERCENT Register**

|          |    |    |    |                   |    |    |    |
|----------|----|----|----|-------------------|----|----|----|
| 31       | 30 | 29 | 28 | 27                | 26 | 25 | 24 |
| Reserved |    |    |    |                   |    |    |    |
| 23       | 22 | 21 | 20 | 19                | 18 | 17 | 16 |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI3_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |
| 15       | 14 | 13 | 12 | 11                | 10 | 9  | 8  |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI2_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |
| 7        | 6  | 5  | 4  | 3                 | 2  | 1  | 0  |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI1_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-161. P1\_SEND\_PERCENT Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 22-16 | PRI3_SEND_PERCENT | R/W  | 0h    | Priority 3 Transmit Percentage - This percentage value is sent from FIFO priority 3 (maximum) when the p1_pri3_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 3 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |
| 14-8  | PRI2_SEND_PERCENT | R/W  | 0h    | Priority 2 Transmit Percentage - This percentage value is sent from FIFO priority 2 (maximum) when the p1_pri2_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 2 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |
| 6-0   | PRI1_SEND_PERCENT | R/W  | 0h    | Priority 1 Transmit Percentage - This percentage value is sent from FIFO priority 1 (maximum) when the p1_pri1_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 1 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |

**14.5.6.27 P1\_RX\_DSCP\_PRI\_MAP0 Register (offset = 130h) [reset = 0h]**

 P1\_RX\_DSCP\_PRI\_MAP0 is shown in [Figure 14-147](#) and described in [Table 14-162](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0

**Figure 14-147. P1\_RX\_DSCP\_PRI\_MAP0 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI7   |    | Reserved |    | PRI6   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI5   |    | Reserved |    | PRI4   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI3   |    | Reserved |    | PRI2   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI1   |    | Reserved |    | PRI0   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-162. P1\_RX\_DSCP\_PRI\_MAP0 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI7  | R/W  | 0h    | Priority<br>7 - A packet TOS of 0d7 is mapped to this received packet priority. |
| 26-24 | PRI6  | R/W  | 0h    | Priority<br>6 - A packet TOS of 0d6 is mapped to this received packet priority. |
| 22-20 | PRI5  | R/W  | 0h    | Priority<br>5 - A packet TOS of 0d5 is mapped to this received packet priority. |
| 18-16 | PRI4  | R/W  | 0h    | Priority<br>4 - A packet TOS of 0d4 is mapped to this received packet priority. |
| 14-12 | PRI3  | R/W  | 0h    | Priority<br>3 - A packet TOS of 0d3 is mapped to this received packet priority. |
| 10-8  | PRI2  | R/W  | 0h    | Priority<br>2 - A packet TOS of 0d2 is mapped to this received packet priority. |
| 6-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet TOS of 0d1 is mapped to this received packet priority. |
| 2-0   | PRI0  | R/W  | 0h    | Priority<br>0 - A packet TOS of 0d0 is mapped to this received packet priority. |

**14.5.6.28 P1\_RX\_DSCP\_PRI\_MAP1 Register (offset = 134h) [reset = 0h]**

P1\_RX\_DSCP\_PRI\_MAP1 is shown in [Figure 14-148](#) and described in [Table 14-163](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1

**Figure 14-148. P1\_RX\_DSCP\_PRI\_MAP1 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI15  |    | Reserved |    | PRI14  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI13  |    | Reserved |    | PRI12  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI11  |    | Reserved |    | PRI10  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI9   |    | Reserved |    | PRI8   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-163. P1\_RX\_DSCP\_PRI\_MAP1 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI15 | R/W  | 0h    | Priority<br>15 - A packet TOS of 0d15 is mapped to this received packet priority. |
| 26-24 | PRI14 | R/W  | 0h    | Priority<br>14 - A packet TOS of 0d14 is mapped to this received packet priority. |
| 22-20 | PRI13 | R/W  | 0h    | Priority<br>13 - A packet TOS of 0d13 is mapped to this received packet priority. |
| 18-16 | PRI12 | R/W  | 0h    | Priority<br>12 - A packet TOS of 0d12 is mapped to this received packet priority. |
| 14-12 | PRI11 | R/W  | 0h    | Priority<br>11 - A packet TOS of 0d11 is mapped to this received packet priority. |
| 10-8  | PRI10 | R/W  | 0h    | Priority<br>10 - A packet TOS of 0d10 is mapped to this received packet priority. |
| 6-4   | PRI9  | R/W  | 0h    | Priority<br>9 - A packet TOS of 0d9 is mapped to this received packet priority.   |
| 2-0   | PRI8  | R/W  | 0h    | Priority<br>8 - A packet TOS of 0d8 is mapped to this received packet priority.   |

**14.5.6.29 P1\_RX\_DSCP\_PRI\_MAP2 Register (offset = 138h) [reset = 0h]**

 P1\_RX\_DSCP\_PRI\_MAP2 is shown in [Figure 14-149](#) and described in [Table 14-164](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2

**Figure 14-149. P1\_RX\_DSCP\_PRI\_MAP2 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI23  |    | Reserved |    | PRI22  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI21  |    | Reserved |    | PRI20  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI19  |    | Reserved |    | PRI18  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI17  |    | Reserved |    | PRI16  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-164. P1\_RX\_DSCP\_PRI\_MAP2 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI23 | R/W  | 0h    | Priority<br>23 - A packet TOS of 0d23 is mapped to this received packet priority. |
| 26-24 | PRI22 | R/W  | 0h    | Priority<br>22 - A packet TOS of 0d22 is mapped to this received packet priority. |
| 22-20 | PRI21 | R/W  | 0h    | Priority<br>21 - A packet TOS of 0d21 is mapped to this received packet priority. |
| 18-16 | PRI20 | R/W  | 0h    | Priority<br>20 - A packet TOS of 0d20 is mapped to this received packet priority. |
| 14-12 | PRI19 | R/W  | 0h    | Priority<br>19 - A packet TOS of 0d19 is mapped to this received packet priority. |
| 10-8  | PRI18 | R/W  | 0h    | Priority<br>18 - A packet TOS of 0d18 is mapped to this received packet priority. |
| 6-4   | PRI17 | R/W  | 0h    | Priority<br>17 - A packet TOS of 0d17 is mapped to this received packet priority. |
| 2-0   | PRI16 | R/W  | 0h    | Priority<br>16 - A packet TOS of 0d16 is mapped to this received packet priority. |

**14.5.6.30 P1\_RX\_DSCP\_PRI\_MAP3 Register (offset = 13Ch) [reset = 0h]**

P1\_RX\_DSCP\_PRI\_MAP3 is shown in [Figure 14-150](#) and described in [Table 14-165](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3

**Figure 14-150. P1\_RX\_DSCP\_PRI\_MAP3 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI31  |    | Reserved |    | PRI30  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI29  |    | Reserved |    | PRI28  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI27  |    | Reserved |    | PRI26  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI25  |    | Reserved |    | PRI24  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-165. P1\_RX\_DSCP\_PRI\_MAP3 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI31 | R/W  | 0h    | Priority<br>31 - A packet TOS of 0d31 is mapped to this received packet priority. |
| 26-24 | PRI30 | R/W  | 0h    | Priority<br>30 - A packet TOS of 0d30 is mapped to this received packet priority. |
| 22-20 | PRI29 | R/W  | 0h    | Priority<br>29 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 18-16 | PRI28 | R/W  | 0h    | Priority<br>28 - A packet TOS of 0d28 is mapped to this received packet priority. |
| 14-12 | PRI27 | R/W  | 0h    | Priority<br>27 - A packet TOS of 0d27 is mapped to this received packet priority. |
| 10-8  | PRI26 | R/W  | 0h    | Priority<br>26 - A packet TOS of 0d26 is mapped to this received packet priority. |
| 6-4   | PRI25 | R/W  | 0h    | Priority<br>25 - A packet TOS of 0d25 is mapped to this received packet priority. |
| 2-0   | PRI24 | R/W  | 0h    | Priority<br>24 - A packet TOS of 0d24 is mapped to this received packet priority. |

**14.5.6.31 P1\_RX\_DSCP\_PRI\_MAP4 Register (offset = 140h) [reset = 0h]**

 P1\_RX\_DSCP\_PRI\_MAP4 is shown in [Figure 14-151](#) and described in [Table 14-166](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4

**Figure 14-151. P1\_RX\_DSCP\_PRI\_MAP4 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI39  |    | Reserved |    | PRI38  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI37  |    | Reserved |    | PRI36  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI35  |    | Reserved |    | PRI34  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI33  |    | Reserved |    | PRI32  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-166. P1\_RX\_DSCP\_PRI\_MAP4 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI39 | R/W  | 0h    | Priority<br>39 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 26-24 | PRI38 | R/W  | 0h    | Priority<br>38 - A packet TOS of 0d38 is mapped to this received packet priority. |
| 22-20 | PRI37 | R/W  | 0h    | Priority<br>37 - A packet TOS of 0d37 is mapped to this received packet priority. |
| 18-16 | PRI36 | R/W  | 0h    | Priority<br>36 - A packet TOS of 0d36 is mapped to this received packet priority. |
| 14-12 | PRI35 | R/W  | 0h    | Priority<br>35 - A packet TOS of 0d35 is mapped to this received packet priority. |
| 10-8  | PRI34 | R/W  | 0h    | Priority<br>34 - A packet TOS of 0d34 is mapped to this received packet priority. |
| 6-4   | PRI33 | R/W  | 0h    | Priority<br>33 - A packet TOS of 0d33 is mapped to this received packet priority. |
| 2-0   | PRI32 | R/W  | 0h    | Priority<br>32 - A packet TOS of 0d32 is mapped to this received packet priority. |

**14.5.6.32 P1\_RX\_DSCP\_PRI\_MAP5 Register (offset = 144h) [reset = 0h]**

P1\_RX\_DSCP\_PRI\_MAP5 is shown in [Figure 14-152](#) and described in [Table 14-167](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5

**Figure 14-152. P1\_RX\_DSCP\_PRI\_MAP5 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI47  |    | Reserved |    | PRI46  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI45  |    | Reserved |    | PRI44  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI43  |    | Reserved |    | PRI42  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI41  |    | Reserved |    | PRI40  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-167. P1\_RX\_DSCP\_PRI\_MAP5 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI47 | R/W  | 0h    | Priority<br>47 - A packet TOS of 0d47 is mapped to this received packet priority. |
| 26-24 | PRI46 | R/W  | 0h    | Priority<br>46 - A packet TOS of 0d46 is mapped to this received packet priority. |
| 22-20 | PRI45 | R/W  | 0h    | Priority<br>45 - A packet TOS of 0d45 is mapped to this received packet priority. |
| 18-16 | PRI44 | R/W  | 0h    | Priority<br>44 - A packet TOS of 0d44 is mapped to this received packet priority. |
| 14-12 | PRI43 | R/W  | 0h    | Priority<br>43 - A packet TOS of 0d43 is mapped to this received packet priority. |
| 10-8  | PRI42 | R/W  | 0h    | Priority<br>42 - A packet TOS of 0d42 is mapped to this received packet priority. |
| 6-4   | PRI41 | R/W  | 0h    | Priority<br>41 - A packet TOS of 0d41 is mapped to this received packet priority. |
| 2-0   | PRI40 | R/W  | 0h    | Priority<br>40 - A packet TOS of 0d40 is mapped to this received packet priority. |

**14.5.6.33 P1\_RX\_DSCP\_PRI\_MAP6 Register (offset = 148h) [reset = 0h]**

 P1\_RX\_DSCP\_PRI\_MAP6 is shown in [Figure 14-153](#) and described in [Table 14-168](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6

**Figure 14-153. P1\_RX\_DSCP\_PRI\_MAP6 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI55  |    | Reserved |    | PRI54  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI53  |    | Reserved |    | PRI52  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI51  |    | Reserved |    | PRI50  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI49  |    | Reserved |    | PRI48  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-168. P1\_RX\_DSCP\_PRI\_MAP6 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI55 | R/W  | 0h    | Priority<br>55 - A packet TOS of 0d55 is mapped to this received packet priority. |
| 26-24 | PRI54 | R/W  | 0h    | Priority<br>54 - A packet TOS of 0d54 is mapped to this received packet priority. |
| 22-20 | PRI53 | R/W  | 0h    | Priority<br>53 - A packet TOS of 0d53 is mapped to this received packet priority. |
| 18-16 | PRI52 | R/W  | 0h    | Priority<br>52 - A packet TOS of 0d52 is mapped to this received packet priority. |
| 14-12 | PRI51 | R/W  | 0h    | Priority<br>51 - A packet TOS of 0d51 is mapped to this received packet priority. |
| 10-8  | PRI50 | R/W  | 0h    | Priority<br>50 - A packet TOS of 0d50 is mapped to this received packet priority. |
| 6-4   | PRI49 | R/W  | 0h    | Priority<br>49 - A packet TOS of 0d49 is mapped to this received packet priority. |
| 2-0   | PRI48 | R/W  | 0h    | Priority<br>48 - A packet TOS of 0d48 is mapped to this received packet priority. |

**14.5.6.34 P1\_RX\_DSCP\_PRI\_MAP7 Register (offset = 14Ch) [reset = 0h]**

P1\_RX\_DSCP\_PRI\_MAP7 is shown in [Figure 14-154](#) and described in [Table 14-169](#).

CPSW PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7

**Figure 14-154. P1\_RX\_DSCP\_PRI\_MAP7 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI63  |    | Reserved |    | PRI62  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI61  |    | Reserved |    | PRI60  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI59  |    | Reserved |    | PRI58  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI57  |    | Reserved |    | PRI56  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-169. P1\_RX\_DSCP\_PRI\_MAP7 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI63 | R/W  | 0h    | Priority<br>63 - A packet TOS of 0d63 is mapped to this received packet priority. |
| 26-24 | PRI62 | R/W  | 0h    | Priority<br>62 - A packet TOS of 0d62 is mapped to this received packet priority. |
| 22-20 | PRI61 | R/W  | 0h    | Priority<br>61 - A packet TOS of 0d61 is mapped to this received packet priority. |
| 18-16 | PRI60 | R/W  | 0h    | Priority<br>60 - A packet TOS of 0d60 is mapped to this received packet priority. |
| 14-12 | PRI59 | R/W  | 0h    | Priority<br>59 - A packet TOS of 0d59 is mapped to this received packet priority. |
| 10-8  | PRI58 | R/W  | 0h    | Priority<br>58 - A packet TOS of 0d58 is mapped to this received packet priority. |
| 6-4   | PRI57 | R/W  | 0h    | Priority<br>57 - A packet TOS of 0d57 is mapped to this received packet priority. |
| 2-0   | PRI56 | R/W  | 0h    | Priority<br>56 - A packet TOS of 0d56 is mapped to this received packet priority. |

#### 14.5.6.35 P2\_CONTROL Register (offset = 200h) [reset = 0h]

P2\_CONTROL is shown in [Figure 14-155](#) and described in [Table 14-170](#).

##### CPSW\_3GF PORT 2 CONTROL REGISTER

**Figure 14-155. P2\_CONTROL Register**

|                                |           |                    |                   |                    |                |                    |                    |  |
|--------------------------------|-----------|--------------------|-------------------|--------------------|----------------|--------------------|--------------------|--|
| 31                             | 30        | 29                 | 28                | 27                 | 26             | 25                 | 24                 |  |
| Reserved                       |           |                    |                   |                    |                |                    | P2_PASS_PRI_TAGGED |  |
| R/W-0h                         |           |                    |                   |                    |                |                    |                    |  |
| 23                             | 22        | 21                 | 20                | 19                 | 18             | 17                 | 16                 |  |
| Reserved                       |           | P2_VLAN_LTYPE2_EN  | P2_VLAN_LTYPE1_EN | Reserved           |                |                    | P2_DSCP_PRI_EN     |  |
| R/W-0h      R/W-0h      R/W-0h |           |                    |                   |                    |                |                    |                    |  |
| 15                             | 14        | 13                 | 12                | 11                 | 10             | 9                  | 8                  |  |
| Reserved                       | P2_TS_320 | P2_TS_319          | P2_TS_132         | P2_TS_131          | P2_TS_130      | P2_TS_129          | P2_TS_TTL_NONZERO  |  |
| R/W-0h      R/W-0h             |           | R/W-0h      R/W-0h |                   | R/W-0h      R/W-0h |                | R/W-0h      R/W-0h |                    |  |
| 7                              | 6         | 5                  | 4                 | 3                  | 2              | 1                  | 0                  |  |
| Reserved                       |           |                    | P2_TS_ANX_D_EN    | P2_TS_LTYP2_EN     | P2_TS_LTYP1_EN | P2_TS_TX_EN        | P2_TS_RX_EN        |  |
| R/W-0h                         |           |                    | R-0h              |                    | R/W-0h         |                    | R/W-0h             |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-170. P2\_CONTROL Register Field Descriptions**

| Bit | Field              | Type | Reset | Description   |
|-----|--------------------|------|-------|---|
| 24  | P2_PASS_PRI_TAGGED | R/W  | 0h    | Port 2 Pass Priority Tagged<br>0 - Priority tagged packets have the zero VID replaced with the input port P2_PORT_VLAN [11:0]<br>1 - Priority tagged packets are processed unchanged.   |
| 21  | P2_VLAN_LTYPE2_EN  | R/W  | 0h    | Port 2 VLAN LTYPE 2 enable<br>0 - disabled<br>1 - VLAN LTYPE2 enabled on transmit and receive   |
| 20  | P2_VLAN_LTYPE1_EN  | R/W  | 0h    | Port 2 VLAN LTYPE 1 enable<br>0 - disabled<br>1 - VLAN LTYPE1 enabled on transmit and receive   |
| 16  | P2_DSCP_PRI_EN     | R/W  | 0h    | Port 0 DSCP Priority Enable<br>0 - DSCP priority disabled<br>1 - DSCP priority enabled.<br>All non-tagged IPv4 packets have their received packet priority determined by mapping the 6 TOS bits through the port DSCP priority mapping registers. |
| 14  | P2_TS_320          | R/W  | 0h    | Port 2 Time Sync Destination Port Number 320 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination port number 320 (decimal) is enabled.  |
| 13  | P2_TS_319          | R/W  | 0h    | Port 2 Time Sync Destination Port Number 319 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination port number 319 (decimal) is enabled.  |
| 12  | P2_TS_132          | R/W  | 0h    | Port 2 Time Sync Destination IP Address 132 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 132 (decimal) is enabled.   |
| 11  | P2_TS_131          | R/W  | 0h    | Port 2 Time Sync Destination IP Address 131 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 131 (decimal) is enabled.   |

**Table 14-170. P2\_CONTROL Register Field Descriptions (continued)**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 10  | P2_TS_130         | R/W  | 0h    | Port 2 Time Sync Destination IP Address 130 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 130 (decimal) is enabled. |
| 9   | P2_TS_129         | R/W  | 0h    | Port 2 Time Sync Destination IP Address 129 enable<br>0 - disabled<br>1 - Annex D (UDP/IPv4) time sync packet destination IP address number 129 (decimal) is enabled. |
| 8   | P2_TS_TTL_NONZERO | R/W  | 0h    | Port 2 Time Sync Time To Live Non-zero enable.<br>0 = TTL must be zero.<br>1 = TTL may be any value.  |
| 4   | P2_TS_ANNEX_D_EN  | R/W  | 0h    | Port 2 Time Sync Annex D enable<br>0 - Annex D disabled<br>1 - Annex D enabled  |
| 3   | P2_TS_LTYPE2_EN   | R    | 0h    | Port 2 Time Sync LTYPE 2 enable<br>0 - disabled<br>1 - enabled  |
| 2   | P2_TS_LTYPE1_EN   | R/W  | 0h    | Port 2 Time Sync LTYPE 1 enable<br>0 - disabled<br>1 - enabled  |
| 1   | P2_TS_TX_EN       | R/W  | 0h    | Port 2 Time Sync Transmit Enable<br>0 - disabled<br>1 - enabled   |
| 0   | P2_TS_RX_EN       | R/W  | 0h    | Port 2 Time Sync Receive Enable<br>0 - Port 1 Receive Time Sync disabled<br>1 - Port 1 Receive Time Sync enabled  |

#### 14.5.6.36 P2\_MAX\_BLKS Register (offset = 208h) [reset = 113h]

P2\_MAX\_BLKS is shown in [Figure 14-156](#) and described in [Table 14-171](#).

##### CPSW PORT 2 MAXIMUM FIFO BLOCKS REGISTER

**Figure 14-156. P2\_MAX\_BLKS Register**

| 31             | 30 | 29 | 28 | 27             | 26 | 25             | 24 |
|----------------|----|----|----|----------------|----|----------------|----|
| Reserved       |    |    |    |                |    |                |    |
| 23             | 22 | 21 | 20 | 19             | 18 | 17             | 16 |
| Reserved       |    |    |    |                |    |                |    |
| 15             | 14 | 13 | 12 | 11             | 10 | 9              | 8  |
| Reserved       |    |    |    |                |    | P2_TX_MAX_BLKS |    |
| R/W-11h        |    |    |    |                |    |                |    |
| 7              | 6  | 5  | 4  | 3              | 2  | 1              | 0  |
| P2_TX_MAX_BLKS |    |    |    | P2_RX_MAX_BLKS |    |                |    |
| R/W-11h        |    |    |    | R/W-3h         |    |                |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-171. P2\_MAX\_BLKS Register Field Descriptions**

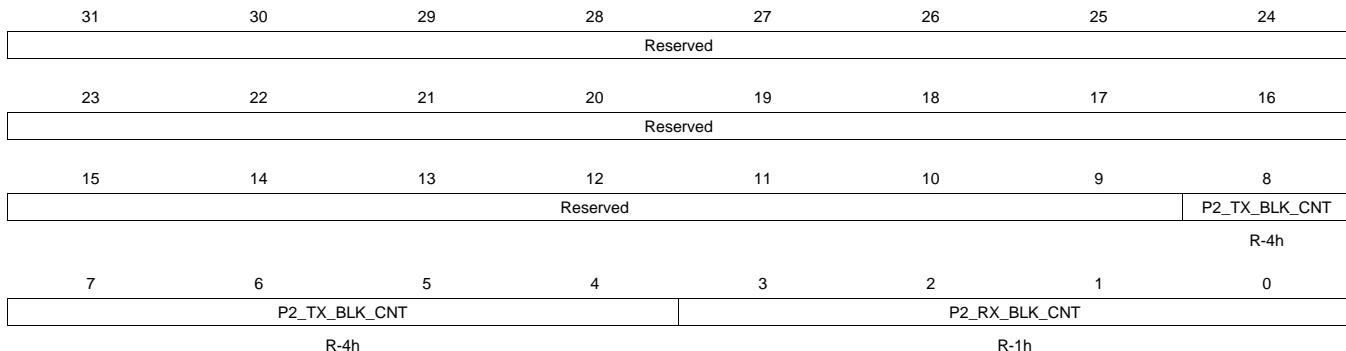
| Bit | Field          | Type | Reset | Description  |
|-----|----------------|------|-------|--|
| 8-4 | P2_TX_MAX_BLKS | R/W  | 11h   | Transmit FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical transmit priority queues.<br>0x11 is the recommended value of p2_tx_max_blks unless the port is in fullduplex flow control mode.<br>In flow control mode, the p2_rx_max_blks will need to increase in order to accept the required run out in fullduplex mode.<br>This value will need to decrease by the amount of increase in p2_rx_max_blks.<br>0xe is the minimum value tx max blks. |
| 3-0 | P2_RX_MAX_BLKS | R/W  | 3h    | Receive FIFO Maximum Blocks - This value is the maximum number of 1k memory blocks that may be allocated to the FIFO's logical receive queue.<br>This value must be greater than or equal to 0x3.<br>It should be increased in fullduplex flow control mode to 0x5 or 0x6 depending on the required runout space.<br>The p2_tx_max_blks value must be decreased by the amount of increase in p2_rx_max_blks.<br>0x3 is the minimum value rx max blks and 0x6 is the maximum value.                                       |

**14.5.6.37 P2\_BLK\_CNT Register (offset = 20Ch) [reset = 41h]**

P2\_BLK\_CNT is shown in [Figure 14-157](#) and described in [Table 14-172](#).

CPSW PORT 2 FIFO BLOCK USAGE COUNT (READ ONLY)

**Figure 14-157. P2\_BLK\_CNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-172. P2\_BLK\_CNT Register Field Descriptions**

| Bit | Field         | Type | Reset | Description   |
|-----|---------------|------|-------|---|
| 8-4 | P2_TX_BLK_CNT | R    | 4h    | Port 2 Transmit Block Count Usage - This value is the number of blocks allocated to the FIFO logical transmit queues. |
| 3-0 | P2_RX_BLK_CNT | R    | 1h    | Port 2 Receive Block Count Usage - This value is the number of blocks allocated to the FIFO logical receive queues.   |

#### 14.5.6.38 P2\_TX\_IN\_CTL Register (offset = 210h) [reset = 80040C0h]

P2\_TX\_IN\_CTL is shown in [Figure 14-158](#) and described in [Table 14-173](#).

CPSW PORT 2 TRANSMIT FIFO CONTROL

**Figure 14-158. P2\_TX\_IN\_CTL Register**

|             |    |    |    |               |    |            |    |
|-------------|----|----|----|---------------|----|------------|----|
| 31          | 30 | 29 | 28 | 27            | 26 | 25         | 24 |
| Reserved    |    |    |    | HOST_BLKS_Rem |    |            |    |
| R/W-8h      |    |    |    |               |    |            |    |
| 23          | 22 | 21 | 20 | 19            | 18 | 17         | 16 |
| TX_RATE_EN  |    |    |    | Reserved      |    | TX_IN_SEL  |    |
| R/W-0h      |    |    |    |               |    |            |    |
| 15          | 14 | 13 | 12 | 11            | 10 | 9          | 8  |
| TX_BLKS_Rem |    |    |    | Reserved      |    | TX_PRI_WDS |    |
| R/W-4h      |    |    |    |               |    |            |    |
| 7           | 6  | 5  | 4  | 3             | 2  | 1          | 0  |
| TX_PRI_WDS  |    |    |    |               |    |            |    |
| R/W-C0h     |    |    |    |               |    |            |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-173. P2\_TX\_IN\_CTL Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 27-24 | HOST_BLKS_Rem | R/W  | 8h    | Transmit FIFO Blocks that must be free before a non rate-limited CPDMA channel can begin sending a packet to the FIFO.         |
| 23-20 | TX_RATE_EN    | R/W  | 0h    | Transmit FIFO Input Rate Enable  |
| 17-16 | TX_IN_SEL     | R/W  | 0h    | Transmit FIFO Input Queue Type Select<br>00 - Normal priority mode<br>01 - reserved<br>10 - Rate Limit mode<br>11 - reserved   |
| 15-12 | TX_BLKS_Rem   | R/W  | 4h    | Transmit FIFO Input Blocks to subtract in dual mac mode and blocks to subtract on non rate-limited traffic in rate-limit mode. |
| 9-0   | TX_PRI_WDS    | R/W  | C0h   | Transmit FIFO Words in queue   |

#### 14.5.6.39 P2\_PORT\_VLAN Register (offset = 214h) [reset = 0h]

P2\_PORT\_VLAN is shown in [Figure 14-159](#) and described in [Table 14-174](#).

##### CPSW PORT 2 VLAN REGISTER

**Figure 14-159. P2\_PORT\_VLAN Register**

|          |    |          |    |          |    |    |    |
|----------|----|----------|----|----------|----|----|----|
| 31       | 30 | 29       | 28 | 27       | 26 | 25 | 24 |
| Reserved |    |          |    |          |    |    |    |
| 23       | 22 | 21       | 20 | 19       | 18 | 17 | 16 |
| Reserved |    |          |    |          |    |    |    |
| 15       | 14 | 13       | 12 | 11       | 10 | 9  | 8  |
| PORT_PRI |    | PORT_CFI |    | PORT_VID |    |    |    |
| R/W-0h   |    | R/W-0h   |    | R/W-0h   |    |    |    |
| 7        | 6  | 5        | 4  | 3        | 2  | 1  | 0  |
| PORT_VID |    |          |    |          |    |    |    |
| R/W-0h   |    |          |    |          |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-174. P2\_PORT\_VLAN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description                                |
|-------|----------|------|-------|--|
| 15-13 | PORT_PRI | R/W  | 0h    | Port VLAN Priority (7 is highest priority) |
| 12    | PORT_CFI | R/W  | 0h    | Port CFI bit                               |
| 11-0  | PORT_VID | R/W  | 0h    | Port VLAN ID                               |

#### 14.5.6.40 P2\_TX\_PRI\_MAP Register (offset = 218h) [reset = 33221001h]

P2\_TX\_PRI\_MAP is shown in [Figure 14-160](#) and described in [Table 14-175](#).

CPSW PORT 2 TX HEADER PRIORITY TO SWITCH PRI MAPPING REGISTER

**Figure 14-160. P2\_TX\_PRI\_MAP Register**

| 31       | 30 | 29   | 28     | 27       | 26 | 25   | 24     |
|----------|----|------|--------|----------|----|------|--------|
| Reserved |    | PRI7 |        | Reserved |    | PRI6 |        |
|          |    |      | R/W-3h |          |    |      | R/W-3h |
| 23       | 22 | 21   | 20     | 19       | 18 | 17   | 16     |
| Reserved |    | PRI5 |        | Reserved |    | PRI4 |        |
|          |    |      | R/W-2h |          |    |      | R/W-2h |
| 15       | 14 | 13   | 12     | 11       | 10 | 9    | 8      |
| Reserved |    | PRI3 |        | Reserved |    | PRI2 |        |
|          |    |      | R/W-1h |          |    |      | R/W-0h |
| 7        | 6  | 5    | 4      | 3        | 2  | 1    | 0      |
| Reserved |    | PRI1 |        | Reserved |    | PRI0 |        |
|          |    |      | R/W-0h |          |    |      | R/W-1h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-175. P2\_TX\_PRI\_MAP Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 29-28 | PRI7  | R/W  | 3h    | Priority<br>7 - A packet header priority of 0x7 is given this switch queue pri. |
| 25-24 | PRI6  | R/W  | 3h    | Priority<br>6 - A packet header priority of 0x6 is given this switch queue pri. |
| 21-20 | PRI5  | R/W  | 2h    | Priority<br>5 - A packet header priority of 0x5 is given this switch queue pri. |
| 17-16 | PRI4  | R/W  | 2h    | Priority<br>4 - A packet header priority of 0x4 is given this switch queue pri. |
| 13-12 | PRI3  | R/W  | 1h    | Priority<br>3 - A packet header priority of 0x3 is given this switch queue pri. |
| 9-8   | PRI2  | R/W  | 0h    | Priority<br>2 - A packet header priority of 0x2 is given this switch queue pri. |
| 5-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet header priority of 0x1 is given this switch queue pri. |
| 1-0   | PRI0  | R/W  | 1h    | Priority<br>0 - A packet header priority of 0x0 is given this switch queue pri. |

**14.5.6.41 P2\_TS\_SEQ\_MTYPE Register (offset = 21Ch) [reset = 1E0000h]**

P2\_TS\_SEQ\_MTYPE is shown in [Figure 14-161](#) and described in [Table 14-176](#).

CPSW\_3GF PORT 2 TIME SYNC SEQUENCE ID OFFSET AND MSG TYPE.

**Figure 14-161. P2\_TS\_SEQ\_MTYPE Register**

|                   |    |                     |    |    |    |    |         |
|-------------------|----|---------------------|----|----|----|----|---------|
| 31                | 30 | 29                  | 28 | 27 | 26 | 25 | 24      |
| Reserved          |    |                     |    |    |    |    |         |
| 23                | 22 | 21                  | 20 | 19 | 18 | 17 | 16      |
| Reserved          |    | P2_TS_SEQ_ID_OFFSET |    |    |    |    |         |
|                   |    |                     |    |    |    |    | R/W-1Eh |
| 15                | 14 | 13                  | 12 | 11 | 10 | 9  | 8       |
| P2_TS_MSG_TYPE_EN |    |                     |    |    |    |    |         |
|                   |    |                     |    |    |    |    | R/W-0h  |
| 7                 | 6  | 5                   | 4  | 3  | 2  | 1  | 0       |
| P2_TS_MSG_TYPE_EN |    |                     |    |    |    |    |         |
|                   |    |                     |    |    |    |    | R/W-0h  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-176. P2\_TS\_SEQ\_MTYPE Register Field Descriptions**

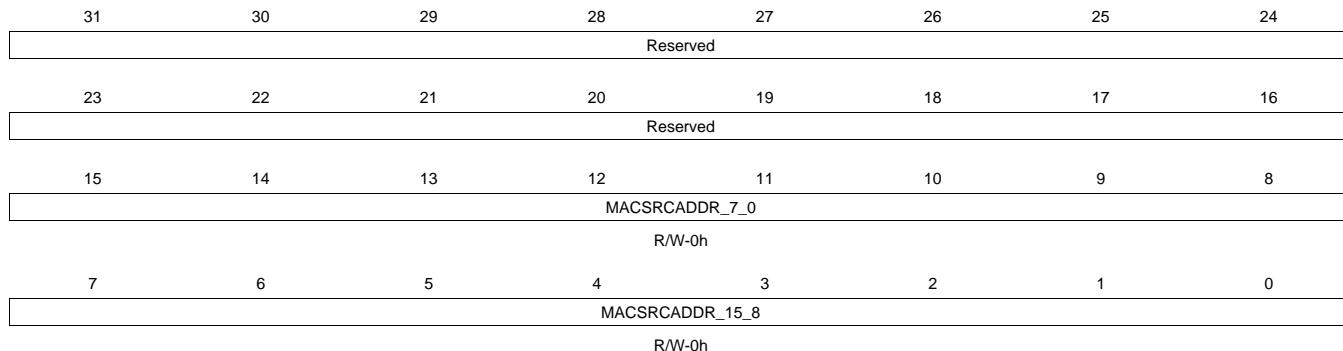
| Bit   | Field               | Type | Reset | Description  |
|-------|---------------------|------|-------|--|
| 21-16 | P2_TS_SEQ_ID_OFFSET | R/W  | 1Eh   | Port 2 Time Sync Sequence ID Offset This is the number of octets that the sequence ID is offset in the tx and rx time sync message header.<br>The minimum value is 6.                |
| 15-0  | P2_TS_MSG_TYPE_EN   | R/W  | 0h    | Port 2 Time Sync Message Type Enable - Each bit in this field enables the corresponding message type in receive and transmit time sync messages (Bit 0 enables message type 0 etc.). |

#### 14.5.6.42 P2\_SA\_LO Register (offset = 220h) [reset = 0h]

P2\_SA\_LO is shown in [Figure 14-162](#) and described in [Table 14-177](#).

##### CPSW\_CPGMAC\_SL2 SOURCE ADDRESS LOW REGISTER

**Figure 14-162. P2\_SA\_LO Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-177. P2\_SA\_LO Register Field Descriptions**

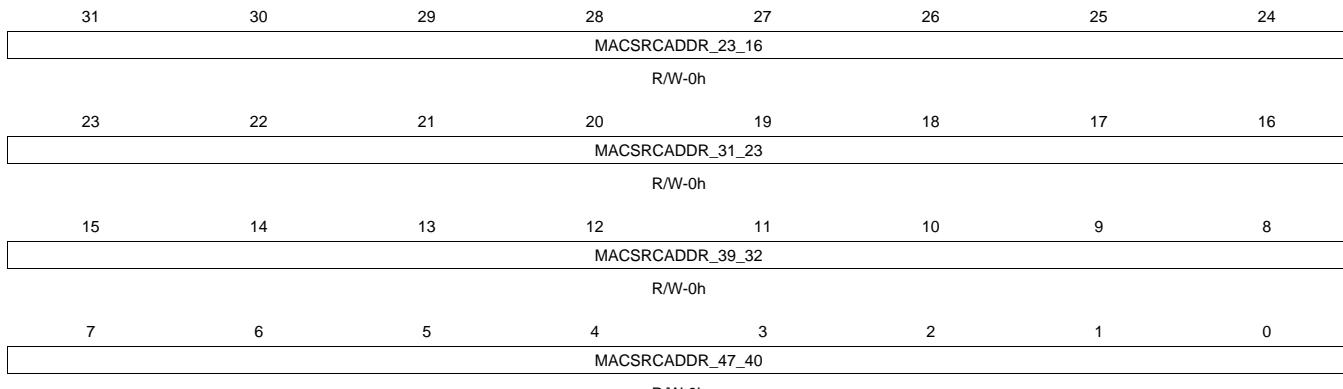
| Bit  | Field           | Type | Reset | Description                          |
|------|-----------------|------|-------|--------------------------------------|
| 15-8 | MACSRCADDR_7_0  | R/W  | 0h    | Source Address Lower 8 bits (byte 0) |
| 7-0  | MACSRCADDR_15_8 | R/W  | 0h    | Source Address bits 15:8 (byte 1)    |

#### 14.5.6.43 P2\_SA\_HI Register (offset = 224h) [reset = 0h]

P2\_SA\_HI is shown in [Figure 14-163](#) and described in [Table 14-178](#).

##### CPSW CPGMAC\_SL2 SOURCE ADDRESS HIGH REGISTER

**Figure 14-163. P2\_SA\_HI Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-178. P2\_SA\_HI Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description                        |
|-------|------------------|------|-------|------------------------------------|
| 31-24 | MACSRCADDR_23_16 | R/W  | 0h    | Source Address bits 23:16 (byte 2) |
| 23-16 | MACSRCADDR_31_23 | R/W  | 0h    | Source Address bits 31:23 (byte 3) |
| 15-8  | MACSRCADDR_39_32 | R/W  | 0h    | Source Address bits 39:32 (byte 4) |
| 7-0   | MACSRCADDR_47_40 | R/W  | 0h    | Source Address bits 47:40 (byte 5) |

#### 14.5.6.44 P2\_SEND\_PERCENT Register (offset = 228h) [reset = 0h]

P2\_SEND\_PERCENT is shown in [Figure 14-164](#) and described in [Table 14-179](#).

##### CPSW PORT 2 TRANSMIT QUEUE SEND PERCENTAGES

**Figure 14-164. P2\_SEND\_PERCENT Register**

|          |    |    |    |                   |    |    |    |
|----------|----|----|----|-------------------|----|----|----|
| 31       | 30 | 29 | 28 | 27                | 26 | 25 | 24 |
| Reserved |    |    |    |                   |    |    |    |
| 23       | 22 | 21 | 20 | 19                | 18 | 17 | 16 |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI3_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |
| 15       | 14 | 13 | 12 | 11                | 10 | 9  | 8  |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI2_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |
| 7        | 6  | 5  | 4  | 3                 | 2  | 1  | 0  |
| Reserved |    |    |    |                   |    |    |    |
|          |    |    |    | PRI1_SEND_PERCENT |    |    |    |
|          |    |    |    | R/W-0h            |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-179. P2\_SEND\_PERCENT Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 22-16 | PRI3_SEND_PERCENT | R/W  | 0h    | Priority 3 Transmit Percentage - This percentage value is sent from FIFO priority 3 (maximum) when the p1_pri3_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 3 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |
| 14-8  | PRI2_SEND_PERCENT | R/W  | 0h    | Priority 2 Transmit Percentage - This percentage value is sent from FIFO priority 2 (maximum) when the p1_pri2_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 2 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |
| 6-0   | PRI1_SEND_PERCENT | R/W  | 0h    | Priority 1 Transmit Percentage - This percentage value is sent from FIFO priority 1 (maximum) when the p1_pri1_shape_en is set (queue shaping enabled).<br>This is the percentage of the wire that packets from priority 1 receive (which includes interpacket gap and preamble bytes).<br>If shaping is enabled on this queue then this value must be between zero and 0d100 (not inclusive). |

**14.5.6.45 P2\_RX\_DSCP\_PRI\_MAP0 Register (offset = 230h) [reset = 0h]**

P2\_RX\_DSCP\_PRI\_MAP0 is shown in [Figure 14-165](#) and described in [Table 14-180](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0

**Figure 14-165. P2\_RX\_DSCP\_PRI\_MAP0 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI7   |    | Reserved |    | PRI6   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI5   |    | Reserved |    | PRI4   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI3   |    | Reserved |    | PRI2   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI1   |    | Reserved |    | PRI0   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-180. P2\_RX\_DSCP\_PRI\_MAP0 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI7  | R/W  | 0h    | Priority<br>7 - A packet TOS of 0d7 is mapped to this received packet priority. |
| 26-24 | PRI6  | R/W  | 0h    | Priority<br>6 - A packet TOS of 0d6 is mapped to this received packet priority. |
| 22-20 | PRI5  | R/W  | 0h    | Priority<br>5 - A packet TOS of 0d5 is mapped to this received packet priority. |
| 18-16 | PRI4  | R/W  | 0h    | Priority<br>4 - A packet TOS of 0d4 is mapped to this received packet priority. |
| 14-12 | PRI3  | R/W  | 0h    | Priority<br>3 - A packet TOS of 0d3 is mapped to this received packet priority. |
| 10-8  | PRI2  | R/W  | 0h    | Priority<br>2 - A packet TOS of 0d2 is mapped to this received packet priority. |
| 6-4   | PRI1  | R/W  | 0h    | Priority<br>1 - A packet TOS of 0d1 is mapped to this received packet priority. |
| 2-0   | PRI0  | R/W  | 0h    | Priority<br>0 - A packet TOS of 0d0 is mapped to this received packet priority. |

**14.5.6.46 P2\_RX\_DSCP\_PRI\_MAP1 Register (offset = 234h) [reset = 0h]**

 P2\_RX\_DSCP\_PRI\_MAP1 is shown in [Figure 14-166](#) and described in [Table 14-181](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1

**Figure 14-166. P2\_RX\_DSCP\_PRI\_MAP1 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI15  |    | Reserved |    | PRI14  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI13  |    | Reserved |    | PRI12  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI11  |    | Reserved |    | PRI10  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI9   |    | Reserved |    | PRI8   |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-181. P2\_RX\_DSCP\_PRI\_MAP1 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI15 | R/W  | 0h    | Priority<br>15 - A packet TOS of 0d15 is mapped to this received packet priority. |
| 26-24 | PRI14 | R/W  | 0h    | Priority<br>14 - A packet TOS of 0d14 is mapped to this received packet priority. |
| 22-20 | PRI13 | R/W  | 0h    | Priority<br>13 - A packet TOS of 0d13 is mapped to this received packet priority. |
| 18-16 | PRI12 | R/W  | 0h    | Priority<br>12 - A packet TOS of 0d12 is mapped to this received packet priority. |
| 14-12 | PRI11 | R/W  | 0h    | Priority<br>11 - A packet TOS of 0d11 is mapped to this received packet priority. |
| 10-8  | PRI10 | R/W  | 0h    | Priority<br>10 - A packet TOS of 0d10 is mapped to this received packet priority. |
| 6-4   | PRI9  | R/W  | 0h    | Priority<br>9 - A packet TOS of 0d9 is mapped to this received packet priority.   |
| 2-0   | PRI8  | R/W  | 0h    | Priority<br>8 - A packet TOS of 0d8 is mapped to this received packet priority.   |

**14.5.6.47 P2\_RX\_DSCP\_PRI\_MAP2 Register (offset = 238h) [reset = 0h]**

P2\_RX\_DSCP\_PRI\_MAP2 is shown in [Figure 14-167](#) and described in [Table 14-182](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2

**Figure 14-167. P2\_RX\_DSCP\_PRI\_MAP2 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI23  |    | Reserved |    | PRI22  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI21  |    | Reserved |    | PRI20  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI19  |    | Reserved |    | PRI18  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI17  |    | Reserved |    | PRI16  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-182. P2\_RX\_DSCP\_PRI\_MAP2 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI23 | R/W  | 0h    | Priority<br>23 - A packet TOS of 0d23 is mapped to this received packet priority. |
| 26-24 | PRI22 | R/W  | 0h    | Priority<br>22 - A packet TOS of 0d22 is mapped to this received packet priority. |
| 22-20 | PRI21 | R/W  | 0h    | Priority<br>21 - A packet TOS of 0d21 is mapped to this received packet priority. |
| 18-16 | PRI20 | R/W  | 0h    | Priority<br>20 - A packet TOS of 0d20 is mapped to this received packet priority. |
| 14-12 | PRI19 | R/W  | 0h    | Priority<br>19 - A packet TOS of 0d19 is mapped to this received packet priority. |
| 10-8  | PRI18 | R/W  | 0h    | Priority<br>18 - A packet TOS of 0d18 is mapped to this received packet priority. |
| 6-4   | PRI17 | R/W  | 0h    | Priority<br>17 - A packet TOS of 0d17 is mapped to this received packet priority. |
| 2-0   | PRI16 | R/W  | 0h    | Priority<br>16 - A packet TOS of 0d16 is mapped to this received packet priority. |

#### 14.5.6.48 P2\_RX\_DSCP\_PRI\_MAP3 Register (offset = 23Ch) [reset = 0h]

P2\_RX\_DSCP\_PRI\_MAP3 is shown in [Figure 14-168](#) and described in [Table 14-183](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3

**Figure 14-168. P2\_RX\_DSCP\_PRI\_MAP3 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI31  |    | Reserved |    | PRI30  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI29  |    | Reserved |    | PRI28  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI27  |    | Reserved |    | PRI26  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI25  |    | Reserved |    | PRI24  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-183. P2\_RX\_DSCP\_PRI\_MAP3 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI31 | R/W  | 0h    | Priority<br>31 - A packet TOS of 0d31 is mapped to this received packet priority. |
| 26-24 | PRI30 | R/W  | 0h    | Priority<br>30 - A packet TOS of 0d30 is mapped to this received packet priority. |
| 22-20 | PRI29 | R/W  | 0h    | Priority<br>29 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 18-16 | PRI28 | R/W  | 0h    | Priority<br>28 - A packet TOS of 0d28 is mapped to this received packet priority. |
| 14-12 | PRI27 | R/W  | 0h    | Priority<br>27 - A packet TOS of 0d27 is mapped to this received packet priority. |
| 10-8  | PRI26 | R/W  | 0h    | Priority<br>26 - A packet TOS of 0d26 is mapped to this received packet priority. |
| 6-4   | PRI25 | R/W  | 0h    | Priority<br>25 - A packet TOS of 0d25 is mapped to this received packet priority. |
| 2-0   | PRI24 | R/W  | 0h    | Priority<br>24 - A packet TOS of 0d24 is mapped to this received packet priority. |

**14.5.6.49 P2\_RX\_DSCP\_PRI\_MAP4 Register (offset = 240h) [reset = 0h]**

P2\_RX\_DSCP\_PRI\_MAP4 is shown in [Figure 14-169](#) and described in [Table 14-184](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4

**Figure 14-169. P2\_RX\_DSCP\_PRI\_MAP4 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI39  |    | Reserved |    | PRI38  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI37  |    | Reserved |    | PRI36  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI35  |    | Reserved |    | PRI34  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI33  |    | Reserved |    | PRI32  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-184. P2\_RX\_DSCP\_PRI\_MAP4 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI39 | R/W  | 0h    | Priority<br>39 - A packet TOS of 0d39 is mapped to this received packet priority. |
| 26-24 | PRI38 | R/W  | 0h    | Priority<br>38 - A packet TOS of 0d38 is mapped to this received packet priority. |
| 22-20 | PRI37 | R/W  | 0h    | Priority<br>37 - A packet TOS of 0d37 is mapped to this received packet priority. |
| 18-16 | PRI36 | R/W  | 0h    | Priority<br>36 - A packet TOS of 0d36 is mapped to this received packet priority. |
| 14-12 | PRI35 | R/W  | 0h    | Priority<br>35 - A packet TOS of 0d35 is mapped to this received packet priority. |
| 10-8  | PRI34 | R/W  | 0h    | Priority<br>34 - A packet TOS of 0d34 is mapped to this received packet priority. |
| 6-4   | PRI33 | R/W  | 0h    | Priority<br>33 - A packet TOS of 0d33 is mapped to this received packet priority. |
| 2-0   | PRI32 | R/W  | 0h    | Priority<br>32 - A packet TOS of 0d32 is mapped to this received packet priority. |

**14.5.6.50 P2\_RX\_DSCP\_PRI\_MAP5 Register (offset = 244h) [reset = 0h]**

 P2\_RX\_DSCP\_PRI\_MAP5 is shown in [Figure 14-170](#) and described in [Table 14-185](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5

**Figure 14-170. P2\_RX\_DSCP\_PRI\_MAP5 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI47  |    | Reserved |    | PRI46  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI45  |    | Reserved |    | PRI44  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI43  |    | Reserved |    | PRI42  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI41  |    | Reserved |    | PRI40  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-185. P2\_RX\_DSCP\_PRI\_MAP5 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI47 | R/W  | 0h    | Priority<br>47 - A packet TOS of 0d47 is mapped to this received packet priority. |
| 26-24 | PRI46 | R/W  | 0h    | Priority<br>46 - A packet TOS of 0d46 is mapped to this received packet priority. |
| 22-20 | PRI45 | R/W  | 0h    | Priority<br>45 - A packet TOS of 0d45 is mapped to this received packet priority. |
| 18-16 | PRI44 | R/W  | 0h    | Priority<br>44 - A packet TOS of 0d44 is mapped to this received packet priority. |
| 14-12 | PRI43 | R/W  | 0h    | Priority<br>43 - A packet TOS of 0d43 is mapped to this received packet priority. |
| 10-8  | PRI42 | R/W  | 0h    | Priority<br>42 - A packet TOS of 0d42 is mapped to this received packet priority. |
| 6-4   | PRI41 | R/W  | 0h    | Priority<br>41 - A packet TOS of 0d41 is mapped to this received packet priority. |
| 2-0   | PRI40 | R/W  | 0h    | Priority<br>40 - A packet TOS of 0d40 is mapped to this received packet priority. |

**14.5.6.51 P2\_RX\_DSCP\_PRI\_MAP6 Register (offset = 248h) [reset = 0h]**

P2\_RX\_DSCP\_PRI\_MAP6 is shown in [Figure 14-171](#) and described in [Table 14-186](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6

**Figure 14-171. P2\_RX\_DSCP\_PRI\_MAP6 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI55  |    | Reserved |    | PRI54  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI53  |    | Reserved |    | PRI52  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI51  |    | Reserved |    | PRI50  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI49  |    | Reserved |    | PRI48  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-186. P2\_RX\_DSCP\_PRI\_MAP6 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI55 | R/W  | 0h    | Priority<br>55 - A packet TOS of 0d55 is mapped to this received packet priority. |
| 26-24 | PRI54 | R/W  | 0h    | Priority<br>54 - A packet TOS of 0d54 is mapped to this received packet priority. |
| 22-20 | PRI53 | R/W  | 0h    | Priority<br>53 - A packet TOS of 0d53 is mapped to this received packet priority. |
| 18-16 | PRI52 | R/W  | 0h    | Priority<br>52 - A packet TOS of 0d52 is mapped to this received packet priority. |
| 14-12 | PRI51 | R/W  | 0h    | Priority<br>51 - A packet TOS of 0d51 is mapped to this received packet priority. |
| 10-8  | PRI50 | R/W  | 0h    | Priority<br>50 - A packet TOS of 0d50 is mapped to this received packet priority. |
| 6-4   | PRI49 | R/W  | 0h    | Priority<br>49 - A packet TOS of 0d49 is mapped to this received packet priority. |
| 2-0   | PRI48 | R/W  | 0h    | Priority<br>48 - A packet TOS of 0d48 is mapped to this received packet priority. |

#### 14.5.6.52 P2\_RX\_DSCP\_PRI\_MAP7 Register (offset = 24Ch) [reset = 0h]

P2\_RX\_DSCP\_PRI\_MAP7 is shown in [Figure 14-172](#) and described in [Table 14-187](#).

CPSW PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7

**Figure 14-172. P2\_RX\_DSCP\_PRI\_MAP7 Register**

| 31       | 30 | 29     | 28 | 27       | 26 | 25     | 24 |
|----------|----|--------|----|----------|----|--------|----|
| Reserved |    | PRI63  |    | Reserved |    | PRI62  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 23       | 22 | 21     | 20 | 19       | 18 | 17     | 16 |
| Reserved |    | PRI61  |    | Reserved |    | PRI60  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 15       | 14 | 13     | 12 | 11       | 10 | 9      | 8  |
| Reserved |    | PRI59  |    | Reserved |    | PRI58  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |
| 7        | 6  | 5      | 4  | 3        | 2  | 1      | 0  |
| Reserved |    | PRI57  |    | Reserved |    | PRI56  |    |
|          |    | R/W-0h |    |          |    | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-187. P2\_RX\_DSCP\_PRI\_MAP7 Register Field Descriptions**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 30-28 | PRI63 | R/W  | 0h    | Priority<br>63 - A packet TOS of 0d63 is mapped to this received packet priority. |
| 26-24 | PRI62 | R/W  | 0h    | Priority<br>62 - A packet TOS of 0d62 is mapped to this received packet priority. |
| 22-20 | PRI61 | R/W  | 0h    | Priority<br>61 - A packet TOS of 0d61 is mapped to this received packet priority. |
| 18-16 | PRI60 | R/W  | 0h    | Priority<br>60 - A packet TOS of 0d60 is mapped to this received packet priority. |
| 14-12 | PRI59 | R/W  | 0h    | Priority<br>59 - A packet TOS of 0d59 is mapped to this received packet priority. |
| 10-8  | PRI58 | R/W  | 0h    | Priority<br>58 - A packet TOS of 0d58 is mapped to this received packet priority. |
| 6-4   | PRI57 | R/W  | 0h    | Priority<br>57 - A packet TOS of 0d57 is mapped to this received packet priority. |
| 2-0   | PRI56 | R/W  | 0h    | Priority<br>56 - A packet TOS of 0d56 is mapped to this received packet priority. |

#### 14.5.7 CPSW\_SL Registers

[Table 14-188](#) lists the memory-mapped registers for the CPSW\_SL. All register offset addresses not listed in [Table 14-188](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-188. CPSW\_SL REGISTERS**

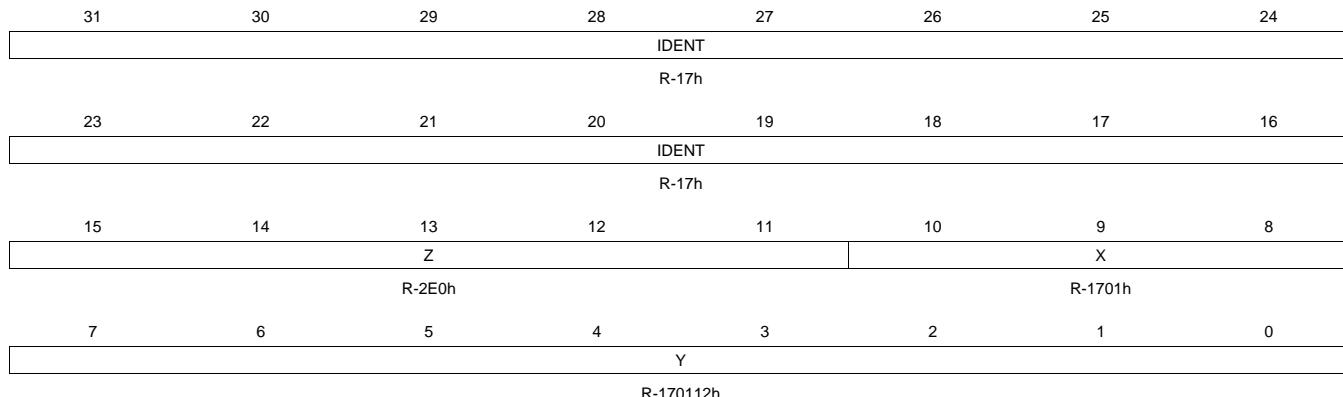
| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b>   | <b>Section</b>                    |
|---------------|----------------|--|-----------------------------------|
| 0h            | IDVER          | CPGMAC_SL ID/VERSION REGISTER                                    | <a href="#">Section 14.5.7.1</a>  |
| 4h            | MACCONTROL     | CPGMAC_SL MAC CONTROL REGISTER                                   | <a href="#">Section 14.5.7.2</a>  |
| 8h            | MACSTATUS      | CPGMAC_SL MAC STATUS REGISTER                                    | <a href="#">Section 14.5.7.3</a>  |
| Ch            | SOFT_RESET     | CPGMAC_SL SOFT RESET REGISTER                                    | <a href="#">Section 14.5.7.4</a>  |
| 10h           | RX_MAXLEN      | CPGMAC_SL RX MAXIMUM LENGTH REGISTER                             | <a href="#">Section 14.5.7.5</a>  |
| 14h           | BOFFTEST       | CPGMAC_SL BACKOFF TEST REGISTER                                  | <a href="#">Section 14.5.7.6</a>  |
| 18h           | RX_PAUSE       | CPGMAC_SL RECEIVE PAUSE TIMER REGISTER                           | <a href="#">Section 14.5.7.7</a>  |
| 1Ch           | TX_PAUSE       | CPGMAC_SL TRANSMIT PAUSE TIMER REGISTER                          | <a href="#">Section 14.5.7.8</a>  |
| 20h           | EMCONTROL      | CPGMAC_SL EMULATION CONTROL REGISTER                             | <a href="#">Section 14.5.7.9</a>  |
| 24h           | RX_PRI_MAP     | CPGMAC_SL RX PKT PRIORITY TO HEADER<br>PRIORITY MAPPING REGISTER | <a href="#">Section 14.5.7.10</a> |
| 28h           | TX_GAP         | TRANSMIT INTER-PACKET GAP REGISTER                               | <a href="#">Section 14.5.7.11</a> |

#### 14.5.7.1 IDVER Register (offset = 0h) [reset = 170112h]

IDVER is shown in [Figure 14-173](#) and described in [Table 14-189](#).

##### CPGMAC\_SL ID/VERSION REGISTER

**Figure 14-173. IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-189. IDVER Register Field Descriptions**

| Bit   | Field | Type | Reset   | Description             |
|-------|-------|------|---------|-------------------------|
| 31-16 | IDENT | R    | 17h     | Rx Identification Value |
| 15-11 | Z     | R    | 2E0h    | Rx Z value (X.Y.Z)      |
| 10-8  | X     | R    | 1701h   | Rx X value (major)      |
| 7-0   | Y     | R    | 170112h | Rx Y value (minor)      |

#### 14.5.7.2 MACCONTROL Register (offset = 4h) [reset = 0h]

MACCONTROL is shown in [Figure 14-174](#) and described in [Table 14-190](#).

##### CPGMAC\_SL MAC CONTROL REGISTER

**Figure 14-174. MACCONTROL Register**

| 31        | 30        | 29                  | 28         | 27              | 26        | 25       | 24         |
|-----------|-----------|---------------------|------------|-----------------|-----------|----------|------------|
| Reserved  |           |                     |            |                 |           |          | RX_CMF_EN  |
| R-0h      |           |                     |            |                 |           |          | R/W-0h     |
| 23        | 22        | 21                  | 20         | 19              | 18        | 17       | 16         |
| RX_CSF_EN | RX_CEF_EN | TX_SHORT_GAP_LIM_EN | Reserved   | EXT_EN          | GIG_FORCE | IFCTL_B  |            |
| R/W-0h    | R/W-0h    | R/W-0h              | R-0h       | R/W-0h          | R/W-0h    | R/W-0h   | R/W-0h     |
| 15        | 14        | 13                  | 12         | 11              | 10        | 9        | 8          |
| IFCTL_A   |           | Reserved            | CMD_IDLE   | TX_SHORT_GAP_EN | Reserved  |          |            |
| R/W-0h    |           | R-0h                | R/W-0h     | R/W-0h          |           | R-0h     |            |
| 7         | 6         | 5                   | 4          | 3               | 2         | 1        | 0          |
| GIG       | TX_PACE   | GMII_EN             | TX_FLOW_EN | RX_FLOW_EN      | MTEST     | LOOPBACK | FULLDUPLEX |
| R/W-0h    | R/W-0h    | R/W-0h              | R/W-0h     | R/W-0h          | R/W-0h    | R/W-0h   | R/W-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-190. MACCONTROL Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description   |
|-------|---------------------|------|-------|---|
| 31-25 | Reserved            | R    | 0h    |   |
| 24    | RX_CMF_EN           | R/W  | 0h    | RX Copy MAC Control Frames Enable - Enables MAC control frames to be transferred to memory.<br>MAC control frames are normally acted upon (if enabled), but not copied to memory.<br>MAC control frames that are pause frames will be acted upon if enabled in the MacControl register, regardless of the value of rx_cmf_en.<br>Frames transferred to memory due to rx_cmf_en will have the control bit set in their EOP buffer descriptor.<br>0 - MAC control frames are filtered (but acted upon if enabled).<br>1 - MAC control frames are transferred to memory. |
| 23    | RX_CSF_EN           | R/W  | 0h    | RX Copy Short Frames Enable - Enables frames or fragments shorter than 64 bytes to be copied to memory.<br>Frames transferred to memory due to rx_csf_en will have the fragment or undersized bit set in their receive footer.<br>Fragments are short frames that contain CRC/align/code errors and undersized are short frames without errors.<br>0 - Short frames are filtered<br>1 - Short frames are transferred to memory.   |
| 22    | RX_CEF_EN           | R/W  | 0h    | RX Copy Error Frames Enable - Enables frames containing errors to be transferred to memory.<br>The appropriate error bit will be set in the frame receive footer.<br>Frames containing errors will be filtered when rx_cef_en is not set.<br>0 - Frames containing errors are filtered.<br>1 - Frames containing errors are transferred to memory.  |
| 21    | TX_SHORT_GAP_LIM_EN | R/W  | 0h    | Transmit Short Gap Limit Enable When set this bit limits the number of short gap packets transmitted to 100ppm.<br>Each time a short gap packet is sent, a counter is loaded with 10,000 and decremented on each wireside clock.<br>Another short gap packet will not be sent out until the counter decrements to zero.<br>This mode is included to preclude the host from filling up the FIFO and sending every packet out with short gap which would violate the maximum number of packets per second allowed.  |
| 20-19 | Reserved            | R    | 0h    |   |

**Table 14-190. MACCONTROL Register Field Descriptions (continued)**

| Bit   | Field           | Type | Reset | Description  |
|-------|-----------------|------|-------|--|
| 18    | EXT_EN          | R/W  | 0h    | Control Enable - Enables the fullduplex and gigabit mode to be selected from the FULLDUPLEX_IN and GIG_IN input signals and not from the fullduplex and gig bits in this register.<br>The FULLDUPLEX_MODE bit reflects the actual fullduplex mode selected.  |
| 17    | GIG_FORCE       | R/W  | 0h    | Gigabit Mode Force - This bit is used to force the CPGMAC_SL into gigabit mode if the input GMII_MTCLK has been stopped by the PHY.  |
| 16    | IFCTL_B         | R/W  | 0h    | Connects to the speed_in input of the respective RMII gasket.<br>When using RMII mode:<br>0 - 10Mbps operation<br>1 - 100Mbps operation  |
| 15    | IFCTL_A         | R/W  | 0h    | Connects to the speed_in input of the respective RMII gasket.<br>When using RMII mode:<br>0 - 10Mbps operation<br>1 - 100Mbps operation  |
| 14-12 | Reserved        | R    | 0h    |  |
| 11    | CMD_IDLE        | R/W  | 0h    | Command Idle<br>0 - Idle not commanded<br>1 - Idle Commanded (read idle in MacStatus)  |
| 10    | TX_SHORT_GAP_EN | R/W  | 0h    | Transmit Short Gap Enable<br>0 - Transmit with a short IPG is disabled<br>1 - Transmit with a short IPG (when TX_SHORT_GAP input is asserted) is enabled.  |
| 9-8   | Reserved        | R    | 0h    |  |
| 7     | GIG             | R/W  | 0h    | Gigabit Mode -<br>0 - 10/100 mode<br>1 - Gigabit mode (full duplex only) The GIG_OUT output is the value of this bit.  |
| 6     | TX_PACE         | R/W  | 0h    | Transmit Pacing Enable<br>0 - Transmit Pacing Disabled<br>1 - Transmit Pacing Enabled  |
| 5     | GMII_EN         | R/W  | 0h    | GMII Enable -<br>0 - GMII RX and TX held in reset.<br>1 - GMII RX and TX released from reset.  |
| 4     | TX_FLOW_EN      | R/W  | 0h    | Transmit Flow Control Enable - Determines if incoming pause frames are acted upon in full-duplex mode.<br>Incoming pause frames are not acted upon in half-duplex mode regardless of this bit setting.<br>The RX_MBP_Enable bits determine whether or not received pause frames are transferred to memory.<br>0 - Transmit Flow Control Disabled.<br>Full-duplex mode - Incoming pause frames are not acted upon.<br>1 - Transmit Flow Control Enabled .<br>Full-duplex mode - Incoming pause frames are acted upon. |
| 3     | RX_FLOW_EN      | R/W  | 0h    | Receive Flow Control Enable -<br>0 - Receive Flow Control Disabled Half-duplex mode - No flow control generated collisions are sent.<br>Full-duplex mode - No outgoing pause frames are sent.<br>1 - Receive Flow Control Enabled Half-duplex mode - Collisions are initiated when receive flow control is triggered.<br>Full-duplex mode - Outgoing pause frames are sent when receive flow control is triggered.   |
| 2     | MTEST           | R/W  | 0h    | Manufacturing Test mode - This bit must be set to allow writes to the Backoff_Test and PauseTimer registers.   |
| 1     | LOOPBACK        | R/W  | 0h    | Loop Back Mode - Loopback mode forces internal fullduplex mode regardless of whether the fullduplex bit is set or not.<br>The loopback bit should be changed only when GMII_en is deasserted.<br>0 - Not looped back<br>1 - Loop Back Mode enabled   |

**Table 14-190. MACCONTROL Register Field Descriptions (continued)**

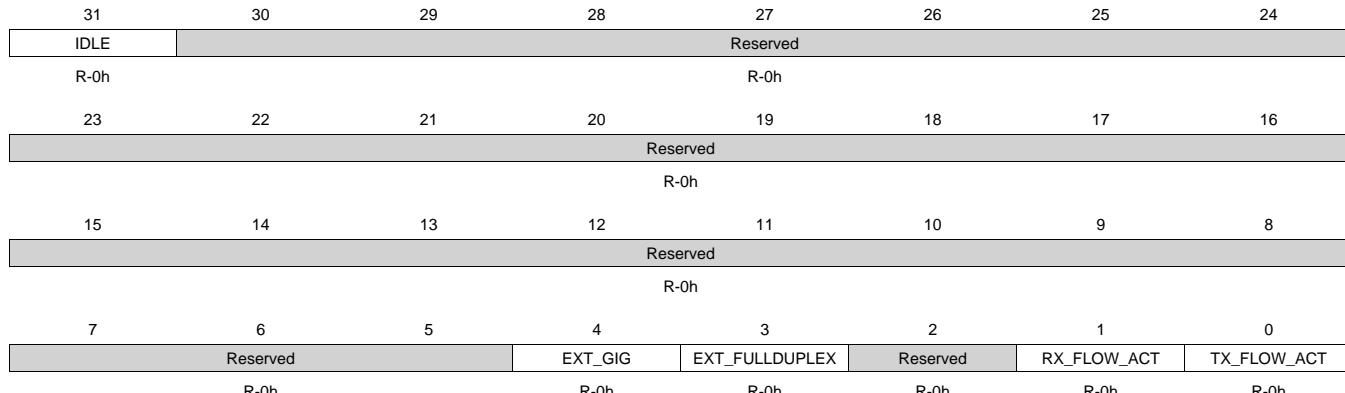
| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 0   | FULLDUPLEX | R/W  | 0h    | Full Duplex mode - Gigabit mode forces fullduplex mode regardless of whether the fullduplex bit is set or not.<br>The FULLDUPLEX_OUT output is the value of this register bit<br>0 - half duplex mode<br>1 - full duplex mode |

#### 14.5.7.3 MACSTATUS Register (offset = 8h) [reset = 0h]

MACSTATUS is shown in [Figure 14-175](#) and described in [Table 14-191](#).

##### CPGMAC\_SL MAC STATUS REGISTER

**Figure 14-175. MACSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-191. MACSTATUS Register Field Descriptions**

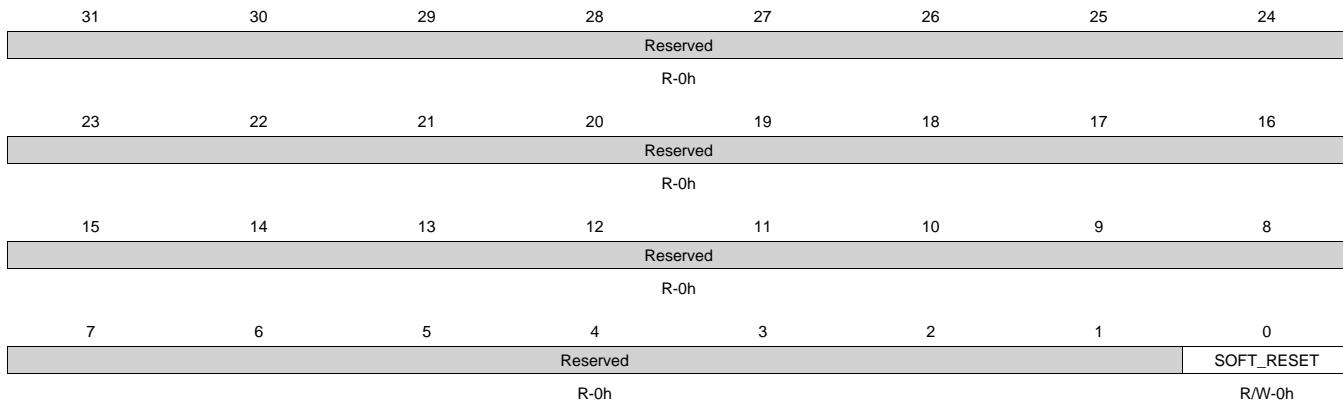
| Bit  | Field          | Type | Reset | Description  |
|------|----------------|------|-------|--|
| 31   | IDLE           | R    | 0h    | CPGMAC_SL IDLE - The CPGMAC_SL is in the idle state (valid after an idle command)<br>0 - The CPGMAC_SL is not in the idle state.<br>1 - The CPGMAC_SL is in the idle state.  |
| 30-5 | Reserved       | R    | 0h    |  |
| 4    | EXT_GIG        | R    | 0h    | External GIG - This is the value of the EXT_GIG input bit.   |
| 3    | EXT_FULLDUPLEX | R    | 0h    | External Fullduplex - This is the value of the EXT_FULLDUPLEX input bit.   |
| 2    | Reserved       | R    | 0h    |  |
| 1    | RX_FLOW_ACT    | R    | 0h    | Receive Flow Control Active - When asserted, indicates that receive flow control is enabled and triggered.   |
| 0    | TX_FLOW_ACT    | R    | 0h    | Transmit Flow Control Active - When asserted, this bit indicates that the pause time period is being observed for a received pause frame. No new transmissions will begin while this bit is asserted except for the transmission of pause frames.<br>Any transmission in progress when this bit is asserted will complete. |

#### 14.5.7.4 SOFT\_RESET Register (offset = Ch) [reset = 0h]

SOFT\_RESET is shown in [Figure 14-176](#) and described in [Table 14-192](#).

##### CPGMAC\_SL SOFT RESET REGISTER

**Figure 14-176. SOFT\_RESET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-192. SOFT\_RESET Register Field Descriptions**

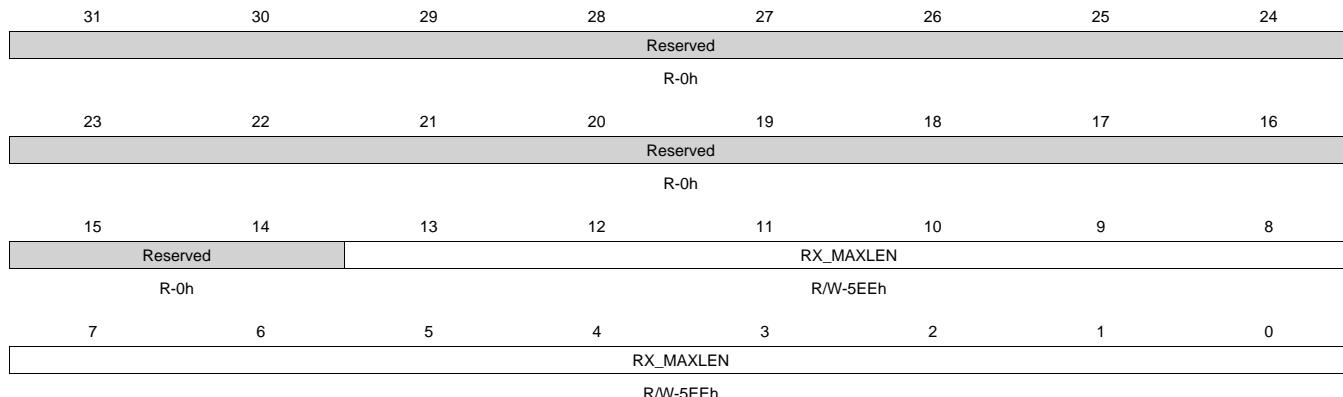
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-1 | Reserved   | R    | 0h    |  |
| 0    | SOFT_RESET | R/W  | 0h    | Software reset - Writing a one to this bit causes the CPGMAC_SL logic to be reset.<br>After writing a one to this bit, it may be polled to determine if the reset has occurred.<br>If a one is read, the reset has not yet occurred.<br>If a zero is read then reset has occurred. |

#### 14.5.7.5 RX\_MAXLEN Register (offset = 10h) [reset = 5EEh]

RX\_MAXLEN is shown in [Figure 14-177](#) and described in [Table 14-193](#).

##### CPGMAC\_SL RX MAXIMUM LENGTH REGISTER

**Figure 14-177. RX\_MAXLEN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-193. RX\_MAXLEN Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description   |
|-------|-----------|------|-------|---|
| 31-14 | Reserved  | R    | 0h    |   |
| 13-0  | RX_MAXLEN | R/W  | 5EEh  | RX Maximum Frame Length - This field determines the maximum length of a received frame.<br>The reset value is 1518 (dec).<br>Frames with byte counts greater than rx_maxlen are long frames.<br>Long frames with no errors are oversized frames.<br>Long frames with CRC, code, or alignment error are jabber frames.<br>The maximum value is 16,383. |

#### 14.5.7.6 BOFFTEST Register (offset = 14h) [reset = 0h]

BOFFTEST is shown in [Figure 14-178](#) and described in [Table 14-194](#).

##### CPGMAC\_SL BACKOFF TEST REGISTER

**Figure 14-178. BOFFTEST Register**

| 31       | 30         | 29 | 28         | 27       | 26 | 25         | 24 |
|----------|------------|----|------------|----------|----|------------|----|
| Reserved |            |    | PACEVAL    |          |    | RNDNUM     |    |
| R-0h     |            |    | R/W-0h     |          |    | R/W-0h     |    |
| 23       | 22         | 21 | 20         | 19       | 18 | 17         | 16 |
|          |            |    | RNDNUM     |          |    | R/W-0h     |    |
| 15       | 14         | 13 | 12         | 11       | 10 | 9          | 8  |
|          | COLL_COUNT |    |            | Reserved |    | TX_BACKOFF |    |
| R-0h     |            |    |            | R-0h     |    | R-0h       |    |
| 7        | 6          | 5  | 4          | 3        | 2  | 1          | 0  |
|          |            |    | TX_BACKOFF |          |    | R-0h       |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-194. BOFFTEST Register Field Descriptions**

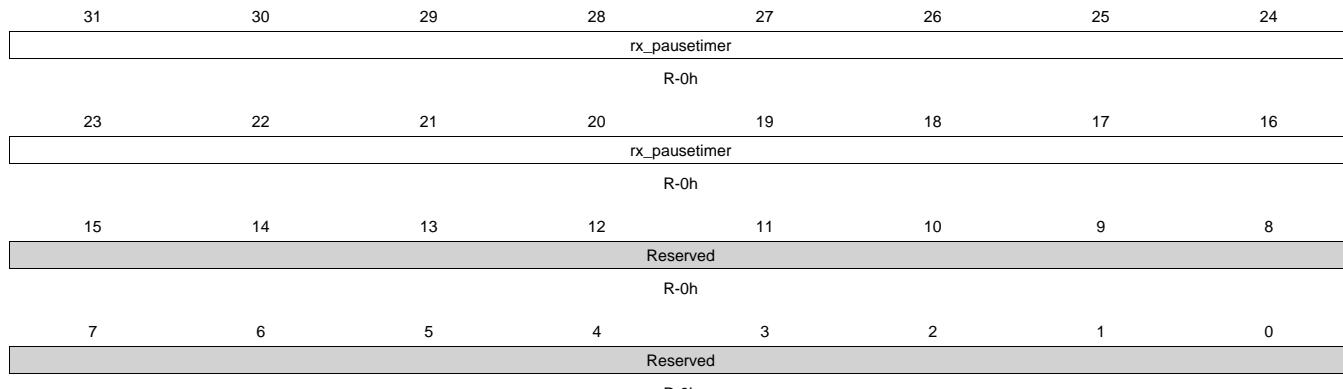
| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 31    | Reserved   | R    | 0h    |   |
| 30-26 | PACEVAL    | R/W  | 0h    | Pacing Register Current Value.<br>A non-zero value in this field indicates that transmit pacing is active. A transmit frame collision or deferral causes paceval to loaded with decimal 31, good frame transmissions (with no collisions or deferrals) cause paceval to be decremented down to zero. When paceval is nonzero, the transmitter delays 4 IPGs between new frame transmissions after each successfully transmitted frame that had no deferrals or collisions.<br>Transmit pacing helps reduce "capture" effects improving overall network bandwidth. |
| 25-16 | RNDNUM     | R/W  | 0h    | Backoff Random Number Generator - This field allows the Backoff Random Number Generator to be read (or written in test mode only). This field can be written only when mtest has previously been set. Reading this field returns the generator's current value.<br>The value is reset to zero and begins counting on the clock after the deassertion of reset.  |
| 15-12 | COLL_COUNT | R    | 0h    | Collision Count - The number of collisions the current frame has experienced.   |
| 11-10 | Reserved   | R    | 0h    |   |
| 9-0   | TX_BACKOFF | R    | 0h    | Backoff Count - This field allows the current value of the backoff counter to be observed for test purposes.<br>This field is loaded automatically according to the backoff algorithm, and is decremented by one for each slot time after the collision.  |

#### 14.5.7.7 RX\_PAUSE Register (offset = 18h) [reset = 0h]

RX\_PAUSE is shown in [Figure 14-179](#) and described in [Table 14-195](#).

##### CPGMAC\_SL RECEIVE PAUSE TIMER REGISTER

**Figure 14-179. RX\_PAUSE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-195. RX\_PAUSE Register Field Descriptions**

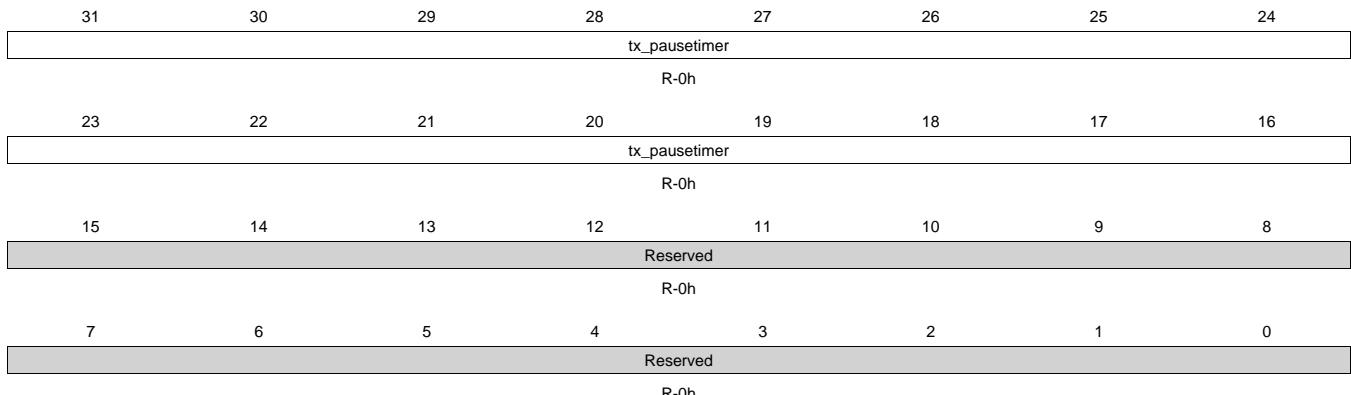
| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31-16 | rx_pausetimer | R    | 0h    | RX Pause Timer Value - This field allows the contents of the receive pause timer to be observed (and written in test mode).<br>The receive pause timer is loaded with 0xFF00 when the CPGMAC_SL sends an outgoing pause frame (with pause time of 0xFFFF).<br>The receive pause timer is decremented at slot time intervals.<br>If the receive pause timer decrements to zero, then another outgoing pause frame will be sent and the load/decrement process will be repeated. |
| 15-0  | Reserved      | R    | 0h    |  |

#### 14.5.7.8 TX\_PAUSE Register (offset = 1Ch) [reset = 0h]

TX\_PAUSE is shown in [Figure 14-180](#) and described in [Table 14-196](#).

##### CPGMAC\_SL TRANSMIT PAUSE TIMER REGISTER

**Figure 14-180. TX\_PAUSE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-196. TX\_PAUSE Register Field Descriptions**

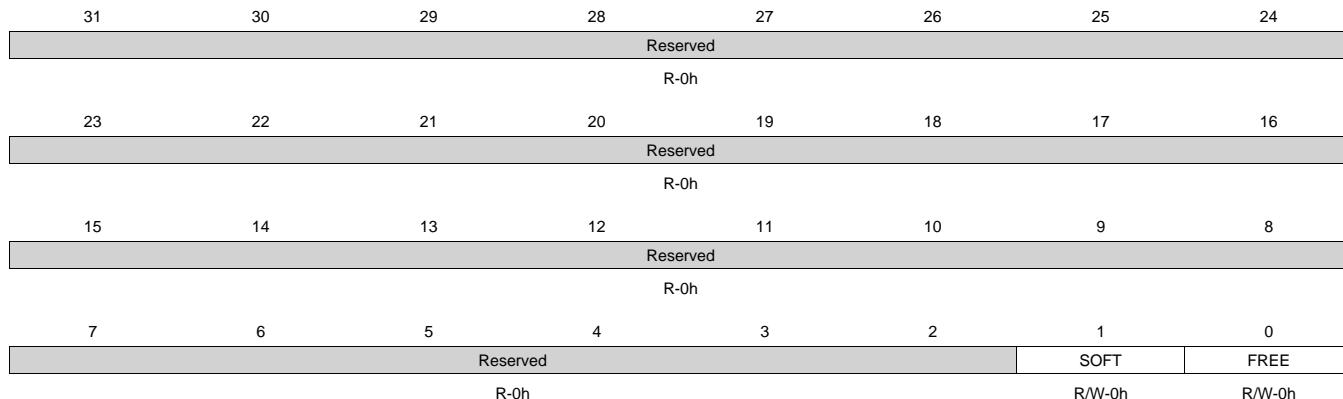
| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31-16 | tx_pausetimer | R    | 0h    | TX Pause Timer Value - This field allows the contents of the transmit pause timer to be observed (and written in test mode). The transmit pause timer is loaded by a received (incoming) pause frame, and then decremented, at slottime intervals, down to zero at which time CPGMAC_SL transmit frames are again enabled. |
| 15-0  | Reserved      | R    | 0h    |  |

#### 14.5.7.9 EMCONTROL Register (offset = 20h) [reset = 0h]

EMCONTROL is shown in [Figure 14-181](#) and described in [Table 14-197](#).

##### CPGMAC\_SL EMULATION CONTROL REGISTER

**Figure 14-181. EMCONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-197. EMCONTROL Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description        |
|------|----------|------|-------|--------------------|
| 31-2 | Reserved | R    | 0h    |                    |
| 1    | SOFT     | R/W  | 0h    | Emulation Soft Bit |
| 0    | FREE     | R/W  | 0h    | Emulation Free Bit |

#### 14.5.7.10 RX\_PRI\_MAP Register (offset = 24h) [reset = 76543210h]

RX\_PRI\_MAP is shown in [Figure 14-182](#) and described in [Table 14-198](#).

CPGMAC\_SL RX PKT PRIORITY TO HEADER PRIORITY MAPPING REGISTER

**Figure 14-182. RX\_PRI\_MAP Register**

| 31        | 30 | 29           | 28 | 27         | 26 | 25            | 24 |
|-----------|----|--------------|----|------------|----|---------------|----|
| Reserved  |    | PRI7         |    | Reserved   |    | PRI6          |    |
| R-0h      |    | R/W-7h       |    | R-Eh       |    | R/W-76h       |    |
| 23        | 22 | 21           | 20 | 19         | 18 | 17            | 16 |
| Reserved  |    | PRI5         |    | Reserved   |    | PRI4          |    |
| R-ECh     |    | R/W-765h     |    | R-ECAh     |    | R/W-7654h     |    |
| 15        | 14 | 13           | 12 | 11         | 10 | 9             | 8  |
| Reserved  |    | PRI3         |    | Reserved   |    | PRI2          |    |
| R-ECA8h   |    | R/W-76543h   |    | R-ECA86h   |    | R/W-765432h   |    |
| 7         | 6  | 5            | 4  | 3          | 2  | 1             | 0  |
| Reserved  |    | PRI1         |    | Reserved   |    | PRI0          |    |
| R-ECA864h |    | R/W-7654321h |    | R-ECA8642h |    | R/W-76543210h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-198. RX\_PRI\_MAP Register Field Descriptions**

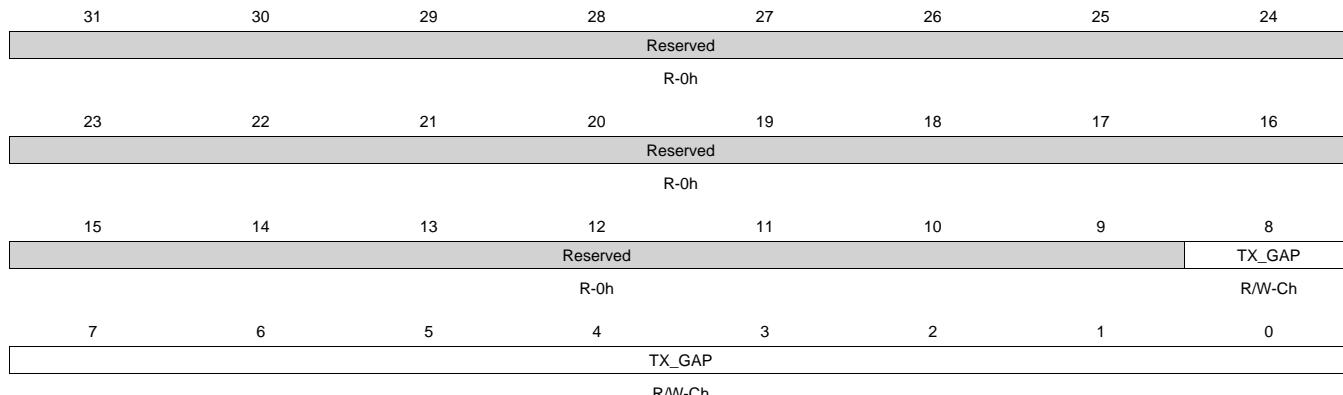
| Bit   | Field    | Type | Reset     | Description   |
|-------|----------|------|-----------|---|
| 31    | Reserved | R    | 0h        |   |
| 30-28 | PRI7     | R/W  | 7h        | Priority<br>7 - A packet priority of 0x7 is mapped (changed) to this value. |
| 27    | Reserved | R    | Eh        |   |
| 26-24 | PRI6     | R/W  | 76h       | Priority<br>6 - A packet priority of 0x6 is mapped (changed) to this value. |
| 23    | Reserved | R    | ECh       |   |
| 22-20 | PRI5     | R/W  | 765h      | Priority<br>5 - A packet priority of 0x5 is mapped (changed) to this value. |
| 19    | Reserved | R    | ECAh      |   |
| 18-16 | PRI4     | R/W  | 7654h     | Priority<br>4 - A packet priority of 0x4 is mapped (changed) to this value. |
| 15    | Reserved | R    | ECA8h     |   |
| 14-12 | PRI3     | R/W  | 76543h    | Priority<br>3 - A packet priority of 0x3 is mapped (changed) to this value. |
| 11    | Reserved | R    | ECA86h    |   |
| 10-8  | PRI2     | R/W  | 765432h   | Priority<br>2 - A packet priority of 0x2 is mapped (changed) to this value. |
| 7     | Reserved | R    | ECA864h   |   |
| 6-4   | PRI1     | R/W  | 7654321h  | Priority<br>1 - A packet priority of 0x1 is mapped (changed) to this value. |
| 3     | Reserved | R    | ECA8642h  |   |
| 2-0   | PRI0     | R/W  | 76543210h | Priority<br>0 - A packet priority of 0x0 is mapped (changed) to this value. |

#### 14.5.7.11 TX\_GAP Register (offset = 28h) [reset = Ch]

TX\_GAP is shown in [Figure 14-183](#) and described in [Table 14-199](#).

##### TRANSMIT INTER-PACKET GAP REGISTER

**Figure 14-183. TX\_GAP Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-199. TX\_GAP Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description               |
|------|----------|------|-------|---------------------------|
| 31-9 | Reserved | R    | 0h    |                           |
| 8-0  | TX_GAP   | R/W  | Ch    | Transmit Inter-Packet Gap |

#### 14.5.8 CPSW\_SS Registers

[Table 14-200](#) lists the memory-mapped registers for the CPSW\_SS. All register offset addresses not listed in [Table 14-200](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-200. CPSW\_SS REGISTERS**

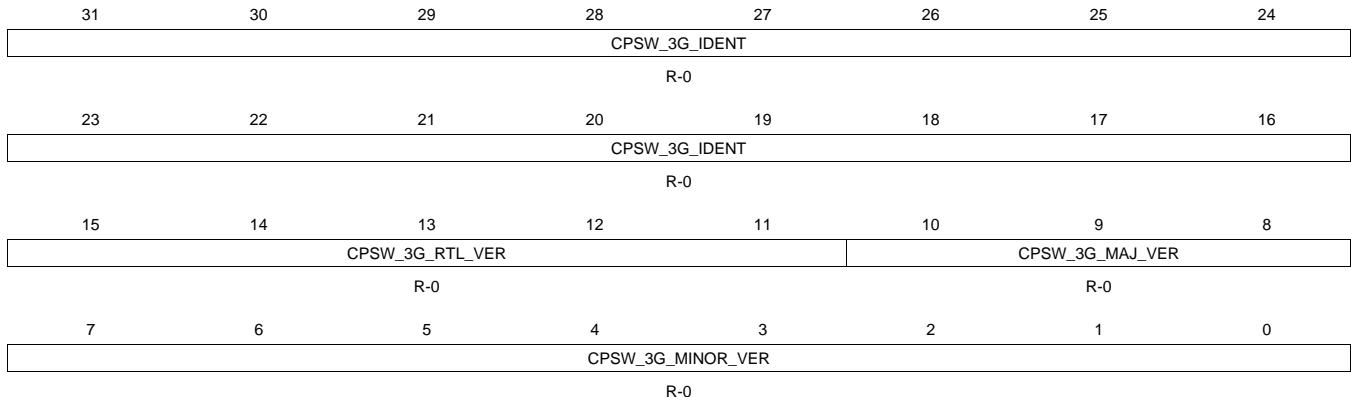
| Offset | Acronym      | Register Name                        | Section                           |
|--------|--------------|--------------------------------------|-----------------------------------|
| 0h     | ID_VER       | ID VERSION REGISTER                  | <a href="#">Section 14.5.8.1</a>  |
| 4h     | CONTROL      | SWITCH CONTROL REGISTER              | <a href="#">Section 14.5.8.2</a>  |
| 8h     | SOFT_RESET   | SOFT RESET REGISTER                  | <a href="#">Section 14.5.8.3</a>  |
| Ch     | STAT_PORT_EN | STATISTICS PORT ENABLE REGISTER      | <a href="#">Section 14.5.8.4</a>  |
| 10h    | PTYPE        | TRANSMIT PRIORITY TYPE REGISTER      | <a href="#">Section 14.5.8.5</a>  |
| 14h    | SOFT_IDLE    | SOFTWARE IDLE                        | <a href="#">Section 14.5.8.6</a>  |
| 18h    | THRU_RATE    | THROUGHPUT RATE                      | <a href="#">Section 14.5.8.7</a>  |
| 1Ch    | GAP_THRESH   | CPGMAC_SL SHORT GAP THRESHOLD        | <a href="#">Section 14.5.8.8</a>  |
| 20h    | TX_START_WDS | TRANSMIT START WORDS                 | <a href="#">Section 14.5.8.9</a>  |
| 24h    | FLOW_CONTROL | FLOW CONTROL                         | <a href="#">Section 14.5.8.10</a> |
| 28h    | VLAN_LTYPE   | LTYPE1 AND LTYPE 2 REGISTER          | <a href="#">Section 14.5.8.11</a> |
| 2Ch    | TS_LTYPE     | VLAN_LTYPE1 AND VLAN_LTYPE2 REGISTER | <a href="#">Section 14.5.8.12</a> |
| 30h    | DLR_LTYPE    | DLR LTYPE REGISTER                   | <a href="#">Section 14.5.8.13</a> |

#### 14.5.8.1 ID\_VER Register (offset = 0h) [reset = 190112h]

ID\_VER is shown in [Figure 14-184](#) and described in [Table 14-201](#).

##### ID VERSION REGISTER

**Figure 14-184. ID\_VER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-201. ID\_VER Register Field Descriptions**

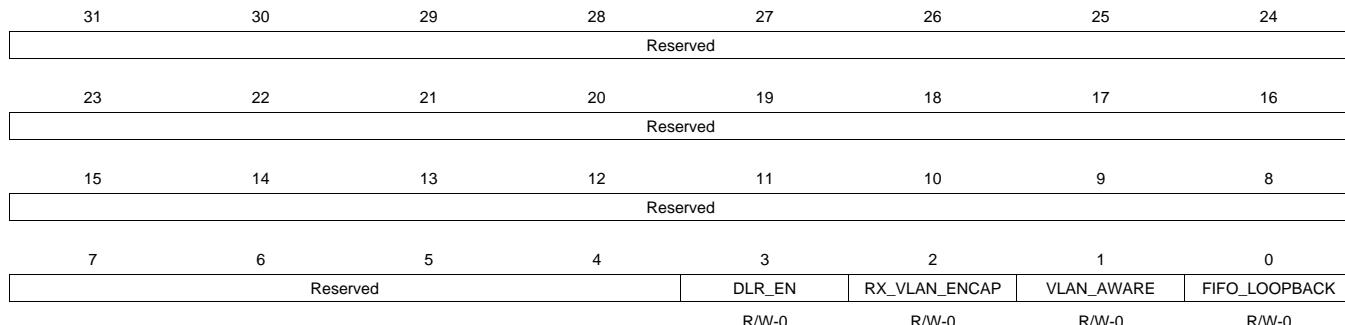
| Bit   | Field             | Type | Reset | Description             |
|-------|-------------------|------|-------|-------------------------|
| 31-16 | CPSW_3G_IDENT     | R-0  | 0     | 3G Identification Value |
| 15-11 | CPSW_3G RTL_VER   | R-0  | 0     | 3G RTL Version Value    |
| 10-8  | CPSW_3G MAJ_VER   | R-0  | 0     | 3G Major Version Value  |
| 7-0   | CPSW_3G_MINOR_VER | R-0  | 0     | 3G Minor Version Value  |

#### 14.5.8.2 CONTROL Register (offset = 4h) [reset = 0h]

CONTROL is shown in [Figure 14-185](#) and described in [Table 14-202](#).

##### SWITCH CONTROL REGISTER

**Figure 14-185. CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-202. CONTROL Register Field Descriptions**

| Bit | Field         | Type  | Reset | Description  |
|-----|---------------|-------|-------|--|
| 3   | DLR_EN        | R/W-0 | 0     | DLR enable<br>0 - DLR is disabled.<br>DLR packets will not be moved to queue priority 3 and will not be separated out onto dlr_cpdma_ch.<br>1 - DLR is disabled.<br>DLR packets be moved to destination port transmit queue priority 3 and will be separated out onto dlr_cpdma_ch when packet is to egress on port 0. |
| 2   | RX_VLAN_ENCAP | R/W-0 | 0     | Port 0 VLAN Encapsulation (egress):<br>0 - Port 2 receive packets (from 3G) are not VLAN encapsulated.<br>1 - Port 2 receive packets (from 3G) are VLAN encapsulated.  |
| 1   | VLAN_AWARE    | R/W-0 | 0     | VLAN Aware Mode:<br>0 - 3G is in the VLAN unaware mode.<br>1 - 3G is in the VLAN aware mode.   |
| 0   | FIFO_LOOPBACK | R/W-0 | 0     | FIFO Loopback Mode<br>0 - Loopback is disabled<br>1 - FIFO Loopback mode enabled.<br>Each packet received is turned around and sent out on the same port's transmit path.<br>Port 2 receive is fixed on channel zero.<br>The RXSOFOVERRUN statistic will increment for every packet sent in FIFO loopback mode.        |

#### 14.5.8.3 SOFT\_RESET Register (offset = 8h) [reset = 0h]

SOFT\_RESET is shown in [Figure 14-186](#) and described in [Table 14-203](#).

##### SOFT RESET REGISTER

**Figure 14-186. SOFT\_RESET Register**

| 31                     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| SOF<br>T_R<br>ESE<br>T |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/<br>W-<br>0          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-203. SOFT\_RESET Register Field Descriptions**

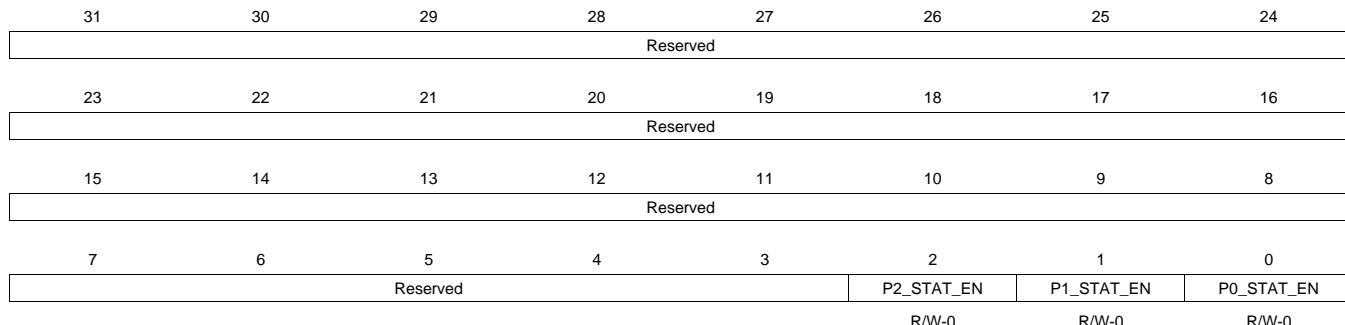
| Bit | Field      | Type  | Reset | Description  |
|-----|------------|-------|-------|--|
| 0   | SOFT_RESET | R/W-0 | 0     | <p>Software reset - Writing a one to this bit causes the 3G logic to be reset.</p> <p>After writing a one to this bit, it may be polled to determine if the reset has occurred.</p> <p>If a one is read, the reset has not yet occurred.</p> <p>If a zero is read then reset has occurred.</p> |

#### 14.5.8.4 STAT\_PORT\_EN Register (offset = Ch) [reset = 0h]

STAT\_PORT\_EN is shown in [Figure 14-187](#) and described in [Table 14-204](#).

##### STATISTICS PORT ENABLE REGISTER

**Figure 14-187. STAT\_PORT\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-204. STAT\_PORT\_EN Register Field Descriptions**

| Bit | Field      | Type  | Reset | Description  |
|-----|------------|-------|-------|--|
| 2   | P2_STAT_EN | R/W-0 | 0     | Port 2 (GMII2 and Port 2 FIFO) Statistics Enable<br>0 - Port 2 statistics are not enabled.<br>1 - Port 2 statistics are enabled.   |
| 1   | P1_STAT_EN | R/W-0 | 0     | Port 1 (GMII1 and Port 1 FIFO) Statistics Enable<br>0 - Port 1 statistics are not enabled.<br>1 - Port 1 statistics are enabled.   |
| 0   | P0_STAT_EN | R/W-0 | 0     | Port 0 Statistics Enable<br>0 - Port 0 statistics are not enabled<br>1 - Port 0 statistics are enabled.<br>FIFO overruns (SOFOVERRUNS) are the only port 0 statistics that are enabled to be kept. |

#### 14.5.8.5 PTYPE Register (offset = 10h) [reset = 0h]

PTYPE is shown in [Figure 14-188](#) and described in [Table 14-205](#).

##### TRANSMIT PRIORITY TYPE REGISTER

**Figure 14-188. PTYPE Register**

| 31       | 30 | 29               | 28               | 27               | 26               | 25               | 24               |
|----------|----|------------------|------------------|------------------|------------------|------------------|------------------|
| Reserved |    |                  |                  |                  |                  |                  |                  |
| 23       | 22 | 21               | 20               | 19               | 18               | 17               | 16               |
| Reserved |    | P2_PRI3_SHAPE_EN | P2_PRI2_SHAPE_EN | P2_PRI1_SHAPE_EN | P1_PRI3_SHAPE_EN | P1_PRI2_SHAPE_EN | P1_PRI1_SHAPE_EN |
|          |    | R/W-0            | R/W-0            | R/W-0            | R/W-0            | R/W-0            | R/W-0            |
| 15       | 14 | 13               | 12               | 11               | 10               | 9                | 8                |
|          |    | Reserved         |                  | P2_PTYPE_ESC     | P1_PTYPE_ESC     | P0_PTYPE_ESC     |                  |
|          |    |                  |                  | R/W-0            | R/W-0            | R/W-0            |                  |
| 7        | 6  | 5                | 4                | 3                | 2                | 1                | 0                |
|          |    | Reserved         |                  | ESC_PRI_LD_VAL   |                  |                  |                  |
|          |    |                  |                  |                  | R/W-0            |                  |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-205. PTYPE Register Field Descriptions**

| Bit | Field            | Type  | Reset | Description  |
|-----|------------------|-------|-------|--|
| 21  | P2_PRI3_SHAPE_EN | R/W-0 | 0     | Port 2 Queue Priority 3 Transmit Shape Enable - If there is only one shaping queue then it must be priority 3.   |
| 20  | P2_PRI2_SHAPE_EN | R/W-0 | 0     | Port 2 Queue Priority 2 Transmit Shape Enable - If there are two shaping queues then they must be priorities 3 and 2.  |
| 19  | P2_PRI1_SHAPE_EN | R/W-0 | 0     | Port 2 Queue Priority 1 Transmit Shape Enable - If there are three shaping queues all three bits should be set.  |
| 18  | P1_PRI3_SHAPE_EN | R/W-0 | 0     | Port 1 Queue Priority 3 Transmit Shape Enable - If there is only one shaping queue then it must be priority 3.   |
| 17  | P1_PRI2_SHAPE_EN | R/W-0 | 0     | Port 1 Queue Priority 2 Transmit Shape Enable- If there are two shaping queues then they must be priorities 3 and 2.   |
| 16  | P1_PRI1_SHAPE_EN | R/W-0 | 0     | Port 1 Queue Priority 1 Transmit Shape Enable- If there are three shaping queues all three bits should be set.   |
| 10  | P2_PTYPE_ESC     | R/W-0 | 0     | Port 2 Priority Type Escalate -<br>0 - Port 2 priority type fixed<br>1 - Port 2 priority type escalate Escalate should not be used with queue shaping.   |
| 9   | P1_PTYPE_ESC     | R/W-0 | 0     | Port 1 Priority Type Escalate -<br>0 - Port 1 priority type fixed<br>1 - Port 1 priority type escalate Escalate should not be used with queue shaping.   |
| 8   | P0_PTYPE_ESC     | R/W-0 | 0     | Port 0 Priority Type Escalate -<br>0 - Port 0 priority type fixed<br>1 - Port 0 priority type escalate Escalate should not be used with queue shaping.   |
| 4-0 | ESC_PRI_LD_VAL   | R/W-0 | 0     | Escalate Priority Load Value When a port is in escalate priority, this is the number of higher priority packets sent before the next lower priority is allowed to send a packet.<br>Escalate priority allows lower priority packets to be sent at a fixed rate relative to the next higher priority. |

#### 14.5.8.6 SOFT\_IDLE Register (offset = 14h) [reset = 0h]

SOFT\_IDLE is shown in Figure 14-189 and described in Table 14-206.

SOFTWARE IDLE

**Figure 14-189. SOFT\_IDLE Register**

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-206. SOFT IDLE Register Field Descriptions**

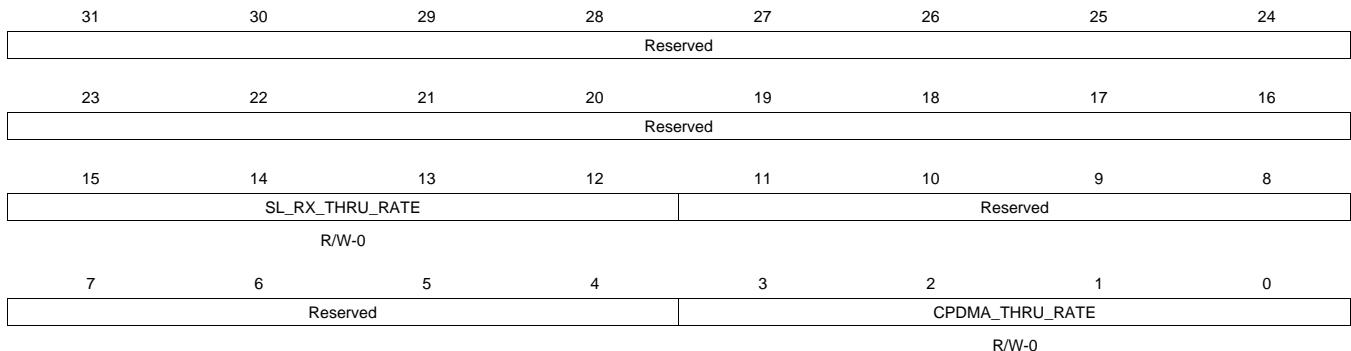
| Bit | Field     | Type  | Reset | Description   |
|-----|-----------|-------|-------|---|
| 0   | SOFT_IDLE | R/W-0 | 0     | Software Idle - Setting this bit causes the switch fabric to stop forwarding packets at the next start of packet. |

#### 14.5.8.7 THRU\_RATE Register (offset = 18h) [reset = 3003h]

THRU\_RATE is shown in [Figure 14-190](#) and described in [Table 14-207](#).

THROUGHPUT RATE

**Figure 14-190. THRU\_RATE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-207. THRU\_RATE Register Field Descriptions**

| Bit   | Field           | Type  | Reset | Description  |
|-------|-----------------|-------|-------|--|
| 15-12 | SL_RX_THRU_RATE | R/W-0 | 0     | CPGMAC_SL Switch FIFO receive through rate.<br>This register value is the maximum throughput of the ethernet ports to the crossbar SCR.<br>The default is one 8-byte word for every 3 CLK periods maximum. |
| 3-0   | CPDMA_THRU_RATE | R/W-0 | 0     | CPDMA Switch FIFO receive through rate.<br>This register value is the maximum throughput of the CPDMA host port to the crossbar SCR.<br>The default is one 8-byte word for every 3 CLK periods maximum.    |

#### 14.5.8.8 GAP\_THRESH Register (offset = 1Ch) [reset = Bh]

GAP\_THRESH is shown in [Figure 14-191](#) and described in [Table 14-208](#).

CPGMAC\_SL SHORT GAP THRESHOLD

**Figure 14-191. GAP\_THRESH Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4          | 3 | 2 | 1     | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|------------|---|---|-------|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   | GAP_THRESH |   |   |       |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |            |   |   | R/W-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-208. GAP\_THRESH Register Field Descriptions**

| Bit | Field      | Type  | Reset | Description   |
|-----|------------|-------|-------|---|
| 4-0 | GAP_THRESH | R/W-0 | 0     | CPGMAC_SL Short Gap Threshold - This is the CPGMAC_SL associated FIFO transmit block usage value for triggering TX_SHORT_GAP. |

#### 14.5.8.9 TX\_START\_WDS Register (offset = 20h) [reset = 20h]

TX\_START\_WDS is shown in [Figure 14-192](#) and described in [Table 14-209](#).

TRANSMIT START WORDS

**Figure 14-192. TX\_START\_WDS Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4            | 3 | 2 | 1 | 0 |       |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|--------------|---|---|---|---|-------|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   | TX_START_WDS |   |   |   |   |       |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |              |   |   |   |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-209. TX\_START\_WDS Register Field Descriptions**

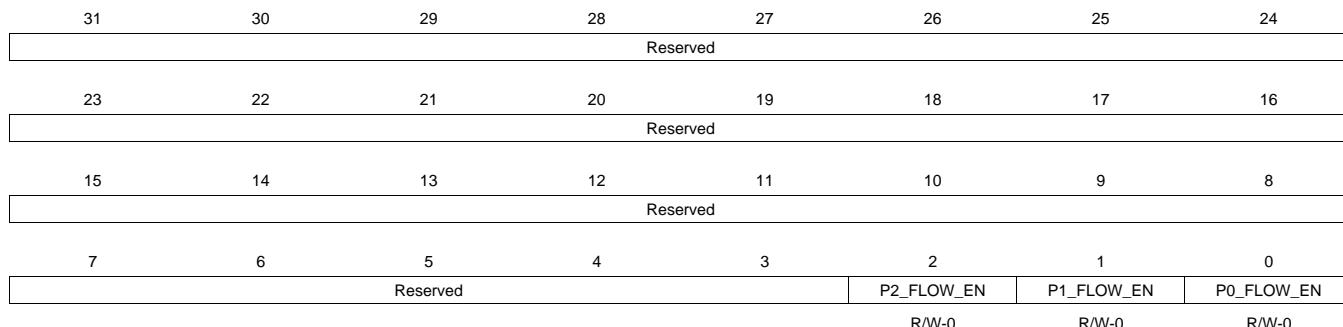
| Bit  | Field        | Type  | Reset | Description  |
|------|--------------|-------|-------|--|
| 10-0 | TX_START_WDS | R/W-0 | 0     | FIFO Packet Transmit (egress) Start Words.<br>This value is the number of required packet words in the transmit FIFO before the packet egress will begin.<br>This value is non-zero to preclude underrun.<br>Decimal 32 is the recommended value.<br>It should not be increased unnecessarily to prevent adding to the switch latency. |

#### 14.5.8.10 FLOW\_CONTROL Register (offset = 24h) [reset = 1h]

FLOW\_CONTROL is shown in [Figure 14-193](#) and described in [Table 14-210](#).

##### FLOW CONTROL

**Figure 14-193. FLOW\_CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-210. FLOW\_CONTROL Register Field Descriptions**

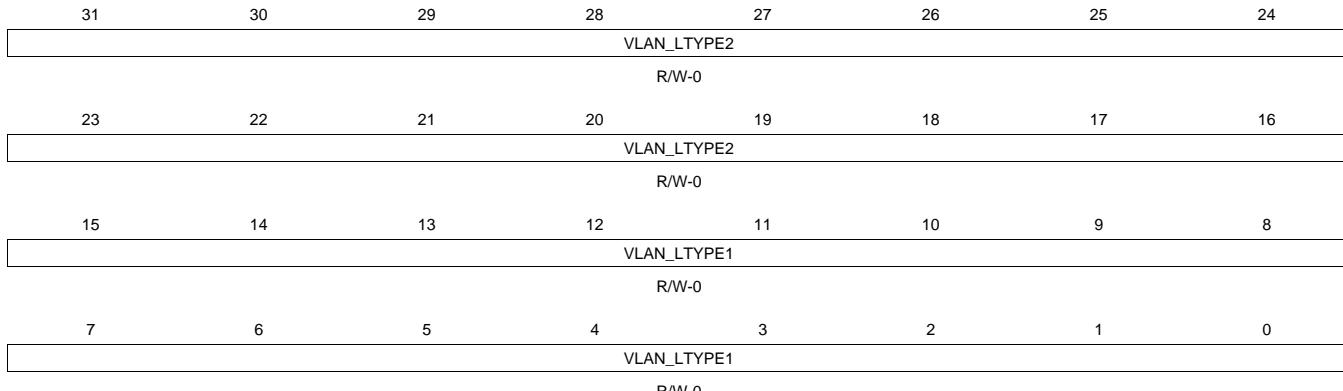
| Bit | Field      | Type  | Reset | Description                        |
|-----|------------|-------|-------|------------------------------------|
| 2   | P2_FLOW_EN | R/W-0 | 0     | Port 2 Receive flow control enable |
| 1   | P1_FLOW_EN | R/W-0 | 0     | Port 1 Receive flow control enable |
| 0   | P0_FLOW_EN | R/W-0 | 0     | Port 0 Receive flow control enable |

#### 14.5.8.11 VLAN\_LTYPE Register (offset = 28h) [reset = 81008100h]

VLAN\_LTYPE is shown in [Figure 14-194](#) and described in [Table 14-211](#).

##### LTYPE1 AND LTYPE 2 REGISTER

**Figure 14-194. VLAN\_LTYPE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-211. VLAN\_LTYPE Register Field Descriptions**

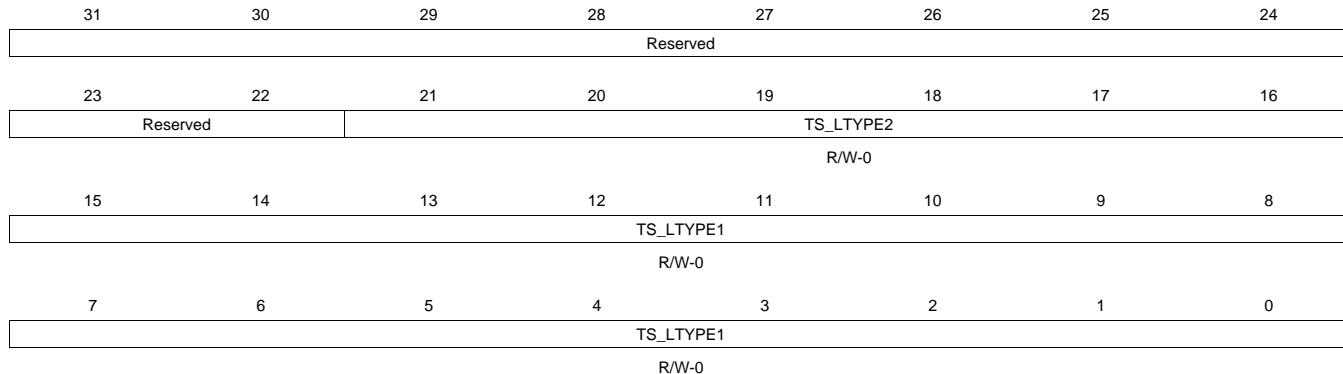
| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 31-16 | VLAN_LTYPE2 | R/W-0 | 0     | Time Sync VLAN LTYPE2 This VLAN LTYPE value is used for tx and rx.<br>This is the inner VLAN if both are present. |
| 15-0  | VLAN_LTYPE1 | R/W-0 | 0     | Time Sync VLAN LTYPE1 This VLAN LTYPE value is used for tx and rx.<br>This is the outer VLAN if both are present. |

#### 14.5.8.12 TS\_LTYPE Register (offset = 2Ch) [reset = 0h]

TS\_LTYPE is shown in [Figure 14-195](#) and described in [Table 14-212](#).

VLAN\_LTYPE1 AND VLAN\_LTYPE2 REGISTER

**Figure 14-195. TS\_LTYPE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-212. TS\_LTYPE Register Field Descriptions**

| Bit   | Field     | Type  | Reset | Description   |
|-------|-----------|-------|-------|---|
| 21-16 | TS_LTYPE2 | R/W-0 | 0     | Time Sync LTYPE2 This is an Ethertype value to match for tx and rx time sync packets. |
| 15-0  | TS_LTYPE1 | R/W-0 | 0     | Time Sync LTYPE1 This is an ethertype value to match for tx and rx time sync packets. |

#### 14.5.8.13 DLR\_LTYPE Register (offset = 30h) [reset = 80E1h]

DLR\_LTYPE is shown in [Figure 14-196](#) and described in [Table 14-213](#).

##### DLR LTYPE REGISTER

**Figure 14-196. DLR\_LTYPE Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | DLR_LTYPE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-213. DLR\_LTYPE Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description |
|------|-----------|-------|-------|-------------|
| 15-0 | DLR_LTYPE | R/W-0 | 0     | DLR LTYPE   |

#### 14.5.9 CPSW\_WR Registers

[Table 14-214](#) lists the memory-mapped registers for the CPSW\_WR. All register offset addresses not listed in [Table 14-214](#) should be considered as reserved locations and the register contents should not be modified.

**Table 14-214. CPSW\_WR REGISTERS**

| Offset | Acronym           | Register Name | Section                           |
|--------|-------------------|---------------|-----------------------------------|
| 0h     | IDVER             |               | <a href="#">Section 14.5.9.1</a>  |
| 4h     | SOFT_RESET        |               | <a href="#">Section 14.5.9.2</a>  |
| 8h     | CONTROL           |               | <a href="#">Section 14.5.9.3</a>  |
| Ch     | INT_CONTROL       |               | <a href="#">Section 14.5.9.4</a>  |
| 10h    | C0_RX_THRESH_EN   |               | <a href="#">Section 14.5.9.5</a>  |
| 14h    | C0_RX_EN          |               | <a href="#">Section 14.5.9.6</a>  |
| 18h    | C0_TX_EN          |               | <a href="#">Section 14.5.9.7</a>  |
| 1Ch    | C0_MISC_EN        |               | <a href="#">Section 14.5.9.8</a>  |
| 20h    | C1_RX_THRESH_EN   |               | <a href="#">Section 14.5.9.9</a>  |
| 24h    | C1_RX_EN          |               | <a href="#">Section 14.5.9.10</a> |
| 28h    | C1_TX_EN          |               | <a href="#">Section 14.5.9.11</a> |
| 2Ch    | C1_MISC_EN        |               | <a href="#">Section 14.5.9.12</a> |
| 30h    | C2_RX_THRESH_EN   |               | <a href="#">Section 14.5.9.13</a> |
| 34h    | C2_RX_EN          |               | <a href="#">Section 14.5.9.14</a> |
| 38h    | C2_TX_EN          |               | <a href="#">Section 14.5.9.15</a> |
| 3Ch    | C2_MISC_EN        |               | <a href="#">Section 14.5.9.16</a> |
| 40h    | C0_RX_THRESH_STAT |               | <a href="#">Section 14.5.9.17</a> |
| 44h    | C0_RX_STAT        |               | <a href="#">Section 14.5.9.18</a> |
| 48h    | C0_TX_STAT        |               | <a href="#">Section 14.5.9.19</a> |
| 4Ch    | C0_MISC_STAT      |               | <a href="#">Section 14.5.9.20</a> |
| 50h    | C1_RX_THRESH_STAT |               | <a href="#">Section 14.5.9.21</a> |
| 54h    | C1_RX_STAT        |               | <a href="#">Section 14.5.9.22</a> |
| 58h    | C1_TX_STAT        |               | <a href="#">Section 14.5.9.23</a> |
| 5Ch    | C1_MISC_STAT      |               | <a href="#">Section 14.5.9.24</a> |
| 60h    | C2_RX_THRESH_STAT |               | <a href="#">Section 14.5.9.25</a> |
| 64h    | C2_RX_STAT        |               | <a href="#">Section 14.5.9.26</a> |
| 68h    | C2_TX_STAT        |               | <a href="#">Section 14.5.9.27</a> |
| 6Ch    | C2_MISC_STAT      |               | <a href="#">Section 14.5.9.28</a> |

**Table 14-214. CPSW\_WR REGISTERS (continued)**

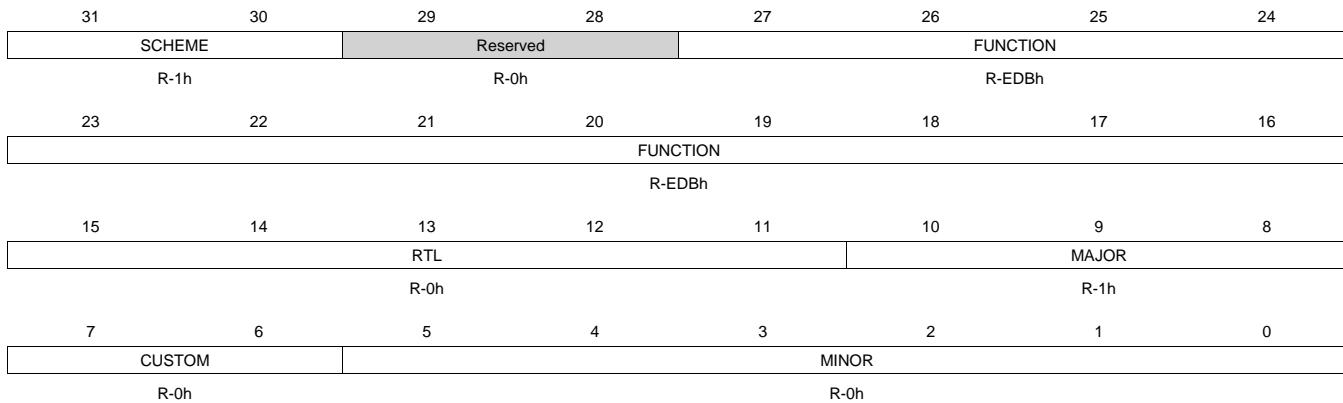
| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b> | <b>Section</b>                    |
|---------------|----------------|----------------------|-----------------------------------|
| 70h           | C0_RX_IMAX     |                      | <a href="#">Section 14.5.9.29</a> |
| 74h           | C0_TX_IMAX     |                      | <a href="#">Section 14.5.9.30</a> |
| 78h           | C1_RX_IMAX     |                      | <a href="#">Section 14.5.9.31</a> |
| 7Ch           | C1_TX_IMAX     |                      | <a href="#">Section 14.5.9.32</a> |
| 80h           | C2_RX_IMAX     |                      | <a href="#">Section 14.5.9.33</a> |
| 84h           | C2_TX_IMAX     |                      | <a href="#">Section 14.5.9.34</a> |
| 88h           | RGMII_CTL      |                      | <a href="#">Section 14.5.9.35</a> |

#### 14.5.9.1 IDVER Register (offset = 0h) [reset = 4EDB0100h]

IDVER is shown in [Figure 14-197](#) and described in [Table 14-215](#).

##### SUBSYSTEM ID VERSION REGISTER

**Figure 14-197. IDVER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-215. IDVER Register Field Descriptions**

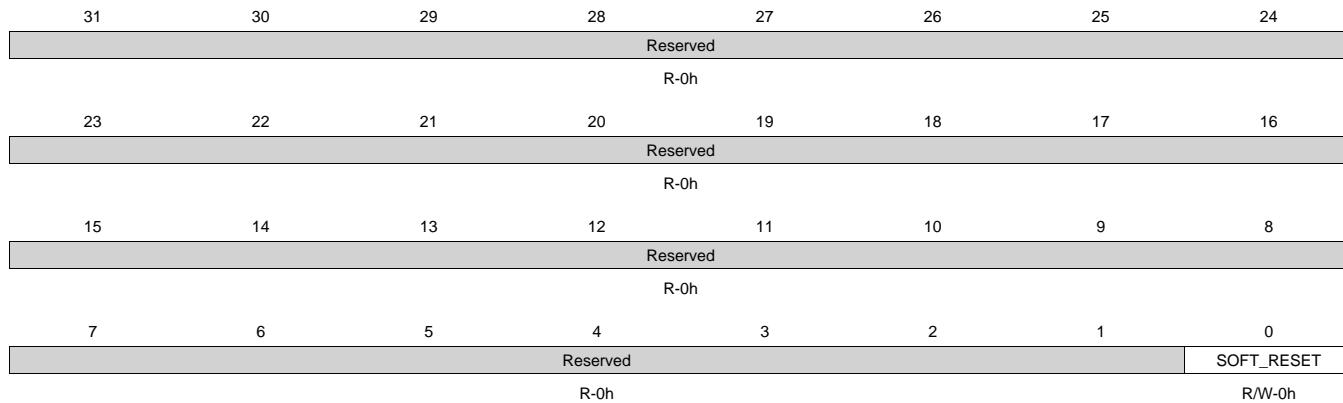
| Bit   | Field    | Type | Reset | Description    |
|-------|----------|------|-------|----------------|
| 31-30 | SCHEME   | R    | 1h    | Scheme value   |
| 29-28 | Reserved | R    | 0h    |                |
| 27-16 | FUNCTION | R    | EDBh  | function value |
| 15-11 | RTL      | R    | 0h    | rtl version    |
| 10-8  | MAJOR    | R    | 1h    | major version  |
| 7-6   | CUSTOM   | R    | 0h    | custom version |
| 5-0   | MINOR    | R    | 0h    | minor version  |

#### 14.5.9.2 SOFT\_RESET Register (offset = 4h) [reset = 0h]

SOFT\_RESET is shown in [Figure 14-198](#) and described in [Table 14-216](#).

##### SUBSYSTEM SOFT RESET REGISTER

**Figure 14-198. SOFT\_RESET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-216. SOFT\_RESET Register Field Descriptions**

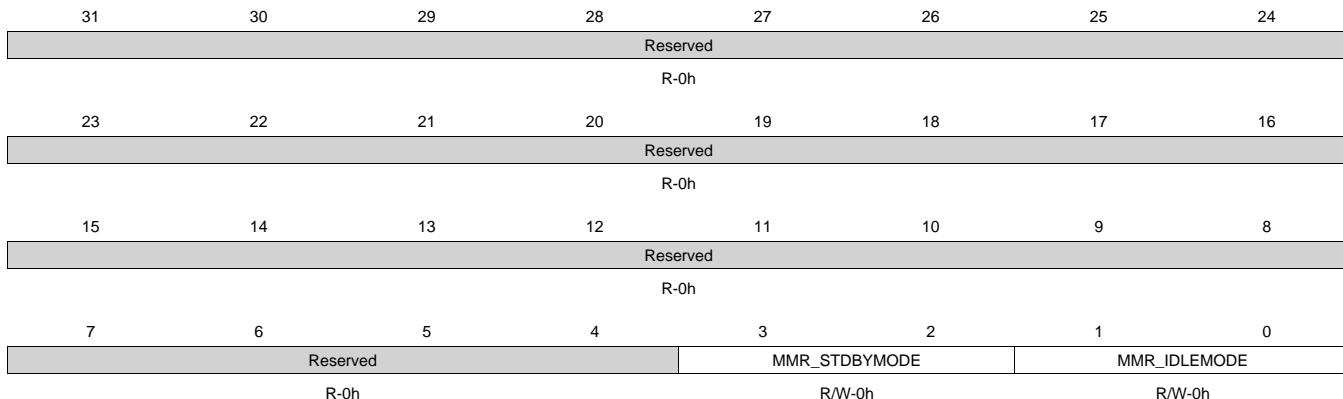
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-1 | Reserved   | R    | 0h    |   |
| 0    | SOFT_RESET | R/W  | 0h    | Software reset - Writing a one to this bit causes the CPGMACSS_R logic to be reset (INT, REGS, CPPI).<br>Software reset occurs on the clock following the register bit write. |

#### 14.5.9.3 CONTROL Register (offset = 8h) [reset = 0h]

CONTROL is shown in [Figure 14-199](#) and described in [Table 14-217](#).

##### SUBSYSTEM CONTROL REGISTER

**Figure 14-199. CONTROL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-217. CONTROL Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-4 | Reserved      | R    | 0h    |   |
| 3-2  | MMR_STDBYMODE | R/W  | 0h    | <p>Configuration of the local initiator state management mode. By definition, initiator may generate read/write transaction as long as it is out of STANDBY state.</p> <p>0x0 = Force-standby mode : Local initiator is unconditionally placed in standby state.</p> <p>0x1 = No-standby mode : Local initiator is unconditionally placed out of standby state.</p> <p>0x2 = Reserved : Reserved.</p> <p>0x3 = Reserved : Reserved.</p> |
| 1-0  | MMR_IDLEMODE  | R/W  | 0h    | <p>Configuration of the local initiator state management mode. By definition, initiator may generate read/write transaction as long as it is out of IDLE state.</p> <p>0x0 = Force-idle mode : Local initiator is unconditionally placed in idle state.</p> <p>0x1 = No-idle mode : Local initiator is unconditionally placed out of idle state.</p> <p>0x2 = Reserved : Reserved.</p> <p>0x3 = Reserved : Reserved.</p>                |

#### 14.5.9.4 INT\_CONTROL Register (offset = Ch) [reset = 0h]

INT\_CONTROL is shown in Figure 14-200 and described in Table 14-218.

##### SUBSYSTEM INTERRUPT CONTROL

**Figure 14-200. INT\_CONTROL Register**

|          |    |    |    |              |    |    |    |
|----------|----|----|----|--------------|----|----|----|
| 31       | 30 | 29 | 28 | 27           | 26 | 25 | 24 |
| INT_TEST |    |    |    | Reserved     |    |    |    |
| R/W-0h   |    |    |    | R-0h         |    |    |    |
| 23       | 22 | 21 | 20 | 19           | 18 | 17 | 16 |
| Reserved |    |    |    | INT_PACE_EN  |    |    |    |
| R-0h     |    |    |    | R/W-0h       |    |    |    |
| 15       | 14 | 13 | 12 | 11           | 10 | 9  | 8  |
| Reserved |    |    |    | INT_PRESCALE |    |    |    |
| R-0h     |    |    |    | R-0h         |    |    |    |
| 7        | 6  | 5  | 4  | 3            | 2  | 1  | 0  |
|          |    |    |    | INT_PRESCALE |    |    |    |
|          |    |    |    | R-0h         |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-218. INT\_CONTROL Register Field Descriptions**

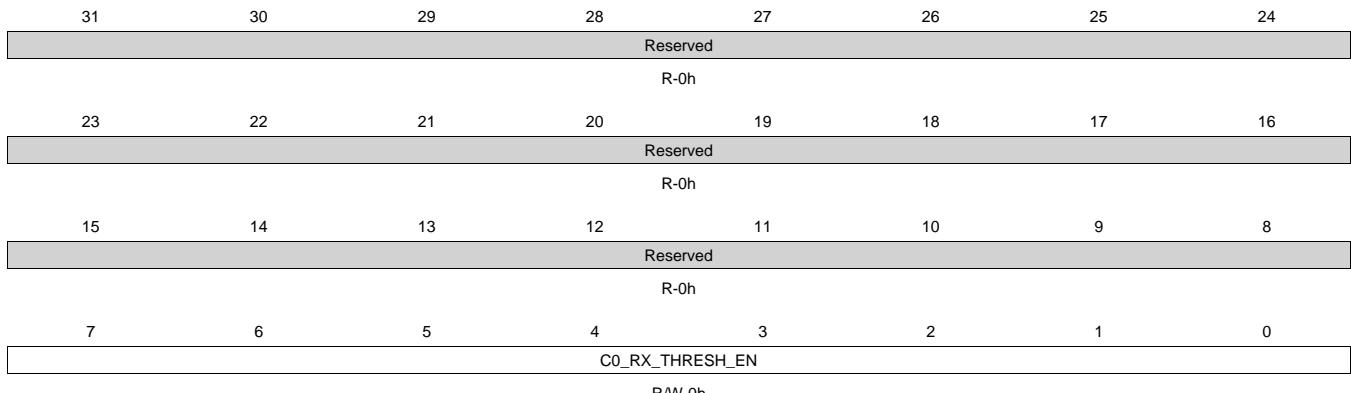
| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31    | INT_TEST     | R/W  | 0h    | Interrupt Test - Test bit to the interrupt pacing blocks  |
| 30-22 | Reserved     | R    | 0h    |   |
| 21-16 | INT_PACE_EN  | R/W  | 0h    | Interrupt Pacing Enable Bus int_pace_en[0] - Enables C0_Rx_Pulse Pacing (0 is pacing bypass) int_pace_en[1] - Enables C0_Tx_Pulse Pacing (0 is pacing bypass) int_pace_en[2] - Enables C1_Rx_Pulse Pacing (0 is pacing bypass) int_pace_en[3] - Enables C1_Tx_Pulse Pacing (0 is pacing bypass) int_pace_en[4] - Enables C2_Rx_Pulse Pacing (0 is pacing bypass) int_pace_en[5] - Enables C2_Tx_Pulse Pacing (0 is pacing bypass) |
| 15-12 | Reserved     | R    | 0h    |   |
| 11-0  | INT_PRESCALE | R    | 0h    | Interrupt Counter Prescaler - The number of MAIN_CLK periods in 4us.  |

#### 14.5.9.5 C0\_RX\_THRESH\_EN Register (offset = 10h) [reset = 0h]

C0\_RX\_THRESH\_EN is shown in [Figure 14-201](#) and described in [Table 14-219](#).

SUBSYSTEM CORE 0 RECEIVE THRESHOLD INT ENABLE REGISTER

**Figure 14-201. C0\_RX\_THRESH\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-219. C0\_RX\_THRESH\_EN Register Field Descriptions**

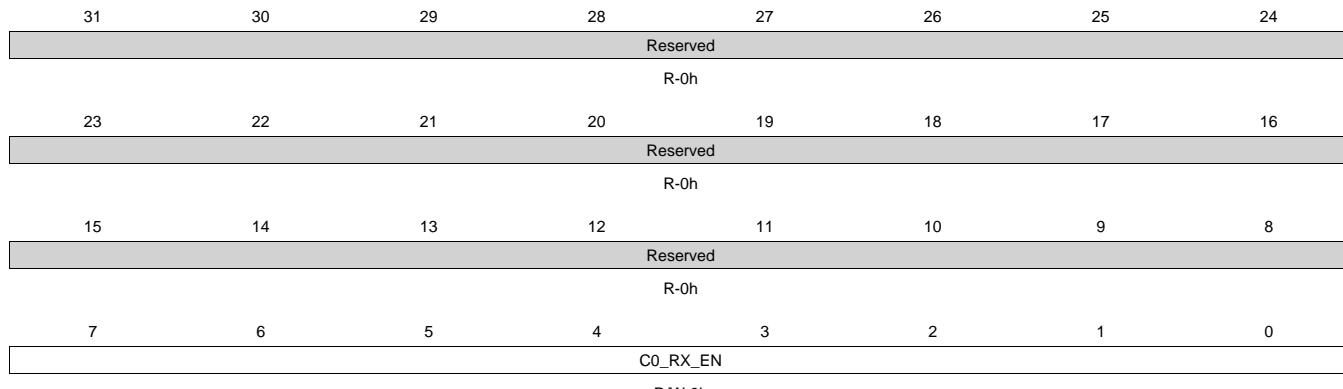
| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-8 | Reserved        | R    | 0h    |   |
| 7-0  | C0_RX_THRESH_EN | R/W  | 0h    | Core 0 Receive Threshold Enable - Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled to generate an interrupt on C0_RX_THRESH_PULSE. |

#### 14.5.9.6 C0\_RX\_EN Register (offset = 14h) [reset = 0h]

C0\_RX\_EN is shown in [Figure 14-202](#) and described in [Table 14-220](#).

SUBSYSTEM CORE 0 RECEIVE INTERRUPT ENABLE REGISTER

**Figure 14-202. C0\_RX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-220. C0\_RX\_EN Register Field Descriptions**

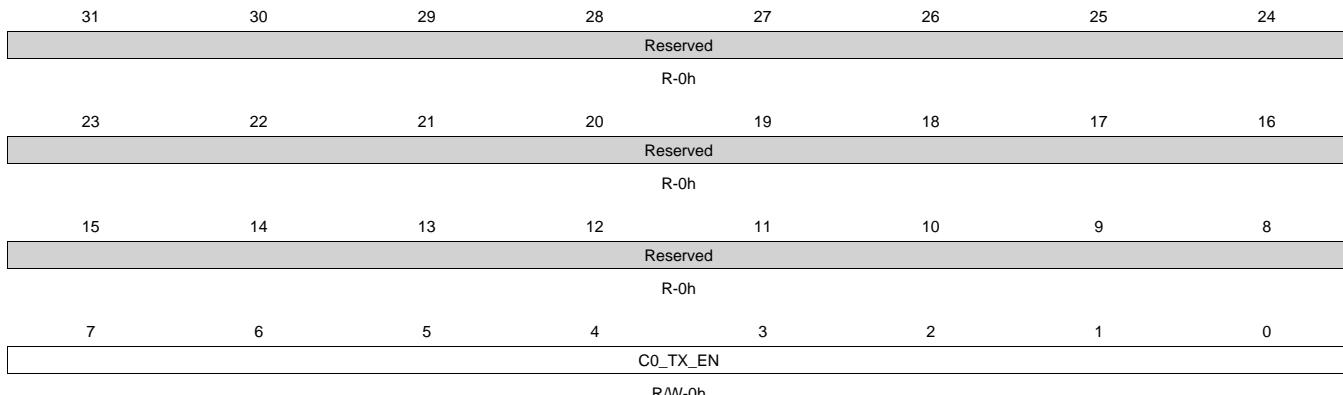
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | C0_RX_EN | R/W  | 0h    | Core 0 Receive Enable - Each bit in this register corresponds to the bit in the rx interrupt that is enabled to generate an interrupt on C0_RX_PULSE. |

#### 14.5.9.7 C0\_TX\_EN Register (offset = 18h) [reset = 0h]

C0\_TX\_EN is shown in [Figure 14-203](#) and described in [Table 14-221](#).

##### SUBSYSTEM CORE 0 TRANSMIT INTERRUPT ENABLE REGISTER

**Figure 14-203. C0\_TX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-221. C0\_TX\_EN Register Field Descriptions**

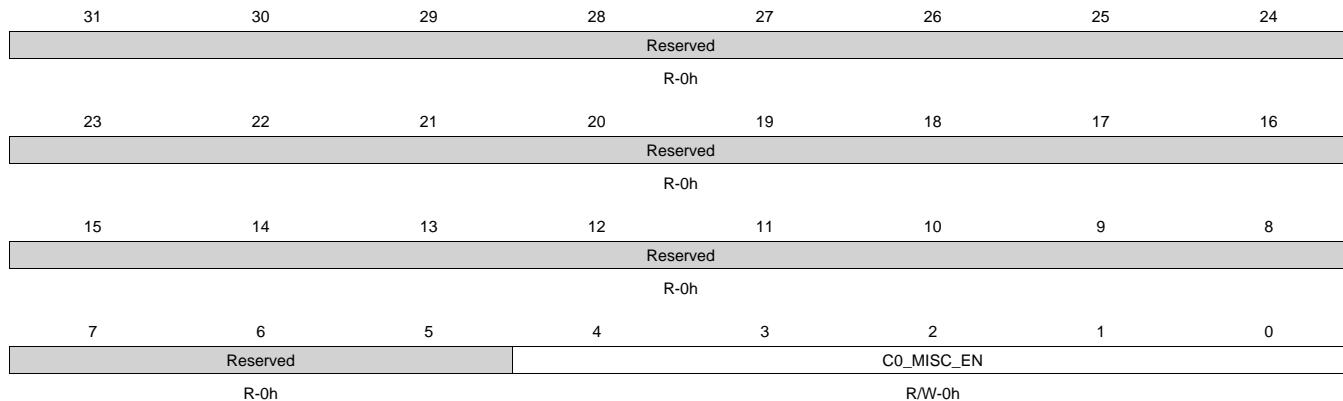
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7-0  | C0_TX_EN | R/W  | 0h    | Core 0 Transmit Enable - Each bit in this register corresponds to the bit in the tx interrupt that is enabled to generate an interrupt on C0_TX_PULSE. |

#### 14.5.9.8 C0\_MISC\_EN Register (offset = 1Ch) [reset = 0h]

C0\_MISC\_EN is shown in [Figure 14-204](#) and described in [Table 14-222](#).

##### SUBSYSTEM CORE 0 MISC INTERRUPT ENABLE REGISTER

**Figure 14-204. C0\_MISC\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-222. C0\_MISC\_EN Register Field Descriptions**

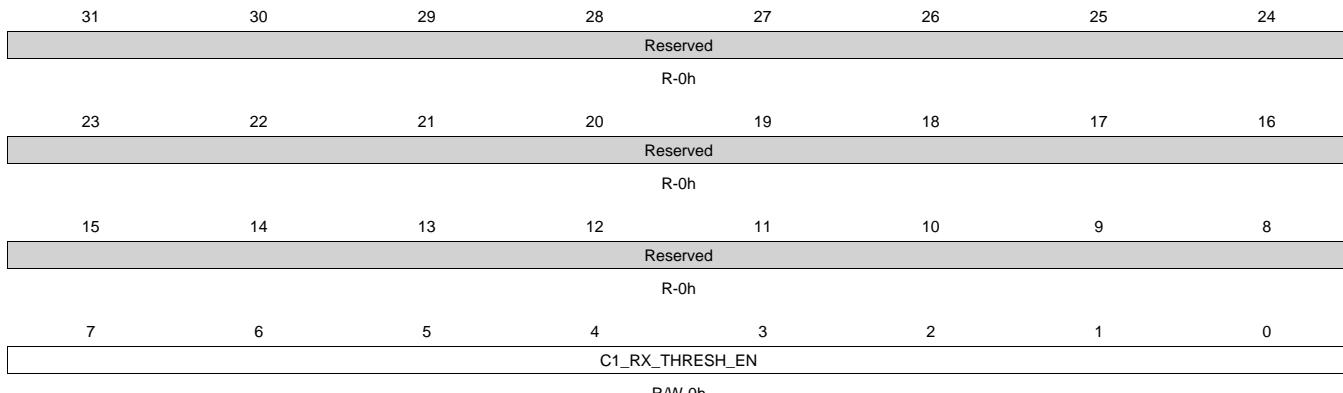
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-5 | Reserved   | R    | 0h    |  |
| 4-0  | C0_MISC_EN | R/W  | 0h    | Core 0 Misc Enable - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled to generate an interrupt on C0_Misc_PULSE. |

#### 14.5.9.9 C1\_RX\_THRESH\_EN Register (offset = 20h) [reset = 0h]

C1\_RX\_THRESH\_EN is shown in [Figure 14-205](#) and described in [Table 14-223](#).

SUBSYSTEM CORE 1 RECEIVE THRESHOLD INT ENABLE REGISTER

**Figure 14-205. C1\_RX\_THRESH\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-223. C1\_RX\_THRESH\_EN Register Field Descriptions**

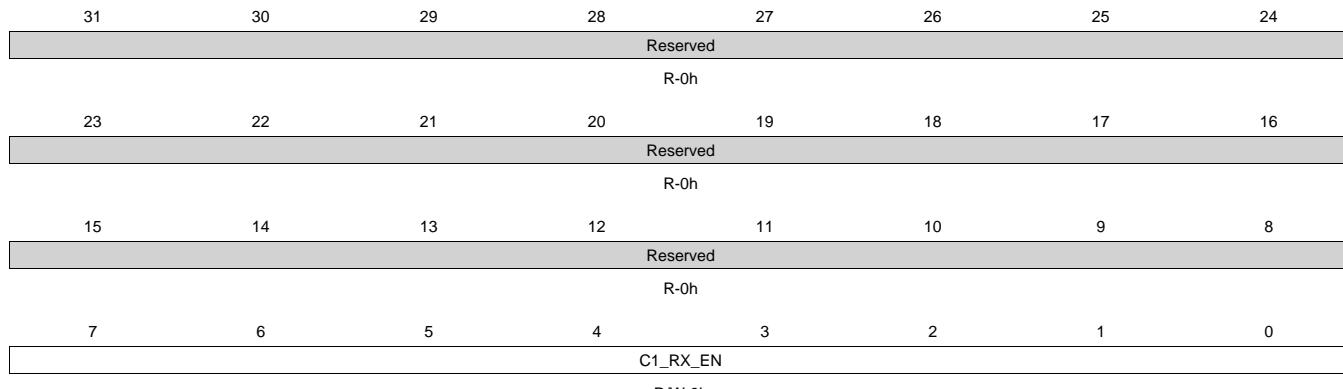
| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-8 | Reserved        | R    | 0h    |   |
| 7-0  | C1_RX_THRESH_EN | R/W  | 0h    | Core 1 Receive Threshold Enable - Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled to generate an interrupt on C1_RX_THRESH_PULSE. |

#### 14.5.9.10 C1\_RX\_EN Register (offset = 24h) [reset = 0h]

C1\_RX\_EN is shown in [Figure 14-206](#) and described in [Table 14-224](#).

SUBSYSTEM CORE 1 RECEIVE INTERRUPT ENABLE REGISTER

**Figure 14-206. C1\_RX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-224. C1\_RX\_EN Register Field Descriptions**

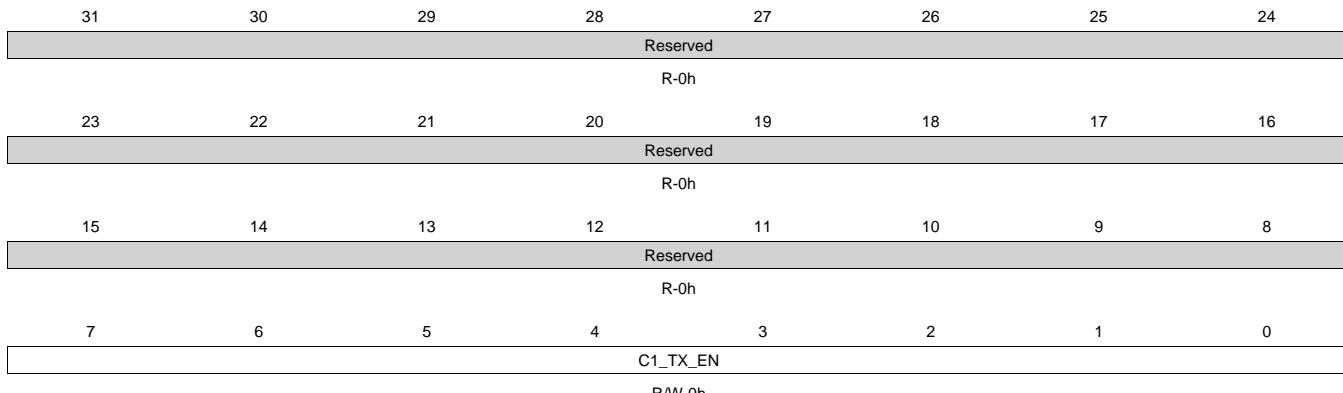
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | C1_RX_EN | R/W  | 0h    | Core 1 Receive Enable - Each bit in this register corresponds to the bit in the rx interrupt that is enabled to generate an interrupt on C1_RX_PULSE. |

#### 14.5.9.11 C1\_TX\_EN Register (offset = 28h) [reset = 0h]

C1\_TX\_EN is shown in [Figure 14-207](#) and described in [Table 14-225](#).

##### SUBSYSTEM CORE 1 TRANSMIT INTERRUPT ENABLE REGISTER

**Figure 14-207. C1\_TX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-225. C1\_TX\_EN Register Field Descriptions**

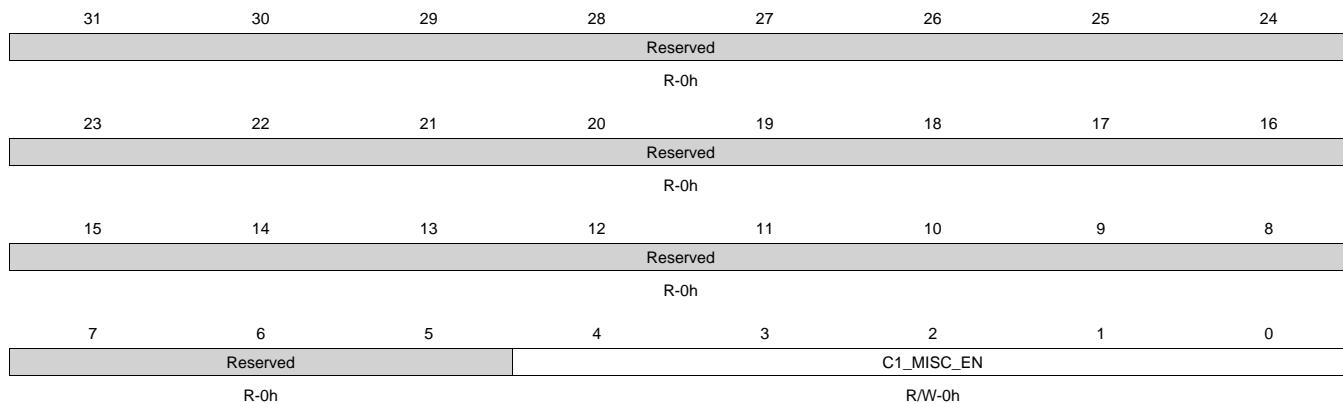
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7-0  | C1_TX_EN | R/W  | 0h    | Core 1 Transmit Enable - Each bit in this register corresponds to the bit in the tx interrupt that is enabled to generate an interrupt on C1_TX_PULSE. |

#### 14.5.9.12 C1\_MISC\_EN Register (offset = 2Ch) [reset = 0h]

C1\_MISC\_EN is shown in [Figure 14-208](#) and described in [Table 14-226](#).

##### SUBSYSTEM CORE 1 MISC INTERRUPT ENABLE REGISTER

**Figure 14-208. C1\_MISC\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-226. C1\_MISC\_EN Register Field Descriptions**

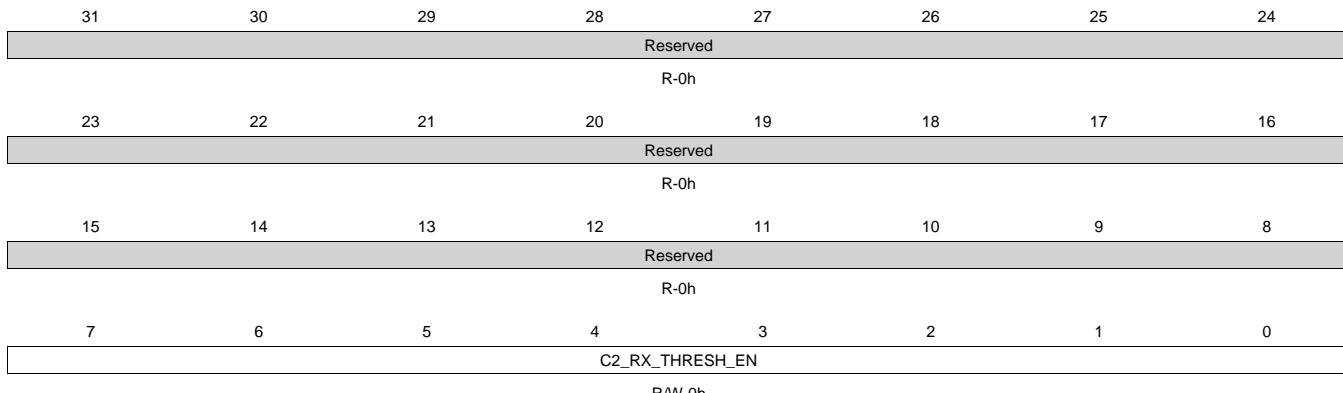
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-5 | Reserved   | R    | 0h    |  |
| 4-0  | C1_MISC_EN | R/W  | 0h    | Core 1 Misc Enable - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled to generate an interrupt on C1_Misc_PULSE. |

#### 14.5.9.13 C2\_RX\_THRESH\_EN Register (offset = 30h) [reset = 0h]

C2\_RX\_THRESH\_EN is shown in [Figure 14-209](#) and described in [Table 14-227](#).

SUBSYSTEM CORE 2 RECEIVE THRESHOLD INT ENABLE REGISTER

**Figure 14-209. C2\_RX\_THRESH\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-227. C2\_RX\_THRESH\_EN Register Field Descriptions**

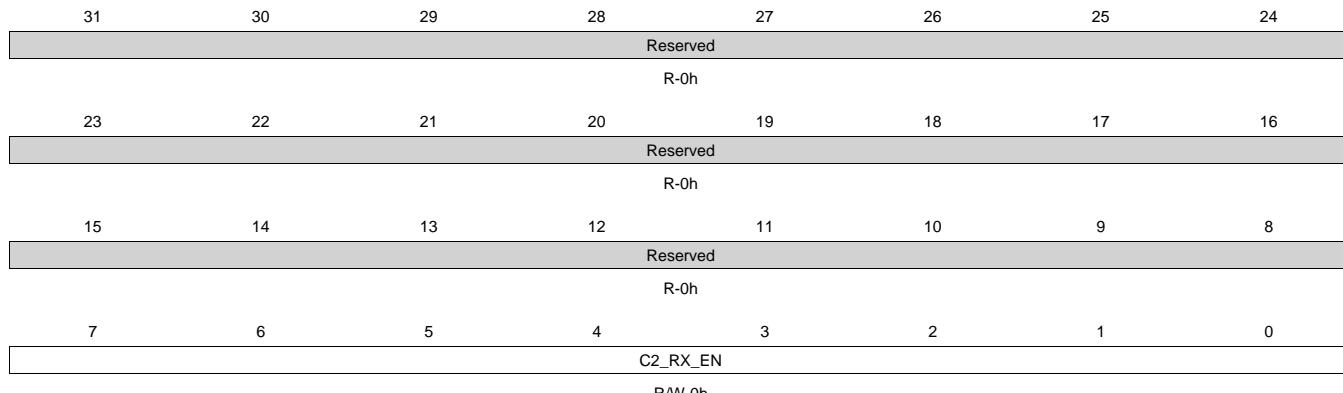
| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-8 | Reserved        | R    | 0h    |   |
| 7-0  | C2_RX_THRESH_EN | R/W  | 0h    | Core 2 Receive Threshold Enable - Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled to generate an interrupt on C2_RX_THRESH_PULSE. |

#### 14.5.9.14 C2\_RX\_EN Register (offset = 34h) [reset = 0h]

C2\_RX\_EN is shown in [Figure 14-210](#) and described in [Table 14-228](#).

##### SUBSYSTEM CORE 2 RECEIVE INTERRUPT ENABLE REGISTER

**Figure 14-210. C2\_RX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-228. C2\_RX\_EN Register Field Descriptions**

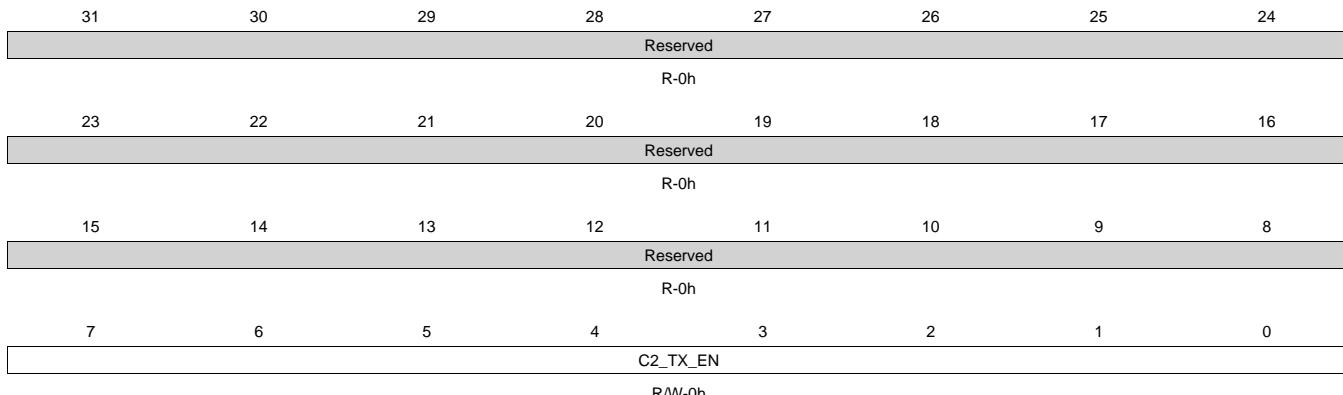
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | C2_RX_EN | R/W  | 0h    | Core 2 Receive Enable - Each bit in this register corresponds to the bit in the rx interrupt that is enabled to generate an interrupt on C2_RX_PULSE. |

#### 14.5.9.15 C2\_TX\_EN Register (offset = 38h) [reset = 0h]

C2\_TX\_EN is shown in [Figure 14-211](#) and described in [Table 14-229](#).

##### SUBSYSTEM CORE 2 TRANSMIT INTERRUPT ENABLE REGISTER

**Figure 14-211. C2\_TX\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-229. C2\_TX\_EN Register Field Descriptions**

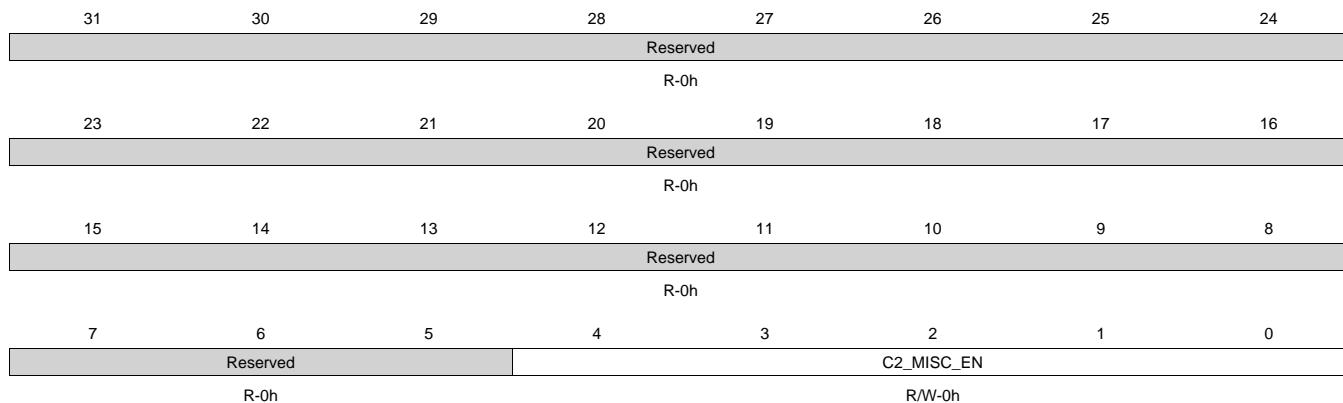
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7-0  | C2_TX_EN | R/W  | 0h    | Core 2 Transmit Enable - Each bit in this register corresponds to the bit in the tx interrupt that is enabled to generate an interrupt on C2_TX_PULSE. |

#### 14.5.9.16 C2\_MISC\_EN Register (offset = 3Ch) [reset = 0h]

C2\_MISC\_EN is shown in [Figure 14-212](#) and described in [Table 14-230](#).

##### SUBSYSTEM CORE 2 MISC INTERRUPT ENABLE REGISTER

**Figure 14-212. C2\_MISC\_EN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-230. C2\_MISC\_EN Register Field Descriptions**

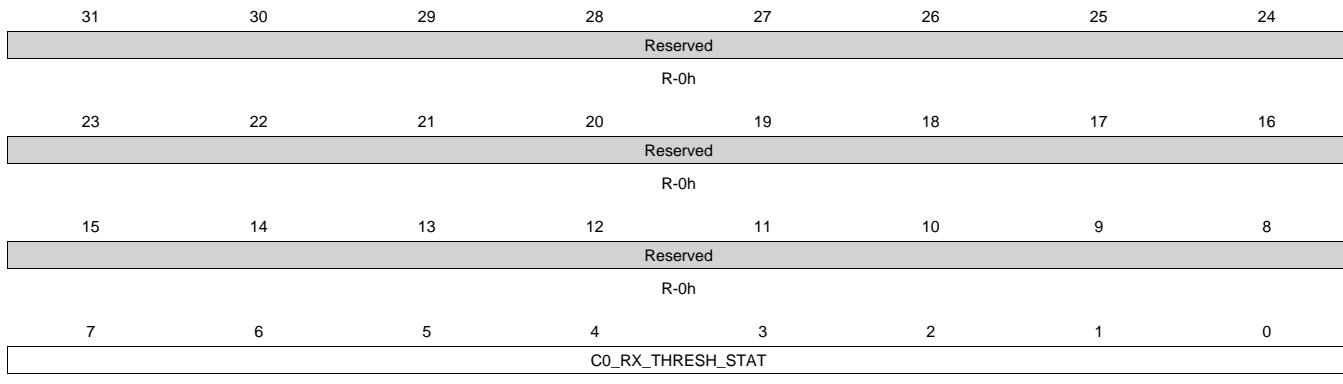
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-5 | Reserved   | R    | 0h    |  |
| 4-0  | C2_MISC_EN | R/W  | 0h    | Core 2 Misc Enable - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled to generate an interrupt on C2_Misc_PULSE. |

#### 14.5.9.17 C0\_RX\_THRESH\_STAT Register (offset = 40h) [reset = 0h]

C0\_RX\_THRESH\_STAT is shown in [Figure 14-213](#) and described in [Table 14-231](#).

##### SUBSYSTEM CORE 0 RX THRESHOLD MASKED INT STATUS REGISTER

**Figure 14-213. C0\_RX\_THRESH\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-231. C0\_RX\_THRESH\_STAT Register Field Descriptions**

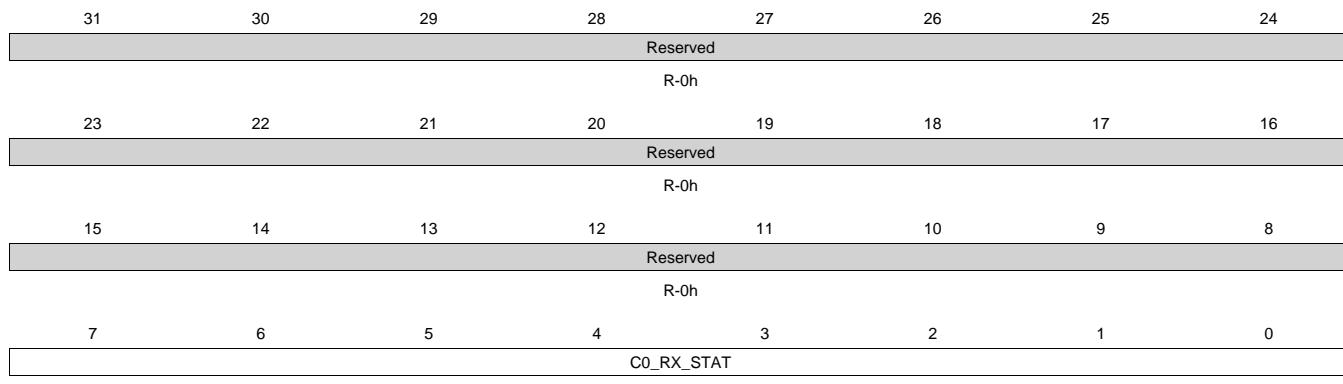
| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-8 | Reserved          | R    | 0h    |   |
| 7-0  | C0_RX_THRESH_STAT | R    | 0h    | Core 0 Receive Threshold Masked Interrupt Status - Each bit in this read only register corresponds to the bit in the receive threshold interrupt that is enabled and generating an interrupt on C0_RX_THRESH_PULSE. |

#### 14.5.9.18 C0\_RX\_STAT Register (offset = 44h) [reset = 0h]

C0\_RX\_STAT is shown in [Figure 14-214](#) and described in [Table 14-232](#).

SUBSYSTEM CORE 0 RX INTERRUPT MASKED INT STATUS REGISTER

**Figure 14-214. C0\_RX\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-232. C0\_RX\_STAT Register Field Descriptions**

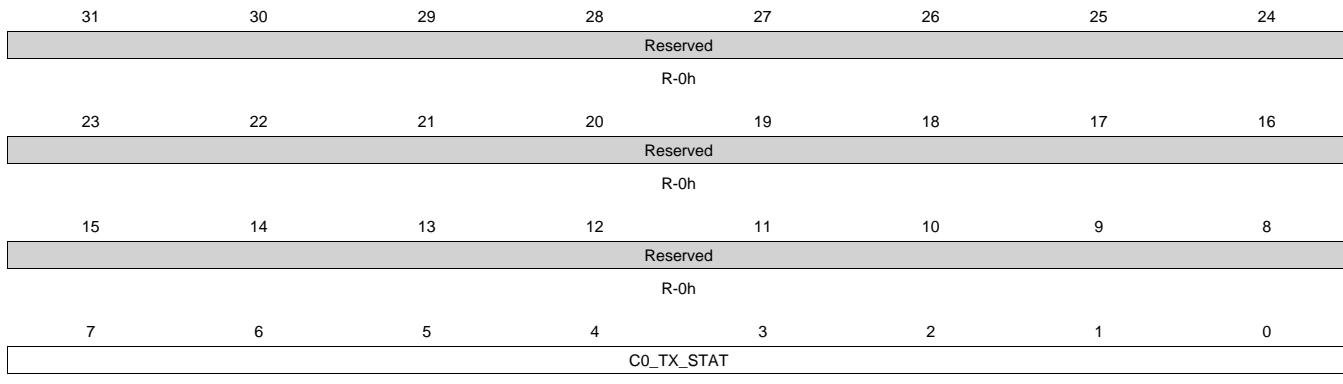
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-8 | Reserved   | R    | 0h    |   |
| 7-0  | C0_RX_STAT | R    | 0h    | Core 0 Receive Masked Interrupt Status - Each bit in this read only register corresponds to the bit in the Rx interrupt that is enabled and generating an interrupt on C0_RX_PULSE. |

#### 14.5.9.19 C0\_TX\_STAT Register (offset = 48h) [reset = 0h]

C0\_TX\_STAT is shown in [Figure 14-215](#) and described in [Table 14-233](#).

SUBSYSTEM CORE 0 TX INTERRUPT MASKED INT STATUS REGISTER

**Figure 14-215. C0\_TX\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-233. C0\_TX\_STAT Register Field Descriptions**

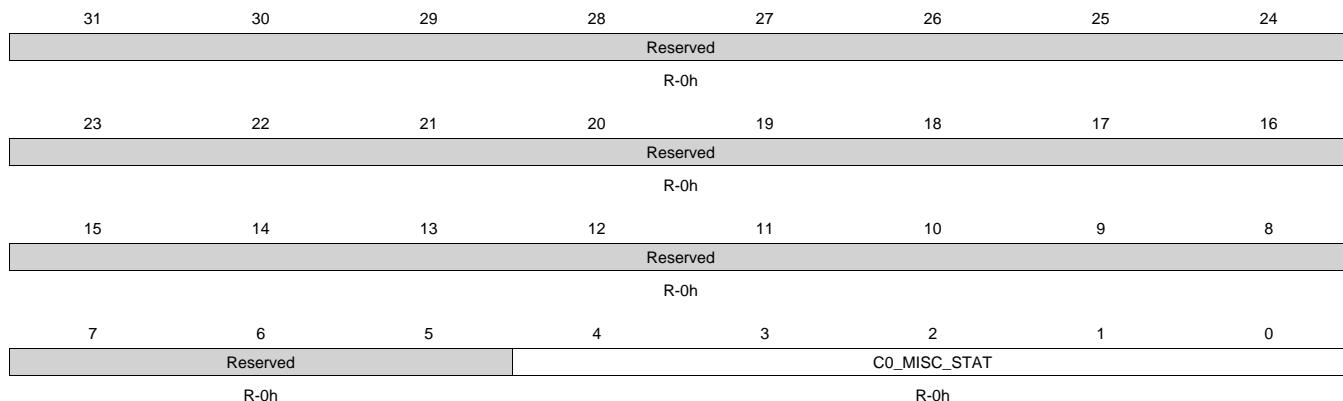
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-8 | Reserved   | R    | 0h    |   |
| 7-0  | C0_TX_STAT | R    | 0h    | Core 0 Transmit Masked Interrupt Status - Each bit in this read only register corresponds to the bit in the Tx interrupt that is enabled and generating an interrupt on C0_TX_PULSE . |

#### 14.5.9.20 C0\_MISC\_STAT Register (offset = 4Ch) [reset = 0h]

C0\_MISC\_STAT is shown in [Figure 14-216](#) and described in [Table 14-234](#).

SUBSYSTEM CORE 0 MISC INTERRUPT MASKED INT STATUS REGISTER

**Figure 14-216. C0\_MISC\_STAT Register**



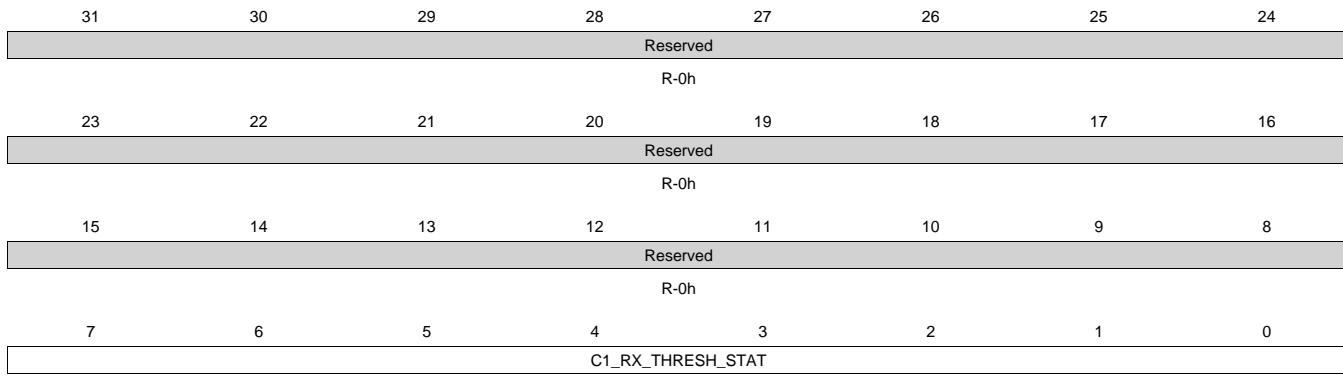
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-234. C0\_MISC\_STAT Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-5 | Reserved     | R    | 0h    |   |
| 4-0  | C0_MISC_STAT | R    | 0h    | Core 0 Misc Masked Interrupt Status - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled and generating an interrupt on C0_MISC_PULSE . |

**14.5.9.21 C1\_RX\_THRESH\_STAT Register (offset = 50h) [reset = 0h]**

C1\_RX\_THRESH\_STAT is shown in [Figure 14-217](#) and described in [Table 14-235](#).

**SUBSYSTEM CORE 1 RX THRESHOLD MASKED INT STATUS REGISTER**
**Figure 14-217. C1\_RX\_THRESH\_STAT Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-235. C1\_RX\_THRESH\_STAT Register Field Descriptions**

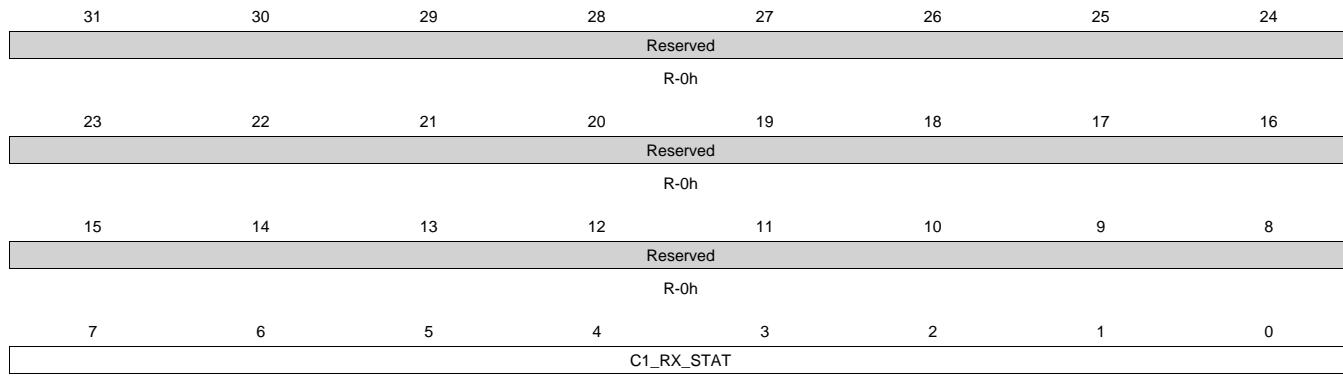
| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-8 | Reserved          | R    | 0h    |   |
| 7-0  | C1_RX_THRESH_STAT | R    | 0h    | Core 1 Receive Threshold Masked Interrupt Status - Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled and generating an interrupt on C1_RX_THRESH_PULSE. |

#### 14.5.9.22 C1\_RX\_STAT Register (offset = 54h) [reset = 0h]

C1\_RX\_STAT is shown in [Figure 14-218](#) and described in [Table 14-236](#).

SUBSYSTEM CORE 1 RECEIVE MASKED INTERRUPT STATUS REGISTER

**Figure 14-218. C1\_RX\_STAT Register**



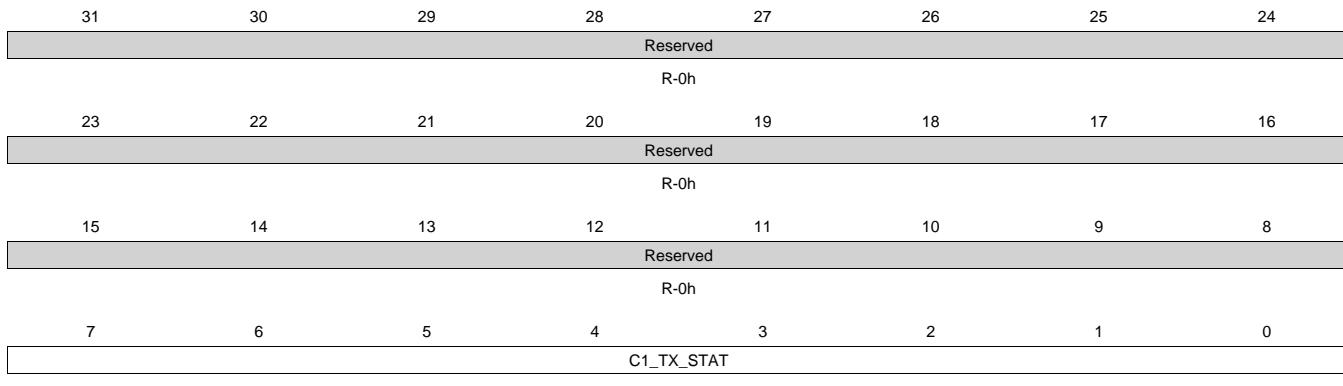
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-236. C1\_RX\_STAT Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-8 | Reserved   | R    | 0h    |   |
| 7-0  | C1_RX_STAT | R    | 0h    | Core 1 Receive Masked Interrupt Status - Each bit in this register corresponds to the bit in the Rx interrupt that is enabled and generating an interrupt on C1_RX_PULSE. |

**14.5.9.23 C1\_TX\_STAT Register (offset = 58h) [reset = 0h]**

C1\_TX\_STAT is shown in [Figure 14-219](#) and described in [Table 14-237](#).

**SUBSYSTEM CORE 1 TRANSMIT MASKED INTERRUPT STATUS REGISTER**
**Figure 14-219. C1\_TX\_STAT Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-237. C1\_TX\_STAT Register Field Descriptions**

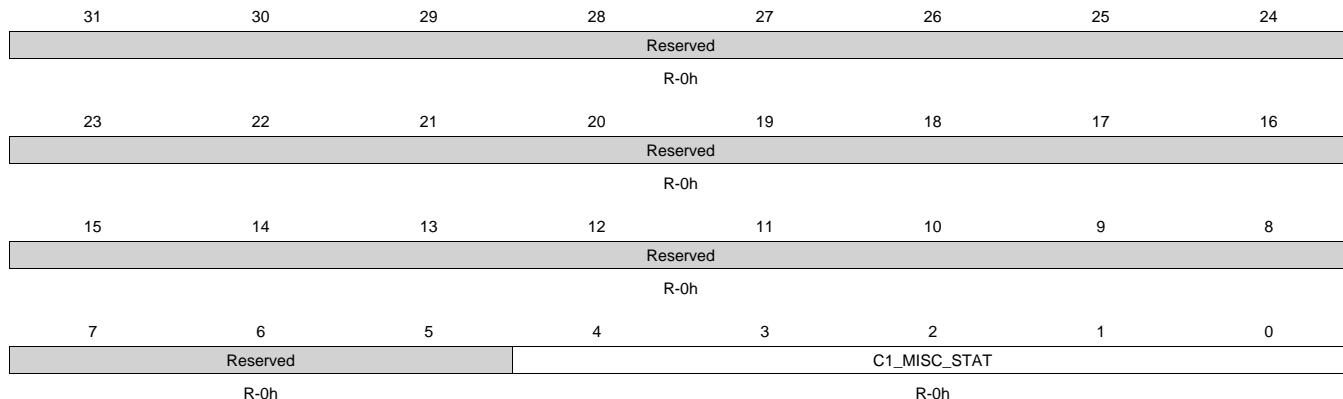
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-8 | Reserved   | R    | 0h    |  |
| 7-0  | C1_TX_STAT | R    | 0h    | Core 1 Transmit Masked Interrupt Status - Each bit in this register corresponds to the bit in the Tx interrupt that is enabled and generating an interrupt on C1_TX_PULSE. |

#### 14.5.9.24 C1\_MISC\_STAT Register (offset = 5Ch) [reset = 0h]

C1\_MISC\_STAT is shown in [Figure 14-220](#) and described in [Table 14-238](#).

##### SUBSYSTEM CORE 1 MISC MASKED INTERRUPT STATUS REGISTER

**Figure 14-220. C1\_MISC\_STAT Register**



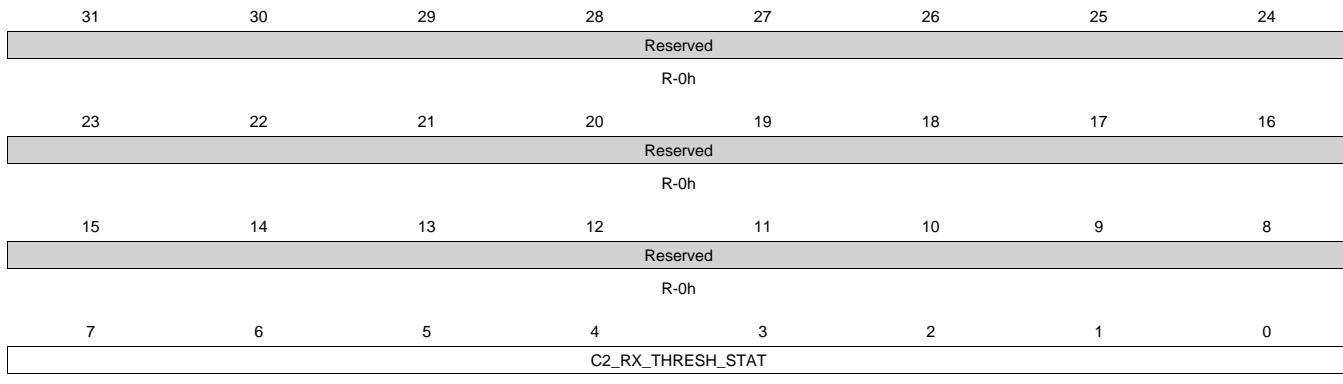
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-238. C1\_MISC\_STAT Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-5 | Reserved     | R    | 0h    |   |
| 4-0  | C1_MISC_STAT | R    | 0h    | Core 1 Misc Masked Interrupt Status - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled and generating an interrupt on C1_MISC_PULSE . |

**14.5.9.25 C2\_RX\_THRESH\_STAT Register (offset = 60h) [reset = 0h]**

C2\_RX\_THRESH\_STAT is shown in [Figure 14-221](#) and described in [Table 14-239](#).

**SUBSYSTEM CORE 2 RX THRESHOLD MASKED INT STATUS REGISTER**
**Figure 14-221. C2\_RX\_THRESH\_STAT Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-239. C2\_RX\_THRESH\_STAT Register Field Descriptions**

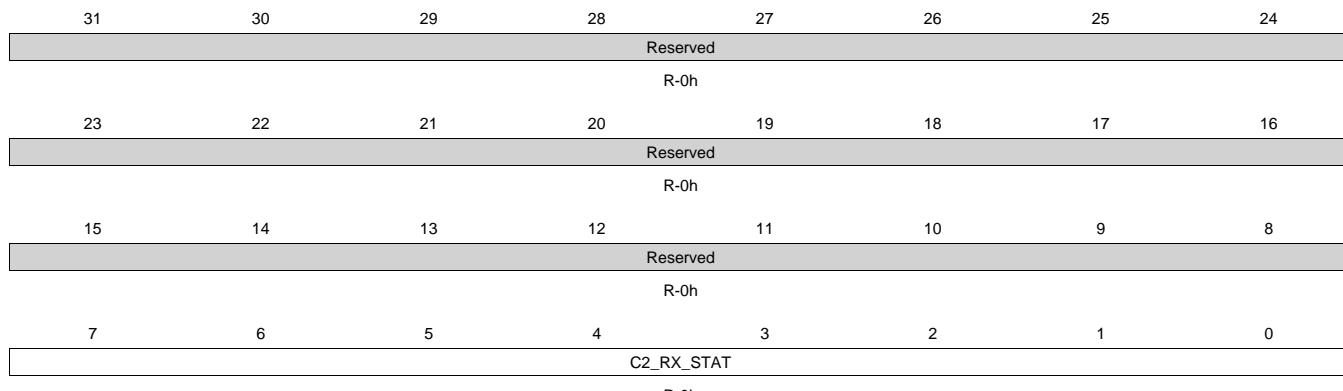
| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 31-8 | Reserved          | R    | 0h    |   |
| 7-0  | C2_RX_THRESH_STAT | R    | 0h    | Core 2 Receive Threshold Masked Interrupt Status - Each bit in this register corresponds to the bit in the receive threshold interrupt that is enabled and generating an interrupt on C2_RX_THRESH_PULSE. |

#### 14.5.9.26 C2\_RX\_STAT Register (offset = 64h) [reset = 0h]

C2\_RX\_STAT is shown in [Figure 14-222](#) and described in [Table 14-240](#).

SUBSYSTEM CORE 2 RECEIVE MASKED INTERRUPT STATUS REGISTER

**Figure 14-222. C2\_RX\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-240. C2\_RX\_STAT Register Field Descriptions**

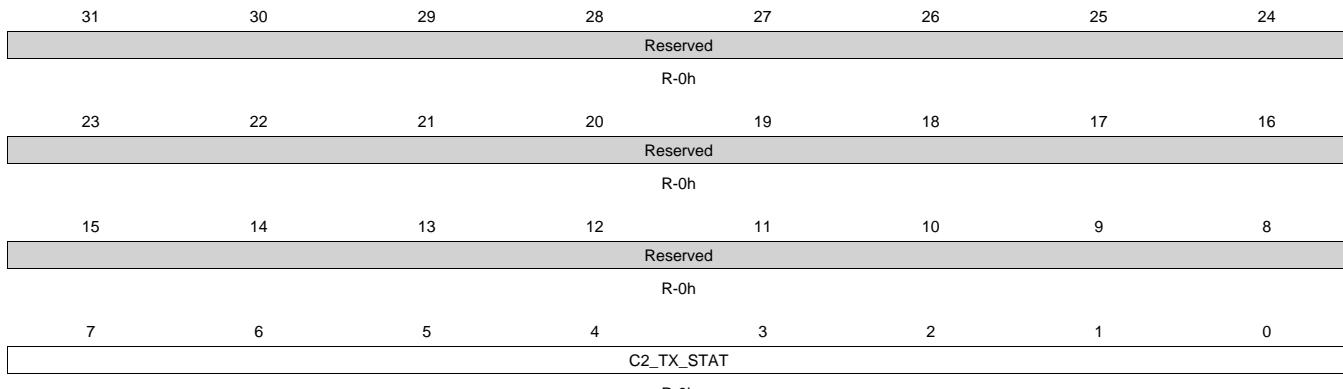
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-8 | Reserved   | R    | 0h    |   |
| 7-0  | C2_RX_STAT | R    | 0h    | Core 2 Receive Masked Interrupt Status - Each bit in this register corresponds to the bit in the Rx interrupt that is enabled and generating an interrupt on C2_RX_PULSE. |

#### 14.5.9.27 C2\_TX\_STAT Register (offset = 68h) [reset = 0h]

C2\_TX\_STAT is shown in [Figure 14-223](#) and described in [Table 14-241](#).

##### SUBSYSTEM CORE 2 TRANSMIT MASKED INTERRUPT STATUS REGISTER

**Figure 14-223. C2\_TX\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-241. C2\_TX\_STAT Register Field Descriptions**

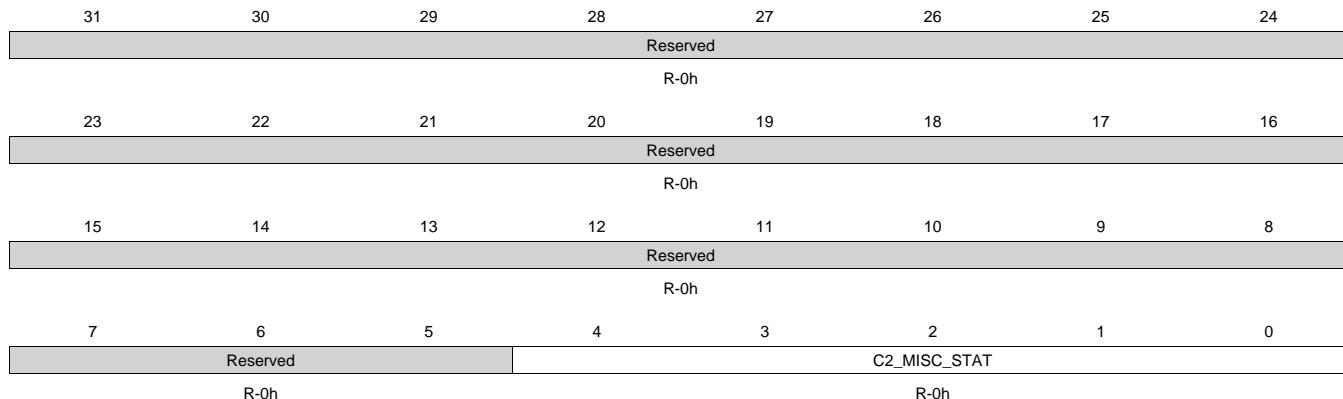
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-8 | Reserved   | R    | 0h    |  |
| 7-0  | C2_TX_STAT | R    | 0h    | Core 2 Transmit Masked Interrupt Status - Each bit in this register corresponds to the bit in the Tx interrupt that is enabled and generating an interrupt on C2_TX_PULSE. |

#### 14.5.9.28 C2\_MISC\_STAT Register (offset = 6Ch) [reset = 0h]

C2\_MISC\_STAT is shown in [Figure 14-224](#) and described in [Table 14-242](#).

##### SUBSYSTEM CORE 2 MISC MASKED INTERRUPT STATUS REGISTER

**Figure 14-224. C2\_MISC\_STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-242. C2\_MISC\_STAT Register Field Descriptions**

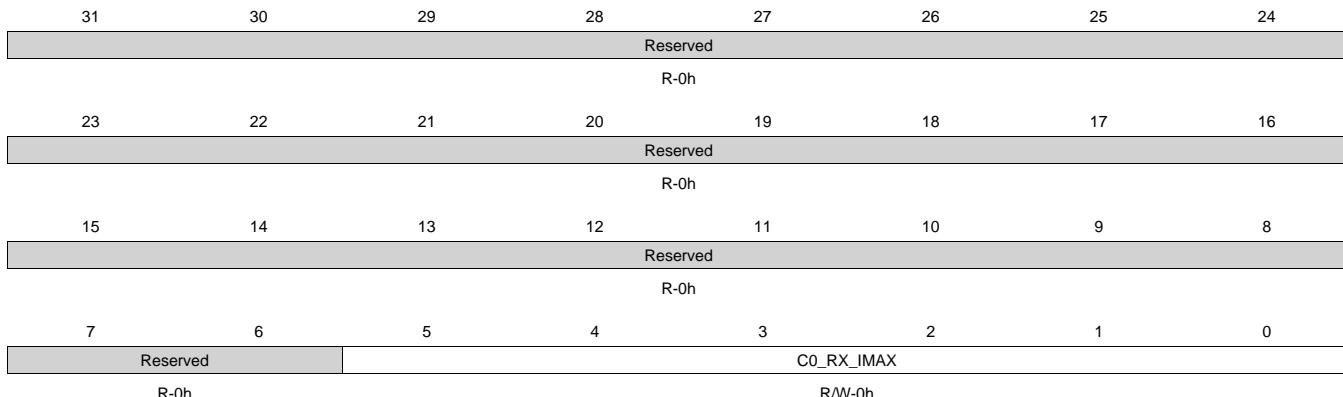
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-5 | Reserved     | R    | 0h    |   |
| 4-0  | C2_MISC_STAT | R    | 0h    | Core 2 Misc Masked Interrupt Status - Each bit in this register corresponds to the miscellaneous interrupt (evnt_pend, stat_pend, host_pend, mdio_linkint, mdio_userint) that is enabled and generating an interrupt on C2_MISC_PULSE . |

#### 14.5.9.29 C0\_RX\_IMAX Register (offset = 70h) [reset = 0h]

C0\_RX\_IMAX is shown in [Figure 14-225](#) and described in [Table 14-243](#).

SUBSYSTEM CORE 0 RECEIVE INTERRUPTS PER MILLISECOND

**Figure 14-225. C0\_RX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-243. C0\_RX\_IMAX Register Field Descriptions**

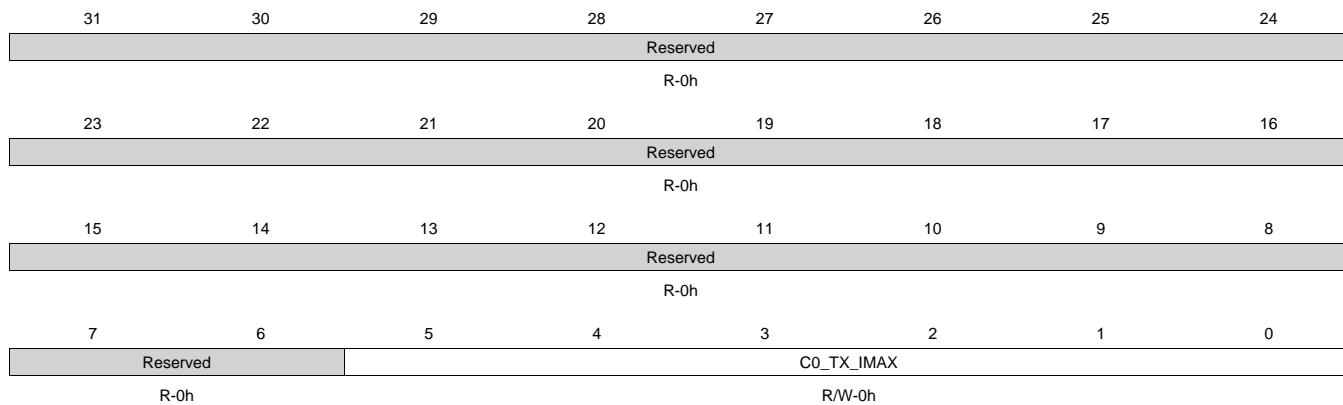
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-6 | Reserved   | R    | 0h    |  |
| 5-0  | C0_RX_IMAX | R/W  | 0h    | Core 0 Receive Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C0_RX_PULSE if pacing is enabled for this interrupt. |

#### 14.5.9.30 C0\_TX\_IMAX Register (offset = 74h) [reset = 0h]

C0\_TX\_IMAX is shown in [Figure 14-226](#) and described in [Table 14-244](#).

SUBSYSTEM CORE 0 TRANSMIT INTERRUPTS PER MILLISECOND

**Figure 14-226. C0\_TX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-244. C0\_TX\_IMAX Register Field Descriptions**

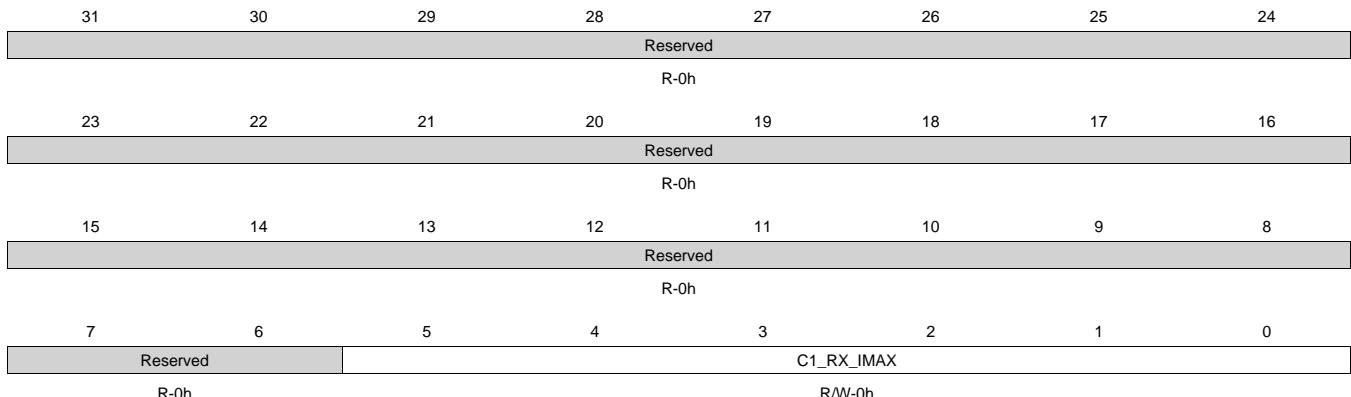
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-6 | Reserved   | R    | 0h    |   |
| 5-0  | C0_TX_IMAX | R/W  | 0h    | Core 0 Transmit Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C0_TX_PULSE if pacing is enabled for this interrupt. |

**14.5.9.31 C1\_RX\_IMAX Register (offset = 78h) [reset = 0h]**

C1\_RX\_IMAX is shown in [Figure 14-227](#) and described in [Table 14-245](#).

SUBSYSTEM CORE 1 RECEIVE INTERRUPTS PER MILLISECOND

**Figure 14-227. C1\_RX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-245. C1\_RX\_IMAX Register Field Descriptions**

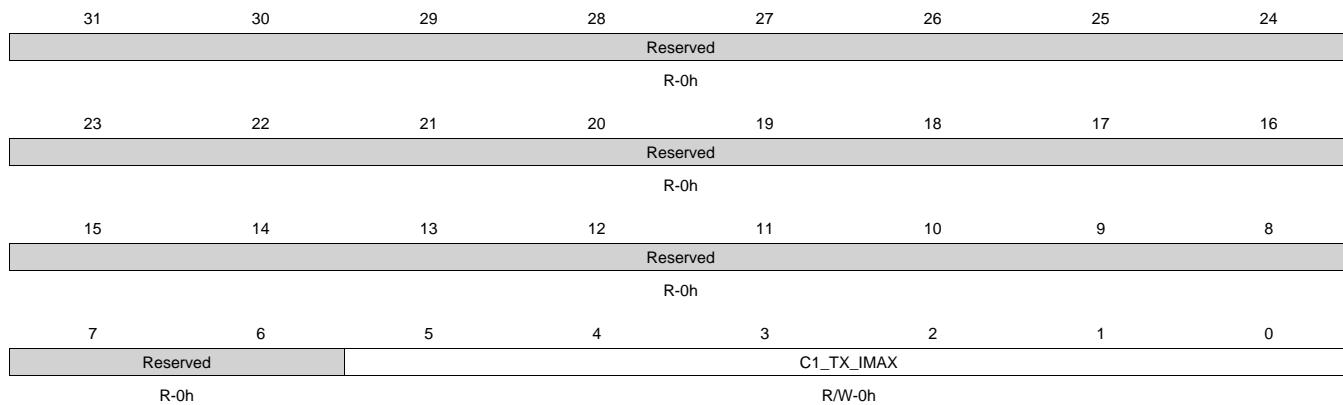
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-6 | Reserved   | R    | 0h    |  |
| 5-0  | C1_RX_IMAX | R/W  | 0h    | Core 1 Receive Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C1_RX_PULSE if pacing is enabled for this interrupt. |

#### 14.5.9.32 C1\_TX\_IMAX Register (offset = 7Ch) [reset = 0h]

C1\_TX\_IMAX is shown in [Figure 14-228](#) and described in [Table 14-246](#).

SUBSYSTEM CORE 1 TRANSMIT INTERRUPTS PER MILLISECOND

**Figure 14-228. C1\_TX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-246. C1\_TX\_IMAX Register Field Descriptions**

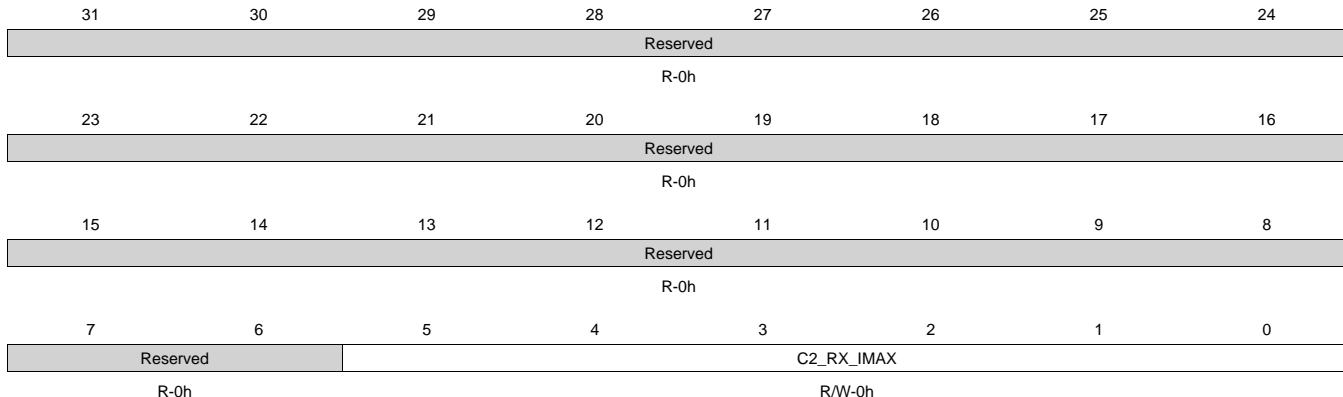
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-6 | Reserved   | R    | 0h    |   |
| 5-0  | C1_TX_IMAX | R/W  | 0h    | Core 1 Transmit Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C1_TX_PULSE if pacing is enabled for this interrupt. |

#### 14.5.9.33 C2\_RX\_IMAX Register (offset = 80h) [reset = 0h]

C2\_RX\_IMAX is shown in [Figure 14-229](#) and described in [Table 14-247](#).

SUBSYSTEM CORE 2 RECEIVE INTERRUPTS PER MILLISECOND

**Figure 14-229. C2\_RX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-247. C2\_RX\_IMAX Register Field Descriptions**

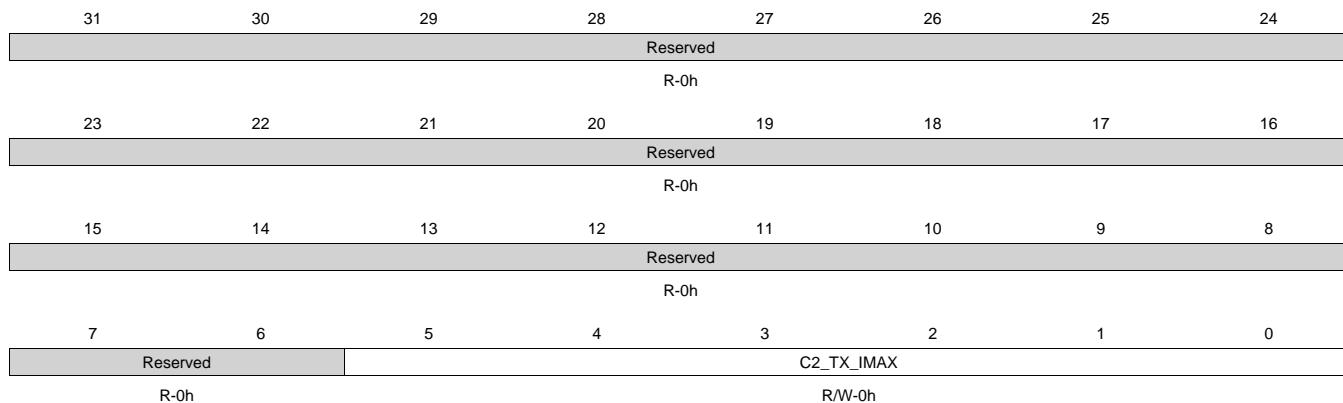
| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-6 | Reserved   | R    | 0h    |  |
| 5-0  | C2_RX_IMAX | R/W  | 0h    | Core 2 Receive Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C2_RX_PULSE if pacing is enabled for this interrupt. |

#### 14.5.9.34 C2\_TX\_IMAX Register (offset = 84h) [reset = 0h]

C2\_TX\_IMAX is shown in [Figure 14-230](#) and described in [Table 14-248](#).

SUBSYSTEM CORE 2 TRANSMIT INTERRUPTS PER MILLISECOND

**Figure 14-230. C2\_TX\_IMAX Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-248. C2\_TX\_IMAX Register Field Descriptions**

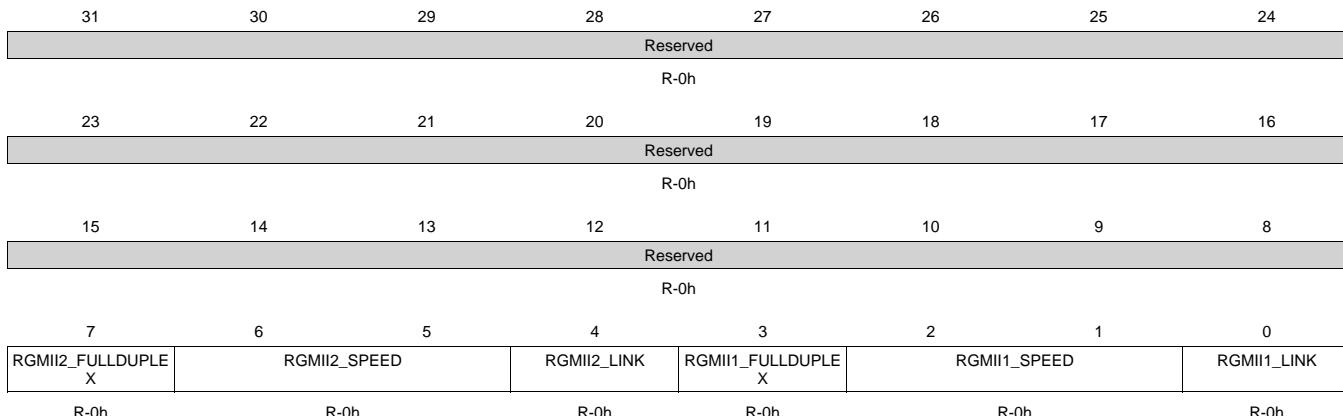
| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-6 | Reserved   | R    | 0h    |   |
| 5-0  | C2_TX_IMAX | R/W  | 0h    | Core 2 Transmit Interrupts per Millisecond - The maximum number of interrupts per millisecond generated on C2_TX_PULSE if pacing is enabled for this interrupt. |

#### 14.5.9.35 RGMII\_CTL Register (offset = 88h) [reset = 0h]

RGMII\_CTL is shown in [Figure 14-231](#) and described in [Table 14-249](#).

##### RGMII CONTROL SIGNAL REGISTER

**Figure 14-231. RGMII\_CTL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 14-249. RGMII\_CTL Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 31-8 | Reserved          | R    | 0h    |  |
| 7    | RGMII2_FULLDUPLEX | R    | 0h    | RGMII 2 Fullduplex - This is the CPRGMII fullduplex output signal.<br>0 - Half-duplex mode<br>1 - Fullduplex mode                            |
| 6-5  | RGMII2_SPEED      | R    | 0h    | RGMII2 Speed - This is the CPRGMII speed output signal<br>00 - 10Mbps mode<br>01 - 100Mbps mode<br>10 - 1000Mbps (gig) mode<br>11 - reserved |
| 4    | RGMII2_LINK       | R    | 0h    | RGMII2 Link Indicator - This is the CPRGMII link output signal<br>0 - RGMII2 link is down<br>1 - RGMII2 link is up                           |
| 3    | RGMII1_FULLDUPLEX | R    | 0h    | RGMII1 Fullduplex - This is the CPRGMII fullduplex output signal.<br>0 - Half-duplex mode<br>1 - Fullduplex mode                             |
| 2-1  | RGMII1_SPEED      | R    | 0h    | RGMII1 Speed - This is the CPRGMII speed output signal<br>00 - 10Mbps mode<br>01 - 100Mbps mode<br>10 - 1000Mbps (gig) mode<br>11 - reserved |
| 0    | RGMII1_LINK       | R    | 0h    | RGMII1 Link Indicator - This is the CPRGMII link output signal<br>0 - RGMII1 link is down  |

#### 14.5.10 Management Data Input/Output (MDIO) Registers

This section describes the memory-mapped registers for the Management Data Input/Output (MDIO).

[Table 14-250](#) lists the memory-mapped registers for the Management Data Input/Output (MDIO).

**Table 14-250. Management Data Input/Output (MDIO) Registers**

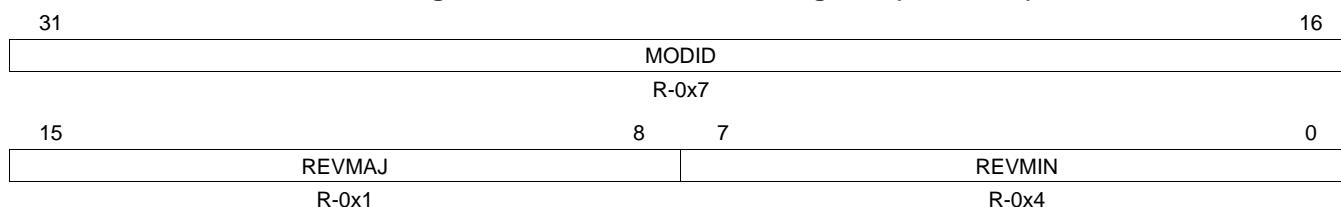
| Offset | Acronym | Register Description  | Section                           |
|--------|---------|-----------------------|-----------------------------------|
| 0      | MDIOVER | MDIO Version Register | <a href="#">Section 14.5.10.1</a> |

**Table 14-250. Management Data Input/Output (MDIO) Registers (continued)**

| Offset | Acronym            | Register Description   | Section            |
|--------|--------------------|--|--------------------|
| 4h     | MDIOCONTROL        | MDIO Control Register  | Section 14.5.10.2  |
| 8h     | MDIOALIVE          | PHY Alive Status Register                                    | Section 14.5.10.3  |
| Ch     | MDIOLINK           | PHY Link Status Register                                     | Section 14.5.10.4  |
| 10h    | MDIOLINKINTRAW     | MDIO Link Status Change Interrupt Register                   | Section 14.5.10.5  |
| 14h    | MDIOLINKINTMASKED  | MDIO Link Status Change Interrupt Register (Masked Value)    | Section 14.5.10.6  |
| 20h    | MDIOUSERINTRAW     | MDIO User Command Complete Interrupt Register (Raw Value)    | Section 14.5.10.7  |
| 24h    | MDIOUSERINTMASKED  | MDIO User Command Complete Interrupt Register (Masked Value) | Section 14.5.10.8  |
| 28h    | MDIOUSERINTMASKSET | MDIO User Command Complete Interrupt Mask Set Register       | Section 14.5.10.9  |
| 2Ch    | MDIOUSERINTMASKCLR | MDIO User Interrupt Mask Clear Register                      | Section 14.5.10.10 |
| 80h    | MDIOUSERACCESS0    | MDIO User Access Register 0                                  | Section 14.5.10.11 |
| 84h    | MDIOUSERPHYSEL0    | MDIO User PHY Select Register 0                              | Section 14.5.10.12 |
| 88h    | MDIOUSERACCESS1    | MDIO User Access Register 1                                  | Section 14.5.10.13 |
| 8Ch    | MDIOUSERPHYSEL1    | MDIO User PHY Select Register 1                              | Section 14.5.10.14 |

#### 14.5.10.1 MDIO Version Register (MDIOVER)

The MDIO version register (MDIOVER) is shown in [Figure 14-232](#) and described in [Table 14-251](#).

**Figure 14-232. MDIO Version Register (MDIOVER)**

LEGEND: R = Read only; -n = value after reset

**Table 14-251. MDIO Version Register (MDIOVER) Field Descriptions**

| Bit   | Field  | Value   | Description                                       |
|-------|--------|---------|---|
| 31-16 | MODID  | 0-FFFFh | Identifies type of peripheral.                    |
| 15-8  | REVMAJ | 0-FFh   | Management interface module major revision value. |
| 7-0   | REVMIN | 0-FFh   | Management interface module minor revision value. |

#### 14.5.10.2 MDIO Control Register (MDIOCONTROL)

The MDIO control register (MDIOCONTROL) is shown in [Figure 14-233](#) and described in [Table 14-252](#).

**Figure 14-233. MDIO Control Register (MDIOCONTROL)**

| 31    | 30      | 29       | 28                   | 24       | 23       | 21        | 20       | 19         | 18       | 17      | 16    |
|-------|---------|----------|----------------------|----------|----------|-----------|----------|------------|----------|---------|-------|
| IDLE  | ENABLE  | Reserved | HIGHEST_USER_CHANNEL | Reserved | PREAMBLE | FAULT     | FAULTENB | INTTESTENB | Reserved |         |       |
| R-0x1 | R/W-0x0 | R-0x0    | R-0x1                | R-0x0    | R/W-0x0  | RWC-0x0   | R/W-0x0  | R/W-0x0    | R/W-0x0  | R/W-0x0 | R-0x0 |
| 15    |         |          |                      |          |          |           |          |            |          |         | 0     |
|       |         |          |                      |          | CLKDIV   |           |          |            |          |         |       |
|       |         |          |                      |          |          | R/W-0x255 |          |            |          |         |       |

LEGEND: R/W = Read/Write; RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-252. MDIO Control Register (MDIOCONTROL) Field Descriptions**

| Bit   | Field                | Value   | Description   |
|-------|----------------------|---------|---|
| 31    | IDLE                 | 0       | MDIO state machine IDLE. Set to 1 when the state machine is in the idle state.  |
|       |                      | 1       | State machine is not in idle state.   |
| 30    | ENABLE               | 0       | Enable control. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit. If using byte access, the enable bit has to be the last bit written in this register. |
|       |                      | 1       | Disables the MDIO state machine.<br>Enable the MDIO state machine.  |
| 29    | Reserved             | 0       | Reserved.   |
| 28-24 | HIGHEST_USER_CHANNEL | 0-1Fh   | Highest user channel. This field specifies the highest user access channel that is available in the module and is currently set to 1. This implies that the MDIouseraccess1 register is the highest available user access channel.                    |
| 23-21 | Reserved             | 0       | Reserved.   |
| 20    | PREAMBLE             | 0       | Preamble disable.   |
|       |                      | 1       | Standard MDIO preamble is used.<br>Disables this device from sending MDIO frame pREAMBLES.  |
| 19    | FAULT                | 0       | Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit.                        |
|       |                      | 1       | No failure.<br>Physical layer fault; the MDIO state machine is reset.   |
| 18    | FAULTENB             | 0       | Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection.  |
|       |                      | 1       | Disables the physical layer fault detection.<br>Enables the physical layer fault detection.   |
| 17    | INTTESTENB           | 0       | Interrupt test enable. This bit can be set to 1 to enable the host to set the USERINT and LINKINT bits for test purposes.   |
|       |                      | 1       | Interrupt bits are not set.<br>Enables the host to set the USERINT and LINKINT bits for test purposes.  |
| 16    | Reserved             | 0       | Reserved.   |
| 15-0  | CLKDIV               | 0-FFFFh | Clock divider. This field specifies the division ratio between CLK and the frequency of MDCLK. MDCLK is disabled when clkdiv is set to 0. MDCLK frequency = clk frequency/(clkdiv+1).   |

#### 14.5.10.3 PHY Acknowledge Status Register (MDIOALIVE)

The PHY acknowledge status register (MDIOALIVE) is shown in [Figure 14-234](#) and described in [Table 14-253](#).

**Figure 14-234. PHY Acknowledge Status Register (MDIOALIVE)**

|         |   |
|---------|---|
| 31      | 0 |
| ALIVE   |   |
| RWC-0x0 |   |

LEGEND: RWC = Read/Write/Clear

**Table 14-253. PHY Acknowledge Status Register (MDIOALIVE) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | ALIVE | 0xFFFF FFFFh | MDIO alive. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY, the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect. |

#### 14.5.10.4 PHY Link Status Register (MDIOLINK)

The PHY link status register (MDIOLINK) is shown in [Figure 14-235](#) and described in [Table 14-254](#).

**Figure 14-235. PHY Link Status Register (MDIOLINK)**

|       |   |
|-------|---|
| 31    | 0 |
| LINK  |   |
| R-0x0 |   |

LEGEND: R = Read only; -n = value after reset

**Table 14-254. PHY Link Status Register (MDIOLINK) Field Descriptions**

| Bit  | Field | Value        | Description   |
|------|-------|--------------|---|
| 31-0 | LINK  | 0xFFFF FFFFh | MDIO link state. This register is updated after a read of the Generic Status Register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect. In addition, the status of the two PHYs specified in the MDIOUSERPHYSEL $n$ registers can be determined using the MLINK input pins. This is determined by the LINKSEL bit in the MDIOUSERPHYSEL $n$ register. |

#### 14.5.10.5 MDIO Link Status Change Interrupt Register (MDIOLINKINTRAW)

The MDIO link status change interrupt register (MDIOLINKINTRAW) is shown in [Figure 14-236](#) and described in [Table 14-255](#).

**Figure 14-236. MDIO Link Status Change Interrupt Register (MDIOLINKINTRAW)**

|    |          |                       |
|----|----------|-----------------------|
| 31 | Reserved | 16                    |
|    | R-0x0    |                       |
| 15 | Reserved | 2      1      0       |
|    | R-0x0    | LINKINTRAW<br>RWC-0x0 |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-255. MDIO Link Status Change Interrupt Register (MDIOLINKINTRAW) Field Descriptions**

| Bit  | Field      | Value | Description   |
|------|------------|-------|---|
| 31-2 | Reserved   | 0     | Reserved.   |
| 1-0  | LINKINTRAW | 0-3h  | MDIO link change event, raw value. When asserted 1, a bit indicates that there was an MDIO link change event (i.e. change in the MDIOLINK register) corresponding to the PHY address in the MDIOUSERPHYSEL $n$ register. LINKINTRAW[0] and LINKINTRAW[1] correspond to MDIOUSERPHYSEL0 and MDIOUSERPHYSEL1, respectively. Writing a 1 will clear the event and writing 0 has no effect. If the INTTESTENB bit in the MDIOCONTROL register is set, the host may set the LINKINTRAW bits to a 1. This mode may be used for test purposes. |

#### 14.5.10.6 MDIO Link Status Change Interrupt Register (Masked Value) (MDIOLINKINTMASKED)

The MDIO link status change interrupt register (Masked Value) (MDIOLINKINTMASKED) is shown in [Figure 14-237](#) and described in [Table 14-256](#).

**Figure 14-237. MDIO Link Status Change Interrupt Register (Masked Value) (MDIOLINKINTMASKED)**

|    |          |                          |
|----|----------|--------------------------|
| 31 | Reserved | 16                       |
|    | R-0x0    |                          |
| 15 | Reserved | 2      1      0          |
|    | R-0x0    | LINKINTMASKED<br>RWC-0x0 |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-256. MDIO Link Status Change Interrupt Register (Masked Value) (MDIOLINKINTMASKED) Field Descriptions**

| Bit  | Field         | Value | Description   |
|------|---------------|-------|---|
| 31-2 | Reserved      | 0     | Reserved.   |
| 1-0  | LINKINTMASKED | 0-3h  | MDIO link change interrupt, masked value. When asserted 1, a bit indicates that there was an MDIO link change event (i.e. change in the MDIO Link register) corresponding to the PHY address in the MDIOUSERPHYSEL $n$ register and the corresponding LINKINTENB bit was set. LINKINTMASKED[0] and LINKINTMASKED[1] correspond to MDIOUSERPHYSEL0 and MDIOUSERPHYSEL1, respectively. Writing a 1 will clear the interrupt and writing 0 has no effect. If the INTTESTENB bit in the MDIOCONTROL register is set, the host may set the LINKINT bits to a 1. This mode may be used for test purposes. |

#### 14.5.10.7 MDIO User Command Complete Interrupt Register (Raw Value) (MDIOUSERINTRAW)

The MDIO user command complete interrupt register (Raw Value) (MDIOUSERINTRAW) is shown in Figure 14-238 and described in Table 14-257.

**Figure 14-238. MDIO User Command Complete Interrupt Register (Raw Value) (MDIOUSERINTRAW)**

|    |          |         |
|----|----------|---------|
| 31 | Reserved | 16      |
|    | R-0x0    |         |
| 15 | Reserved | 2       |
|    | R-0x0    | 1       |
|    |          | 0       |
|    |          | RWC-0x0 |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-257. MDIO User Command Complete Interrupt Register (Raw Value) (MDIOUSERINTRAW) Field Descriptions**

| Bit  | Field      | Value | Description   |
|------|------------|-------|---|
| 31-2 | Reserved   | 0     | Reserved.   |
| 1-0  | USERINTRAW | 0-3h  | Raw value of MDIO user command complete event for the MDIOUSERACCESS1 register through the MDIOUSERACCESS0 register, respectively. When asserted 1, a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUSERACCESSn register has completed. Writing a 1 will clear the event and writing 0 has no effect. If the INTTESTENB bit in the MDIOCONTROL register is set, the host may set the USERINTRAW bits to a 1. This mode may be used for test purposes. |

#### 14.5.10.8 MDIO User Command Complete Interrupt Register (Masked Value) (MDIOUSERINTMASKED)

The MDIO user command complete interrupt register (Masked Value) (MDIOUSERINTMASKED) is shown in Figure 14-239 and described in Table 14-258.

**Figure 14-239. MDIO User Command Complete Interrupt Register (Masked Value) (MDIOUSERINTMASKED)**

|    |          |         |
|----|----------|---------|
| 31 | Reserved | 16      |
|    | R-0x0    |         |
| 15 | Reserved | 2       |
|    | R-0x0    | 1       |
|    |          | 0       |
|    |          | RWC-0x0 |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-258. MDIO User Command Complete Interrupt Register (Masked Value) (MDIOUSERINTMASKED) Field Descriptions**

| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 31-2 | Reserved      | 0     | Reserved.  |
| 1-0  | USERINTMASKED | 0-3h  | Masked value of MDIO user command complete interrupt for the MDIOUSERACCESS1 register through the MDIOUSERACCESS0 register, respectively. When asserted 1, a bit indicates that the previously scheduled PHY read or write command using that particular MDIOUSERACCESSn register has completed and the corresponding USERINTMASKSET bit is set to 1. Writing a 1 will clear the interrupt and writing 0 has no effect. If the INTTESTENB bit in the MDIOCONTROL register is set, the host may set the USERINTMASKED bits to a 1. This mode may be used for test purposes. |

#### 14.5.10.9 MDIO User Command Complete Interrupt Mask Set Register (MDIOUSERINTMASKSET)

The MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET) is shown in Figure 14-240 and described in Table 14-259.

**Figure 14-240. MDIO User Command Complete Interrupt Mask Set Register (MDIOUSERINTMASKSET)**

|    |          |   |   |                |
|----|----------|---|---|----------------|
| 31 | Reserved |   |   | 16             |
|    | R-0x0    |   |   |                |
| 15 | 2        | 1 | 0 |                |
|    | Reserved |   |   | USERINTMASKSET |
|    | R-0x0    |   |   | RWC-0x0        |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

**Table 14-259. MDIO User Command Complete Interrupt Mask Set Register (MDIOUSERINTMASKSET) Field Descriptions**

| Bit  | Field          | Value | Description   |
|------|----------------|-------|---|
| 31-2 | Reserved       | 0     | Reserved.   |
| 1-0  | USERINTMASKSET | 0-3h  | MDIO user interrupt mask set for USERINTMASKED, respectively. Writing a bit to 1 will enable MDIO user command complete interrupts for that particular MDIOUSERACCESS $n$ register. MDIO user interrupt for a particular MDIOUSERACCESS $n$ register is disabled if the corresponding bit is 0. Writing a 0 to this register has no effect. |

#### 14.5.10.10 MDIO User Command Complete Interrupt Mask Clear Register (MDIOUSERINTMASKCLR)

The MDIO user interrupt mask clear register (MDIOUSERINTMASKCLR) is shown in Figure 14-241 and described in Table 14-260.

**Figure 14-241. MDIO User Command Complete Interrupt Mask Clear Register (MDIOUSERINTMASKCLR)**

|    |          |   |   |                  |
|----|----------|---|---|------------------|
| 31 | Reserved |   |   | 16               |
|    | R-0x0    |   |   |                  |
| 15 | 2        | 1 | 0 |                  |
|    | Reserved |   |   | USERINTMASKCLEAR |
|    | R-0x0    |   |   | RWC-0x0          |

LEGEND: RWC = Read/Write/Clear; R = Read only; -n = value after reset

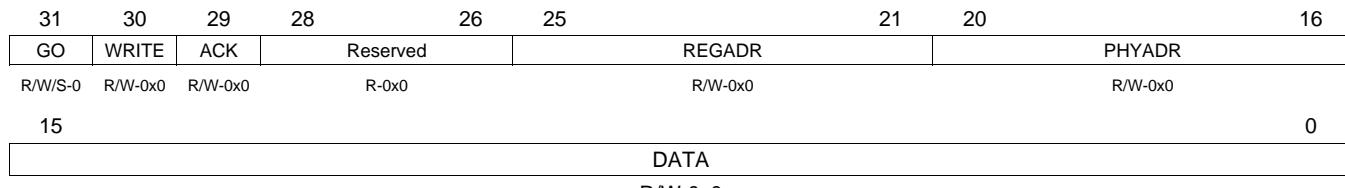
**Table 14-260. MDIO User Command Complete Interrupt Mask Clear Register (MDIOUSERINTMASKCLR) Field Descriptions**

| Bit  | Field            | Value | Description  |
|------|------------------|-------|--|
| 31-2 | Reserved         | 0     | Reserved.  |
| 1-0  | USERINTMASKCLEAR | 0-3h  | MDIO user command complete interrupt mask clear for USERINTMASKED, respectively. Writing a bit to 1 will disable further user command complete interrupts for that particular MDIOUSERACCESS $n$ register. Writing a 0 to this register has no effect. |

#### 14.5.10.11 MDIO User Access Register 0 (MDIouserAccess0)

The MDIO user access register 0 (MDIouserAccess0) is shown in [Figure 14-242](#) and described in [Table 14-261](#).

**Figure 14-242. MDIO User Access Register 0 (MDIouserAccess0)**



LEGEND: R/W = Read/Write; R = Read only; S = Status; -n = value after reset

**Table 14-261. MDIO User Access Register 0 (MDIouserAccess0) Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31    | GO       | 0-1     | Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIouserAccess0 register are blocked when the GO bit is 1. If byte access is being used, the GO bit should be written last. |
| 30    | WRITE    | 0-1     | Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.   |
| 29    | ACK      | 0-1     | Acknowledge. This bit is set if the PHY acknowledged the read transaction.   |
| 28-26 | Reserved | 0       | Reserved.  |
| 25-21 | REGADR   | 0-1Fh   | Register address. Specifies the PHY register to be accessed for this transaction.  |
| 20-16 | PHYADR   | 0-1Fh   | PHY address. Specifies the PHY to be accessed for this transaction.  |
| 15-0  | DATA     | 0-FFFFh | User data. The data value read from or to be written to the specified PHY register.  |

#### 14.5.10.12 MDIO User PHY Select Register 0 (MDIouserPHYSEL0)

The MDIO user PHY select register 0 (MDIouserPHYSEL0) is shown in [Figure 14-243](#) and described in [Table 14-262](#).

**Figure 14-243. MDIO User PHY Select Register 0 (MDIouserPHYSEL0)**

|       |          |   |         |         |       |   |         |    |  |  |  |  |  |  |  |
|-------|----------|---|---------|---------|-------|---|---------|----|--|--|--|--|--|--|--|
| 31    | Reserved |   |         |         |       |   |         | 16 |  |  |  |  |  |  |  |
| R-0x0 |          |   |         |         |       |   |         |    |  |  |  |  |  |  |  |
|       |          |   |         |         |       |   |         |    |  |  |  |  |  |  |  |
| 15    | Reserved | 8 | 7       | 6       | 5     | 4 | 0       |    |  |  |  |  |  |  |  |
|       | R-0x0    |   | R/W-0x0 | R/W-0x0 | R-0x0 |   | R/W-0x0 |    |  |  |  |  |  |  |  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

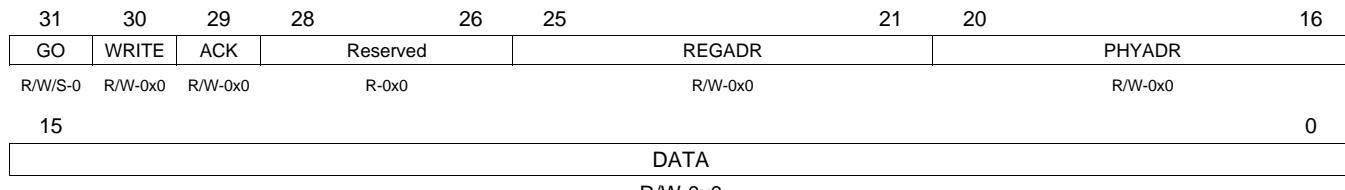
**Table 14-262. MDIO User PHY Select Register 0 (MDIouserPHYSEL0) Field Descriptions**

| Bit  | Field      | Value | Description   |
|------|------------|-------|---|
| 31-8 | Reserved   | 0     | Reserved.   |
| 7    | LINKSEL    | 0-1   | Link status determination select. Set to 1 to determine link status using the MLINK pin. Default value is 0 which implies that the link status is determined by the MDIO state machine.   |
| 6    | LINKINTENB |       | Link change interrupt enable. Set to 1 to enable link change status interrupts for PHY address specified in PHYADDRMON. Link change interrupts are disabled if this bit is set to 0.<br><br>0 Link change interrupts are disabled.<br><br>1 Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled. |
| 5    | Reserved   | 0     | Reserved.   |
| 4-0  | PHYADDRMON | 0-1Fh | PHY address whose link status is to be monitored.   |

#### 14.5.10.13 MDIO User Access Register 1 (MDIOUSERACCESS1)

The MDIO user access register 1 (MDIOUSERACCESS1) is shown in [Figure 14-244](#) and described in [Table 14-263](#).

**Figure 14-244. MDIO User Access Register 1 (MDIOUSERACCESS1)**



LEGEND: R/W = Read/Write; R = Read only; S = Status; -n = value after reset

**Table 14-263. MDIO User Access Register 1 (MDIOUSERACCESS1) Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31    | GO       | 0-1     | Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUSERACCESS0 register are blocked when the GO bit is 1. If byte access is being used, the GO bit should be written last. |
| 30    | WRITE    | 0-1     | Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.   |
| 29    | ACK      | 0-1     | Acknowledge. This bit is set if the PHY acknowledged the read transaction.   |
| 28-26 | Reserved | 0       | Reserved.  |
| 25-21 | REGADR   | 0-1Fh   | Register address; specifies the PHY register to be accessed for this transaction.  |
| 20-16 | PHYADR   | 0-1Fh   | PHY address; specifies the PHY to be accessed for this transaction.  |
| 15-0  | DATA     | 0-FFFFh | User data. The data value read from or to be written to the specified PHY register.  |

#### 14.5.10.14 MDIO User PHY Select Register 1 (MDIouserphySel1)

The MDIO user PHY select register 1 (MDIOUSERPHYSEL1) is shown in [Figure 14-245](#) and described in [Table 14-264](#).

**Figure 14-245. MDIO User PHY Select Register 1 (MDIOUSERPHYSEL1)**

|    |          |   |         |            |          |            |   |    |
|----|----------|---|---------|------------|----------|------------|---|----|
| 31 | Reserved |   |         |            |          |            |   | 16 |
|    | R-0x0    |   |         |            |          |            |   |    |
| 15 |          | 8 | 7       | 6          | 5        | 4          | 0 |    |
|    | Reserved |   | LINKSEL | LINKINTENB | Reserved | PHYADDRMON |   |    |
|    | R-0x0    |   | R/W-0x0 | R/W-0x0    | R-0x0    | R/W-0x0    |   |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 14-264. MDIO User PHY Select Register 1 (MDIouserPhySel1) Field Descriptions**

| Bit  | Field      | Value | Description   |
|------|------------|-------|---|
| 31-8 | Reserved   | 0     | Reserved.   |
| 7    | LINKSEL    | 0-1   | Link status determination select. Set to 1 to determine link status using the MLINK pin. Default value is 0 which implies that the link status is determined by the MDIO state machine.   |
| 6    | LINKINTENB |       | <p>Link change interrupt enable. Set to 1 to enable link change status interrupts for PHY address specified in PHYADDRMON. Link change interrupts are disabled if this bit is cleared to 0.</p> <p>0 Link change interrupts are disabled</p> <p>1 Link change status interrupts for PHY address specified in PHYADDRMON bits are enabled.</p> |
| 5    | Reserved   | 0     | Reserved.   |
| 4-0  | PHYADDRMON | 0-1Fh | PHY address whose link status is to be monitored.   |

## Pulse-Width Modulation Subsystem (PWMSS)

This chapter describes the PWMSS of the device.

| Topic  | Page |
|--|------|
| 15.1 Pulse-Width Modulation Subsystem (PWMSS) .....        | 1478 |
| 15.2 Enhanced PWM (ePWM) Module .....                      | 1486 |
| 15.3 Enhanced Capture (eCAP) Module .....                  | 1599 |
| 15.4 Enhanced Quadrature Encoder Pulse (eQEP) Module ..... | 1638 |

## 15.1 Pulse-Width Modulation Subsystem (PWMSS)

### 15.1.1 Introduction

#### 15.1.1.1 Features

The general features of the PWMSS are:

##### eHRPWM

- Dedicated 16 bit time-base with Period / Frequency control
- Can support 2 independent PWM outputs with Single edge operation
- Can support 2 independent PWM outputs with Dual edge symmetric operation
- Can support 1 independent PWM output with Dual edge asymmetric operation
- Supports Dead-band generation with independent Rising and Falling edge delay control
- Provides asynchronous over-ride control of PWM signals during fault conditions
- Supports “trip zone” allocation of both latched and un-latched fault conditions
- Allows events to trigger both CPU interrupts and start of ADC conversions
- Support PWM chopping by high frequency carrier signal, used for pulse transformer gate drives.
- High-resolution module with programmable delay line.
  - Programmable on a per PWM period basis.
  - Can be inserted either on the rising edge or falling edge of the PWM pulse or both or not at all.

##### eCAP

- Dedicated input Capture pin
- 32 bit Time Base (counter)
- 4 x 32 bit Time-stamp Capture registers (CAP1-CAP4)
- 4 stage sequencer (Mod4 counter) which is synchronized to external events (ECAPx pin edges)
- Independent Edge polarity (Rising / Falling edge) selection for all 4 events
- Input Capture signal pre-scaling (from 1 to 16)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 Time-stamp events
- Control for continuous Time-stamp captures using a 4 deep circular buffer (CAP1-CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

##### eQEP

- Input Synchronization
- Three Stage/Six Stage Digital Noise Filter
- Quadrature Decoder Unit
- Position Counter and Control unit for position measurement
- Quadrature Edge Capture unit for low speed measurement
- Unit Time base for speed/frequency measurement
- Watchdog Timer for detecting stalls

### 15.1.1.2 Unsupported Features

The PWMSS module features not supported are shown in [Table 15-1](#).

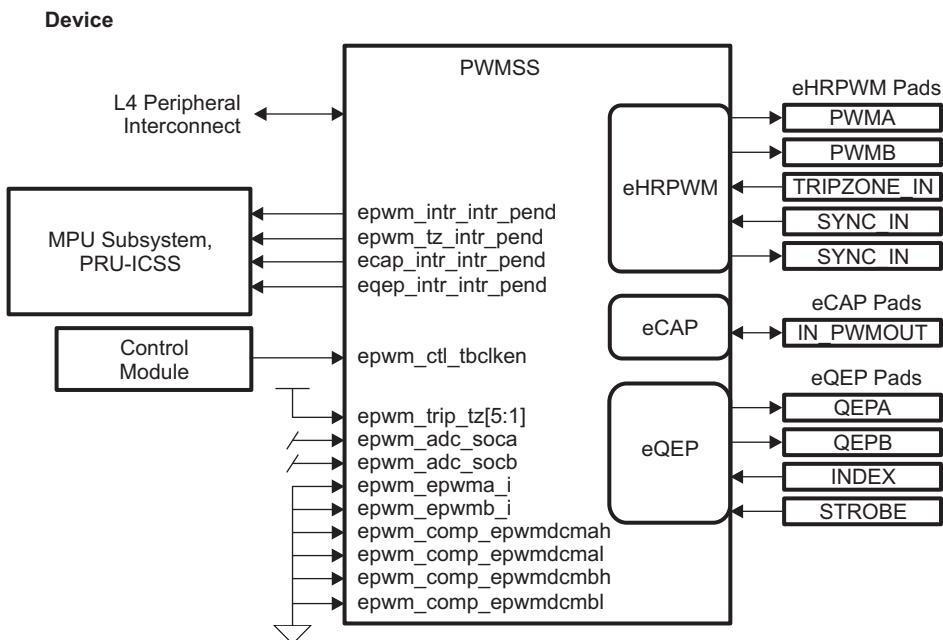
**Table 15-1. Unsupported Features**

| Feature                  | Reason                           |
|--------------------------|----------------------------------|
| ePWM inputs              | Not pinned out                   |
| ePWM tripzone 1-5 inputs | Only Tripzone0 is pinned out     |
| ePWM digital comparators | Inputs not connected             |
| eQEP quadrature outputs  | Only input signals are connected |

### 15.1.2 Integration

The Pulse Width Modulation Subsystem (PWMSS) includes a single instance of the Enhanced High Resolution Pulse Width Modulator (eHRPWM), Enhanced Capture (eCAP), and Enhanced Quadrature Encoded Pulse (eQEP) modules. This includes three instantiations of the PWMSS.

**Figure 15-1. PWMSS Integration**



#### 15.1.2.1 PWMSS Connectivity Attributes

The general connectivity attributes for the PWMSS module are shown in [Table 15-2](#).

**Table 15-2. PWMSS Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain   |
| Clock Domain        | PD_PER_L4LS_GCLK  |
| Reset Signals       | PER_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle  |
| Interrupt Requests  | 2 ePWM interrupts per instance<br>epwm_intr_intr - Event interrupt, ePWMxINT for ARM subsystem, epwm_intr_intr_pend for PRU-ICSS<br>epwm_tz_intr - Tripzone interrupt, ePWMx_TZINT for ARM subsystem, pwm_trip_zone for PRU-ICSS (only 1 for all 3 instances)<br>1 eCAP interrupt per instance<br>ecap_intr - Capture/PWM event interrupt, eCAPxINT for ARM subsystem, ecap_intr_intr_pend for PRU-ICSS<br>1 eQEP Interrupt per instance<br>eqep_intr_intr - Event interrupt, eQEPxINT for ARM subsystem, eqep_intr_intr_pend for PRU-ICSS (only for eQEP0) |
| DMA Requests        | Interrupt requests are redirected as DMA requests:<br><ul style="list-style-type: none"> <li>• 1 DMA request from ePWM per instance (ePMEVTx)</li> <li>• 1 DMA request from eCAP per instance (eCAPEVTx)</li> <li>• 1 DMA request from eQEP per instance (eQEPEVTx)</li> </ul>  |
| Physical Address    | L4 Peripheral slave port  |

### 15.1.2.2 PWMSS Clock and Reset Management

The PWMSS controllers have separate bus interface and functional clocks.

**Table 15-3. PWMSS Clock Signals**

| Clock Signal                                  | Max Freq | Reference / Source | Comments                      |
|---|----------|--------------------|-------------------------------|
| PWMSS_ocp_clk<br>Interface / Functional clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_I4ls_gclk<br>from PRCM |

### 15.1.2.3 PWMSS Pin list

The external signals for the PWMSS module are shown in the following table.

**Table 15-4. PWMSS Pin List**

| Pin                | Type* | Description                     |
|--------------------|-------|---------------------------------|
| EPWMxA             | O     | PWM output A                    |
| EPWMxB             | O     | PWM output B                    |
| EPWM_SYNCIN        | I     | PWM Sync input                  |
| EPWM_SYNCOUT       | O     | PWM Sync output                 |
| EPWM_TRIPZONE[5:0] | I     | PWM Tripzone inputs             |
| ECAP_CAPIN_APWMOUT | I/O   | eCAP Capture input / PWM output |
| EQEP_A             | I/O   | eQEP Quadrature input/output    |
| EQEP_B             | I/O   | eQEP Quadrature input/output    |
| EQEP_INDEX         | I/O   | eQEP Index input/output         |
| EQEP_STROBE        | I/O   | eQEP Strobe input/output        |

### 15.1.3 PWMSS Registers

Table 15-5 lists the memory-mapped registers for the PWMSS. All register offset addresses not listed in Table 15-5 should be considered as reserved locations and the register contents should not be modified.

**Table 15-5. PWMSS REGISTERS**

| Offset | Acronym   | Register Name                 | Section          |
|--------|-----------|-------------------------------|------------------|
| 0h     | IDVER     | IP Revision Register          | Section 15.1.3.1 |
| 4h     | SYSCONFIG | System Configuration Register | Section 15.1.3.2 |
| 8h     | CLKCONFIG | Clock Configuration Register  | Section 15.1.3.3 |
| Ch     | CLKSTATUS | Clock Status Register         | Section 15.1.3.4 |

### 15.1.3.1 IDVER Register (offset = 0h) [reset = 40000000h]

IDVER is shown in Figure 15-2 and described in Table 15-6.

The IP revision register is used by software to track features, bugs, and compatibility.

**Figure 15-2. IDVER Register**

| 31     | 30 | 29       | 28 | 27 | 26      | 25 | 24 |
|--------|----|----------|----|----|---------|----|----|
| SCHEME |    | Reserved |    |    | FUNC    |    |    |
| R-1h   |    | R-0h     |    |    | R-0h    |    |    |
| 23     | 22 | 21       | 20 | 19 | 18      | 17 | 16 |
|        |    | FUNC     |    |    | R-0h    |    |    |
| 15     | 14 | 13       | 12 | 11 | 10      | 9  | 8  |
|        |    | R RTL    |    |    | X_MAJOR |    |    |
|        |    | R-0h     |    |    | R-0h    |    |    |
| 7      | 6  | 5        | 4  | 3  | 2       | 1  | 0  |
| CUSTOM |    | Y_MINOR  |    |    | R-0h    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 15-6. IDVER Register Field Descriptions**

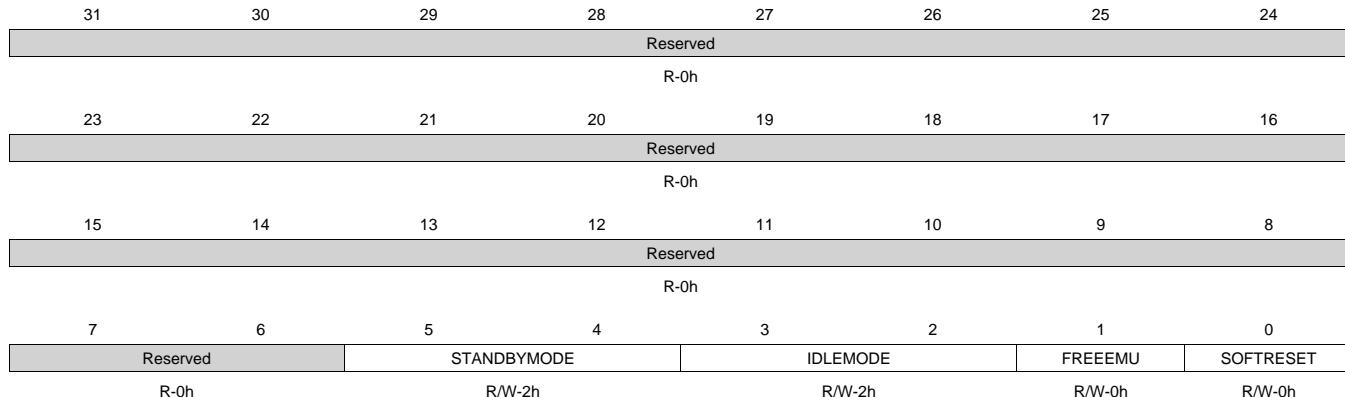
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-30 | SCHEME   | R    | 1h    | Used to distinguish between the old scheme and current. |
| 29-28 | Reserved | R    | 0h    |   |
| 27-16 | FUNC     | R    | 0h    | FUNC  |
| 15-11 | R RTL    | R    | 0h    | RTL version (R), maintained by IP design owner.         |
| 10-8  | X_MAJOR  | R    | 0h    | Major revision (X)                                      |
| 7-6   | CUSTOM   | R    | 0h    | CUSTOM  |
| 5-0   | Y_MINOR  | R    | 0h    | Minor revision (Y)                                      |

### 15.1.3.2 SYSCONFIG Register (offset = 4h) [reset = 28h]

SYSCONFIG is shown in [Figure 15-3](#) and described in [Table 15-7](#).

The system configuration register is used for clock management configuration.

**Figure 15-3. SYSCONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 15-7. SYSCONFIG Register Field Descriptions**

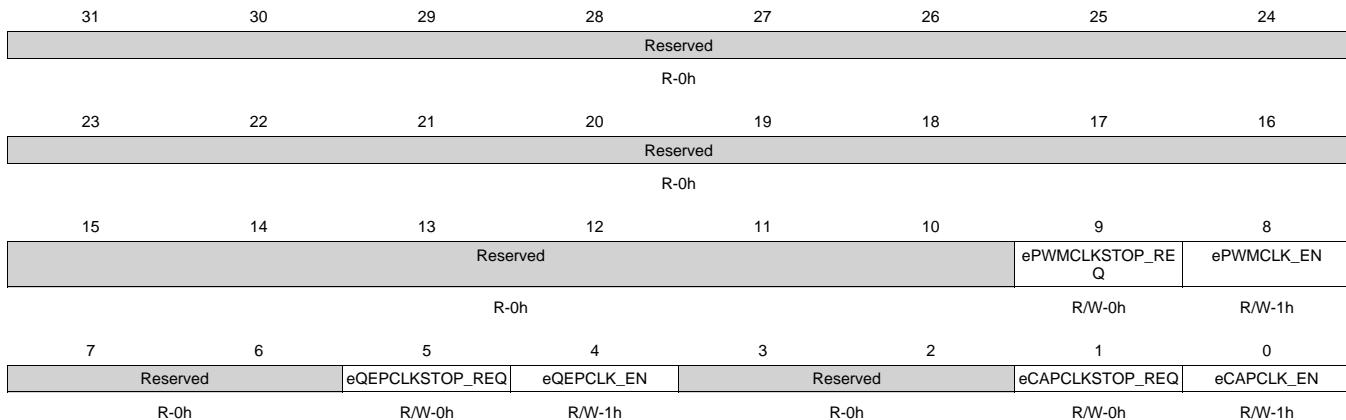
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-6 | Reserved    | R    | 0h    |  |
| 5-4  | STANDBYMODE | R/W  | 2h    | Configuration of the local initiator state management mode.<br>By definition, initiator may generate read/write transaction as long as it is out of STANDBY state.<br>0x0 = Force-standby mode: local initiator is unconditionally placed in standby state. Backup mode, for debug only.<br>0x1 = No-standby mode: local initiator is unconditionally placed out of standby state. Backup mode, for debug only.<br>0x2 = Smart-standby mode: local initiator standby status depends on local conditions, i.e., the module's functional requirement from the initiator. IP module should not generate (initiator-related) wakeup events.<br>0x3 = Reserved.   |
| 3-2  | IDLEMODE    | R/W  | 2h    | Configuration of the local target state management mode.<br>By definition, target can handle read/write transaction as long as it is out of IDLE state.<br>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only.<br>0x1 = No-idle mode: local target never enters idle state. Backup mode, for debug only.<br>0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events.<br>0x3 = Reserved. |
| 1    | FREEEMU     | R/W  | 0h    | Sensitivity to emulation (debug) suspend input signal.<br>0x0 = IP module is sensitive to emulation suspend.<br>0x1 = IP module is not sensitive to emulation suspend.   |
| 0    | SOFTRESET   | R/W  | 0h    | Software reset (optional)  |

### 15.1.3.3 CLKCONFIG Register (offset = 8h) [reset = 111h]

CLKCONFIG is shown in [Figure 15-4](#) and described in [Table 15-8](#).

The clock configuration register is used in the PWMSS submodule for clkstop req and clk\_en control.

**Figure 15-4. CLKCONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 15-8. CLKCONFIG Register Field Descriptions**

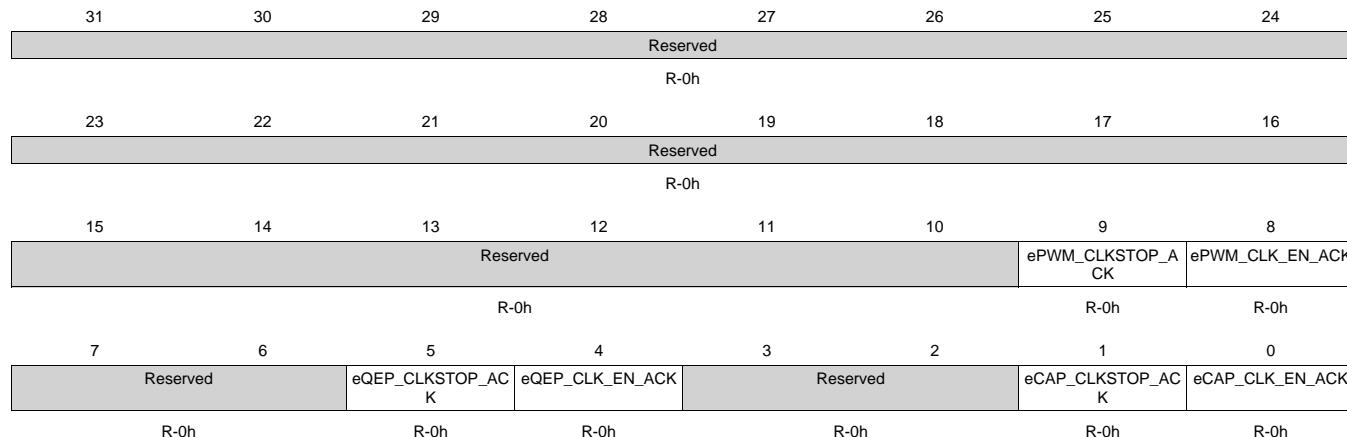
| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-10 | Reserved        | R    | 0h    |   |
| 9     | ePWMCLKSTOP_REQ | R/W  | 0h    | This bit controls the clkstop_req input to the ePWM module. |
| 8     | ePWMCLK_EN      | R/W  | 1h    | This bit controls the clk_en input to the ePWM module.      |
| 7-6   | Reserved        | R    | 0h    |   |
| 5     | eQEPCLKSTOP_REQ | R/W  | 0h    | This bit controls the clkstop_req input to the eQEP module  |
| 4     | eQEPCLK_EN      | R/W  | 1h    | This bit controls the clk_en input to the eQEP module.      |
| 3-2   | Reserved        | R    | 0h    |   |
| 1     | eCAPCLKSTOP_REQ | R/W  | 0h    | This bit controls the clkstop_req input to the eCAP module. |
| 0     | eCAPCLK_EN      | R/W  | 1h    | This bit controls the clk_en input to the eCAP module.      |

#### 15.1.3.4 CLKSTATUS Register (offset = Ch) [reset = 0h]

CLKSTATUS is shown in [Figure 15-5](#) and described in [Table 15-9](#).

The clock status register is used in the PWMSS submodule for clkstop ack and clk\_en ack status.

**Figure 15-5. CLKSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 15-9. CLKSTATUS Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-10 | Reserved         | R    | 0h    |   |
| 9     | ePWM_CLKSTOP_ACK | R    | 0h    | This bit is the clkstop_req_ack status output of the ePWM module. |
| 8     | ePWM_CLK_EN_ACK  | R    | 0h    | This bit is the clk_en status output of the ePWM module.          |
| 7-6   | Reserved         | R    | 0h    |   |
| 5     | eQEP_CLKSTOP_ACK | R    | 0h    | This bit is the clkstop_req_ack status output of the eQEP module. |
| 4     | eQEP_CLK_EN_ACK  | R    | 0h    | This bit is the clk_en status output of the eQEP module.          |
| 3-2   | Reserved         | R    | 0h    |   |
| 1     | eCAP_CLKSTOP_ACK | R    | 0h    | This bit is the clkstop_req_ack status output of the eCAP module. |
| 0     | eCAP_CLK_EN_ACK  | R    | 0h    | This bit is the clk_en status output of the eCAP module.          |

## 15.2 Enhanced PWM (ePWM) Module

### 15.2.1 Introduction

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel modules with separate resources and that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this chapter, the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and, likewise, EPWM4A and EPWM4B belong to ePWM4.

#### 15.2.1.1 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in [Figure 15-6](#). Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in [Section 15.2.2.10](#). See [Section 15.1.2](#) to determine which ePWM instances include this feature. Each ePWM module is indicated by a numerical value starting with 1. For example ePWM0 is the first instance and ePWM2 is the third instance in the system and ePWMx indicates any instance.

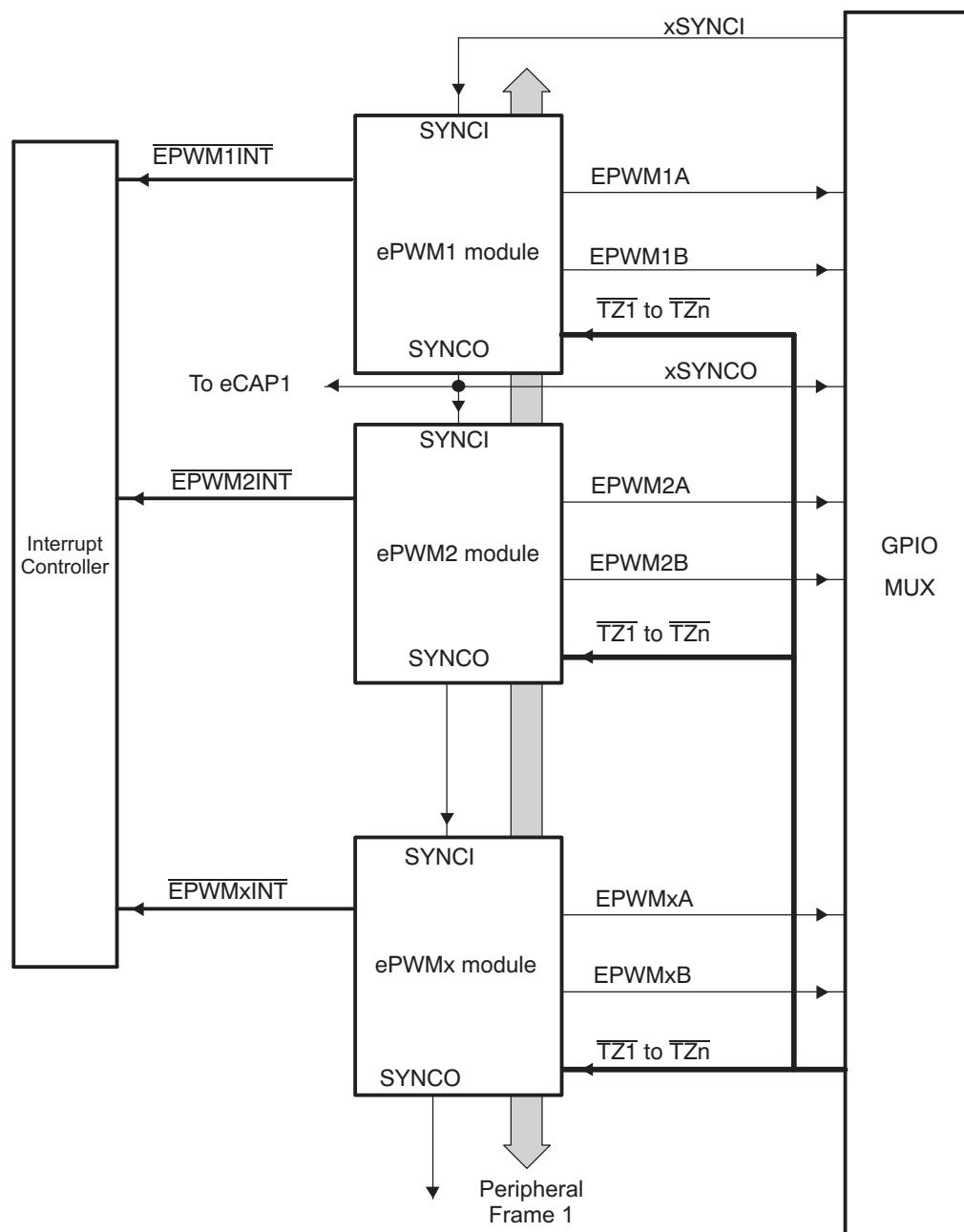
The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

Each ePWM module supports the following features:

- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations:
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in [Figure 15-6](#). The signals are described in detail in subsequent sections.

The order in which the ePWM modules are connected may differ from what is shown in [Figure 15-6](#). See [Section 15.2.2.3.3.2](#) for the synchronization scheme for a particular device. Each ePWM module consists of seven submodules and is connected within a system via the signals shown in [Figure 15-7](#).

**Figure 15-6. Multiple ePWM Modules**


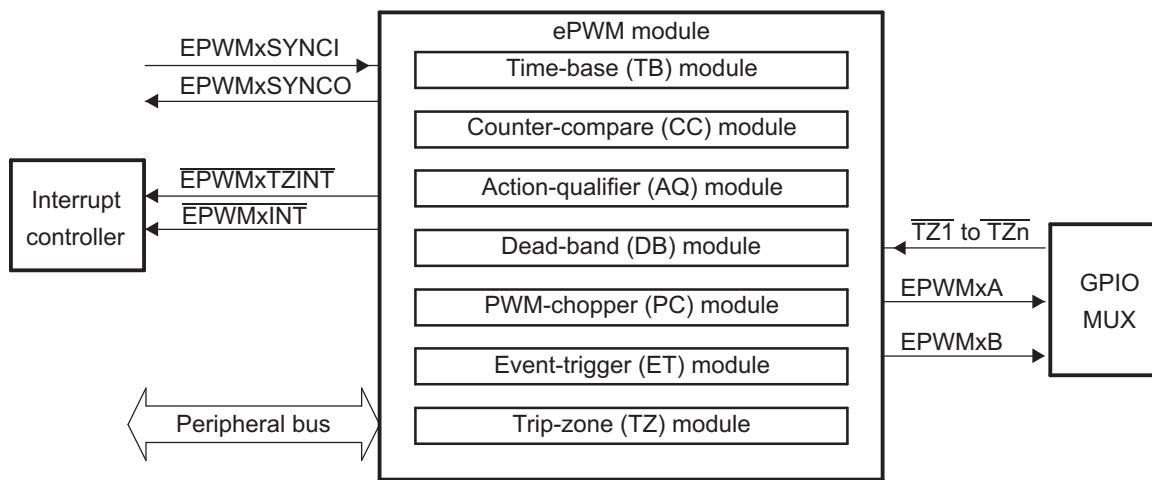
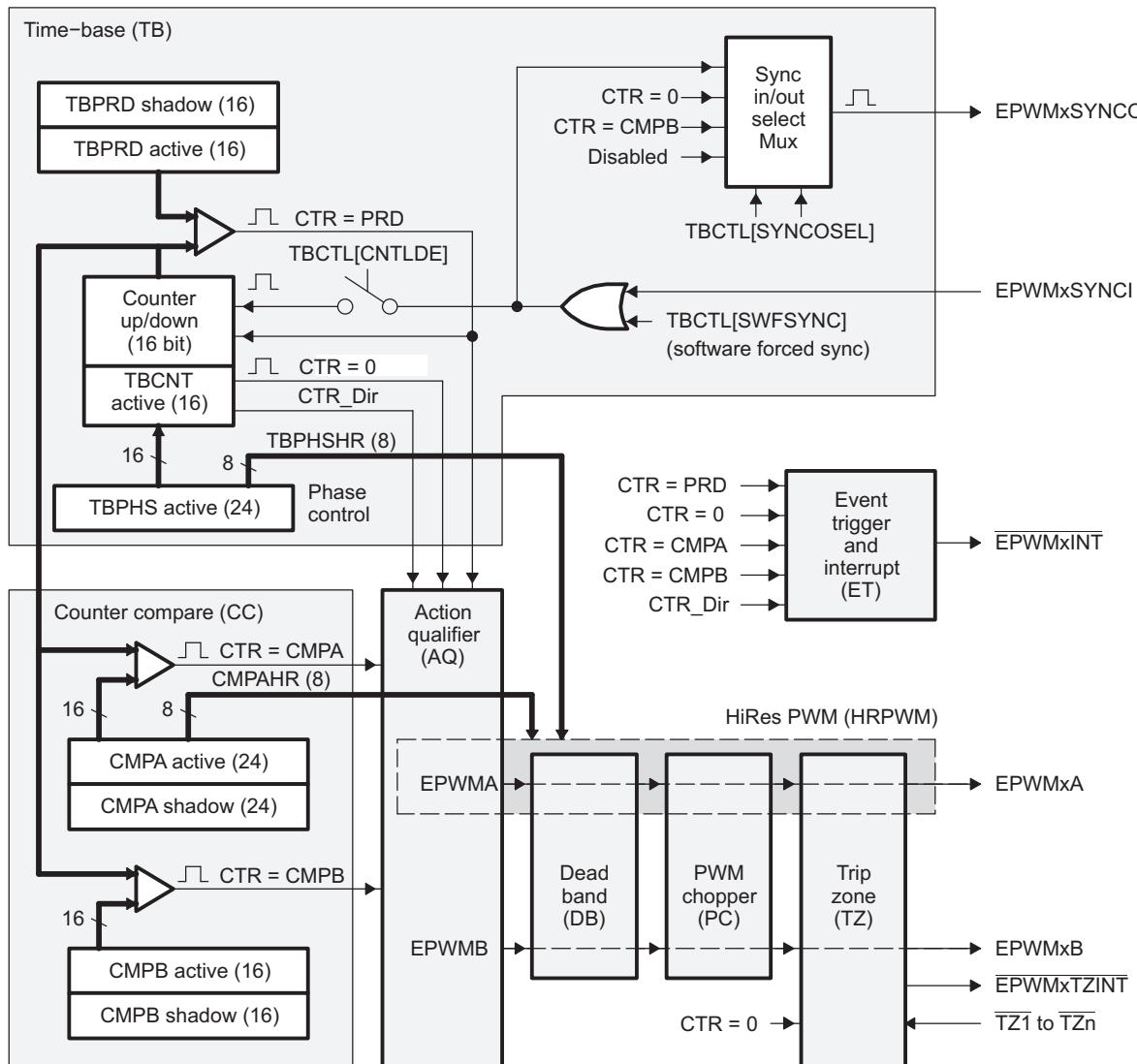
**Figure 15-7. Submodules and Signal Connections for an ePWM Module**

Figure 15-8 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- PWM output signals (EPWMxA and EPWMxB).** The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for your device.
- Trip-zone signals (TZ1 to TZn).** These input signals alert the ePWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signal can be configured as an asynchronous input through the GPIO peripheral. See [Section 15.1.2](#) to determine how many trip-zone pins are available in the device.
- Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals.** The synchronization signals daisy chain the ePWM modules together. Each module can be configured to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The synchronization output for ePWM1 (EPWM1SYNCO) is also connected to the SYNCI of the first enhanced capture module (eCAP1).
- Peripheral Bus.** The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

Figure 15-8 also shows the key internal submodule interconnect signals. Each submodule is described in [Section 15.2.2](#).

**Figure 15-8. ePWM Submodules and Critical Internal Signal Interconnects**



## 15.2.2 Functional Description

Seven submodules are included in every ePWM peripheral. There are some instances that include a high-resolution submodule that allows more precise control of the PWM outputs. Each of these submodules performs specific tasks that can be configured by software.

### 15.2.2.1 Overview

[Table 15-10](#) lists the eight key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in [Section 15.2.2.4](#) for relevant details.

**Table 15-10. Submodule Configuration Parameters**

| Submodule             | Configuration Parameter or Option  |
|-----------------------|--|
| Time-base (TB)        | <ul style="list-style-type: none"> <li>• Scale the time-base clock (TBCLK) relative to the system clock (SYSCLKOUT).</li> <li>• Configure the PWM time-base counter (TBCNT) frequency or period.</li> <li>• Set the mode for the time-base counter:           <ul style="list-style-type: none"> <li>– count-up mode: used for asymmetric PWM</li> <li>– count-down mode: used for asymmetric PWM</li> <li>– count-up-and-down mode: used for symmetric PWM</li> </ul> </li> <li>• Configure the time-base phase relative to another ePWM module.</li> <li>• Synchronize the time-base counter between modules through hardware or software.</li> <li>• Configure the direction (up or down) of the time-base counter after a synchronization event.</li> <li>• Configure how the time-base counter will behave when the device is halted by an emulator.</li> <li>• Specify the source for the synchronization output of the ePWM module:           <ul style="list-style-type: none"> <li>– Synchronization input signal</li> <li>– Time-base counter equal to zero</li> <li>– Time-base counter equal to counter-compare B (CMPB)</li> <li>– No output synchronization signal generated.</li> </ul> </li> </ul> |
| Counter-compare (CC)  | <ul style="list-style-type: none"> <li>• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB</li> <li>• Specify the time at which switching events occur on the EPWMxA or EPWMxB output</li> </ul>   |
| Action-qualifier (AQ) | <ul style="list-style-type: none"> <li>• Specify the type of action taken when a time-base or counter-compare submodule event occurs:           <ul style="list-style-type: none"> <li>– No action taken</li> <li>– Output EPWMxA and/or EPWMxB switched high</li> <li>– Output EPWMxA and/or EPWMxB switched low</li> <li>– Output EPWMxA and/or EPWMxB toggled</li> </ul> </li> <li>• Force the PWM output state through software control</li> <li>• Configure and control the PWM dead-band through software</li> </ul>   |
| Dead-band (DB)        | <ul style="list-style-type: none"> <li>• Control of traditional complementary dead-band relationship between upper and lower switches</li> <li>• Specify the output rising-edge-delay value</li> <li>• Specify the output falling-edge delay value</li> <li>• Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>   |
| PWM-chopper (PC)      | <ul style="list-style-type: none"> <li>• Create a chopping (carrier) frequency.</li> <li>• Pulse width of the first pulse in the chopped pulse train.</li> <li>• Duty cycle of the second and subsequent pulses.</li> <li>• Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.</li> </ul>   |

**Table 15-10. Submodule Configuration Parameters (continued)**

| Submodule                   | Configuration Parameter or Option  |
|-----------------------------|--|
| Trip-zone (TZ)              | <ul style="list-style-type: none"> <li>• Configure the ePWM module to react to one, all, or none of the trip-zone pins.</li> <li>• Specify the tripping action taken when a fault occurs: <ul style="list-style-type: none"> <li>– Force EPWMxA and/or EPWMxB high</li> <li>– Force EPWMxA and/or EPWMxB low</li> <li>– Force EPWMxA and/or EPWMxB to a high-impedance state</li> <li>– Configure EPWMxA and/or EPWMxB to ignore any trip condition.</li> </ul> </li> <li>• Configure how often the ePWM will react to each trip-zone pin: <ul style="list-style-type: none"> <li>– One-shot</li> <li>– Cycle-by-cycle</li> </ul> </li> <li>• Enable the trip-zone to initiate an interrupt.</li> <li>• Bypass the trip-zone module entirely.</li> </ul> |
| Event-trigger (ET)          | <ul style="list-style-type: none"> <li>• Enable the ePWM events that will trigger an interrupt.</li> <li>• Specify the rate at which events cause triggers (every occurrence or every second or third occurrence)</li> <li>• Poll, set, or clear event flags</li> </ul>  |
| High-Resolution PWM (HRPWM) | <ul style="list-style-type: none"> <li>• Enable extended time resolution capabilities</li> <li>• Configure finer time granularity control or edge positioning</li> </ul>   |

Code examples are provided in the remainder of this chapter that show how to implement various ePWM module configurations. These examples use the constant definitions shown in [Example 15-1](#).

### **Example 15-1. Constant Definitions Used in the Code Examples**

```
// TBCTL (Time-Base Control)
// =====
// TBCNT MODE bits
#define TB_COUNT_UP      0x0
#define TB_COUNT_DOWN    0x1
#define TB_COUNT_UPDOWN  0x2
#define TB_FREEZE        0x3
// PHSEN bit
#define TB_DISABLE        0x0
#define TB_ENABLE         0x1
// PRDLD bit
#define TB_SHADOW         0x0
#define TB_IMMEDIATE     0x1
// SYNCSEL bits
#define TB_SYNC_IN        0x0
#define TB_CTR_ZERO       0x1
#define TB_CTR_CMPB      0x2
#define TB_SYNC_DISABLE   0x3
// HSPCLKDIV and CLKDIV bits
#define TB_DIV1           0x0
#define TB_DIV2           0x1
#define TB_DIV4           0x2
// PHSDIR bit
#define TB_DOWN           0x0
#define TB_UP             0x1
// CMPCTL (Compare Control)
// =====
// LOADAMODE and LOADBMODE bits
#define CC_CTR_ZERO       0x0
#define CC_CTR_PRD        0x1
#define CC_CTR_ZERO_PRD   0x2
#define CC_LD_DISABLE     0x3
```

**Example 15-1. Constant Definitions Used in the Code Examples (continued)**

```

// SHDWAMODE and SHDWBMODE bits
#define CC_SHADOW 0x0
#define CC_IMMEDIATE 0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// = = = = = = = = = = = = = = = = = = = = = = = =
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define AQ_NO_ACTION 0x0
#define AQ_CLEAR 0x1
#define AQ_SET 0x2
#define AQ_TOGGLE 0x3
// DBCTL (Dead-Band Control)
// = = = = = = = = = = = = = = = = = =
// MODE bits
#define DB_DISABLE 0x0
#define DBA_ENABLE 0x1
#define DBB_ENABLE 0x2
#define DB_FULL_ENABLE 0x3
// POLSEL bits
#define DB_ACTV_HI 0x0
#define DB_ACTV_LOC 0x1
#define DB_ACTV_HIC 0x2
#define DB_ACTV_LO 0x3
// PCCTL (chopper control)
// = = = = = = = = = = = = = = = = = =
// CHPEN bit
#define CHP_ENABLE 0x0
#define CHP_DISABLE 0x1
// CHPFREQ bits
#define CHP_DIV1 0x0
#define CHP_DIV2 0x1
#define CHP_DIV3 0x2
#define CHP_DIV4 0x3
#define CHP_DIV5 0x4
#define CHP_DIV6 0x5
#define CHP_DIV7 0x6
#define CHP_DIV8 0x7
// CHPDUTY bits
#define CHP1_8TH 0x0
#define CHP2_8TH 0x1
#define CHP3_8TH 0x2
#define CHP4_8TH 0x3
#define CHP5_8TH 0x4
#define CHP6_8TH 0x5
#define CHP7_8TH 0x6
// TZSEL (Trip-zone Select)
// = = = = = = = = = = = = = = = = = =
// CBCn and OSHTn bits
#define TZ_ENABLE 0x0
#define TZ_DISABLE 0x1
// TZCTL (Trip-zone Control)
// = = = = = = = = = = = = = = = = = =
// TZA and TZB bits
#define TZ_HIZ 0x0
#define TZ_FORCE_HI 0x1
#define TZ_FORCE_LO 0x2
#define TZ_DISABLE 0x3
// ETSEL (Event-trigger Select)
// = = = = = = = = = = = = = = = = = =
// INTSEL, SOCASEL, SOCBSEL bits
#define ET_CTR_ZERO 0x1
#define ET_CTR_PRD 0x2
#define ET_CTRU_CMPA 0x4
#define ET_CTRD_CMPA 0x5
#define ET_CTRU_CMPP 0x6

```

**Example 15-1. Constant Definitions Used in the Code Examples (continued)**

```
#define      ET_CTRD_CMPB      0x7
// ETPS (Event-trigger Prescale)
// = = = = = = = = = = = = = = = = = = = = = =
// INTPRD, SOCAPRD, SOCBPRD bits
#define      ET_DISABLE        0x0
#define      ET_1ST            0x1
#define      ET_2ND            0x2
#define      ET_3RD            0x3
```

**15.2.2.2 Proper Interrupt Initialization Procedure**

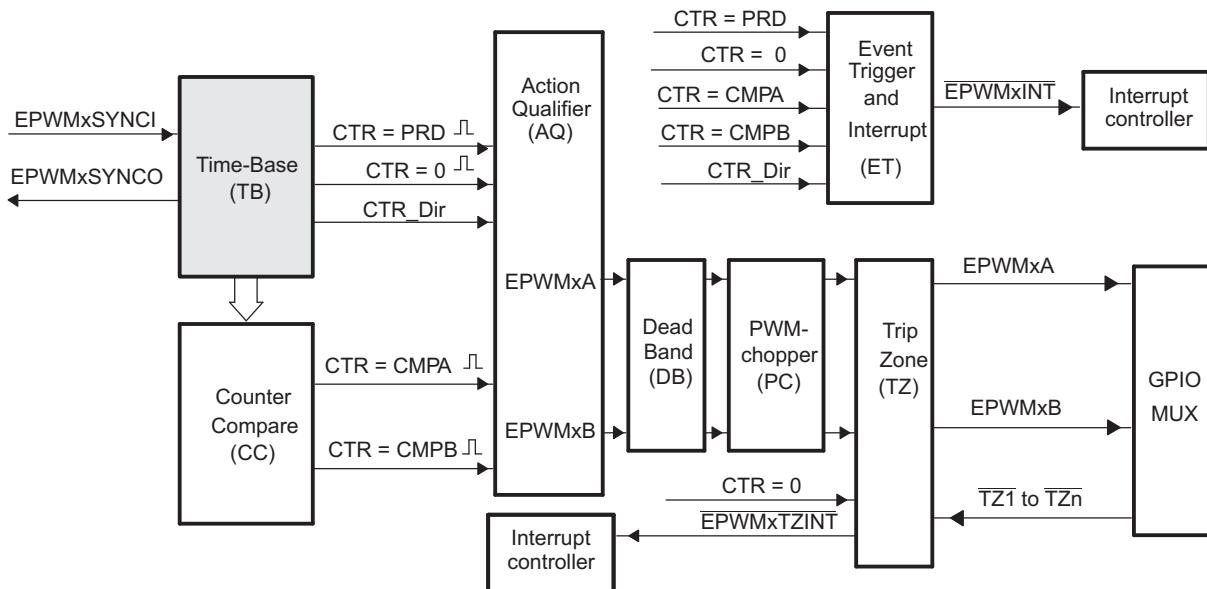
When the ePWM peripheral clock is enabled it may be possible that interrupt flags may be set due to spurious events due to the ePWM registers not being properly initialized. The proper procedure for initializing the ePWM peripheral is:

1. Disable global interrupts (CPU INTM flag)
2. Disable ePWM interrupts
3. Initialize peripheral registers
4. Clear any spurious ePWM flags
5. Enable ePWM interrupts
6. Enable global interrupts

**15.2.2.3 Time-Base (TB) Submodule**

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. [Figure 15-9](#) illustrates the time-base module's place within the ePWM.

**Figure 15-9. Time-Base Submodule Block Diagram**



### 15.2.2.3.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCNT) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCNT = TBPRD) .
  - CTR = 0: Time-base counter equal to zero (TBCNT = 0000h).
- Configure the rate of the time-base clock; a prescaled version of the CPU system clock (SYSCLKOUT). This allows the time-base counter to increment/decrement at a slower rate.

### 15.2.2.3.2 Controlling and Monitoring the Time-Base Submodule

Table 15-11 lists the registers used to control and monitor the time-base submodule.

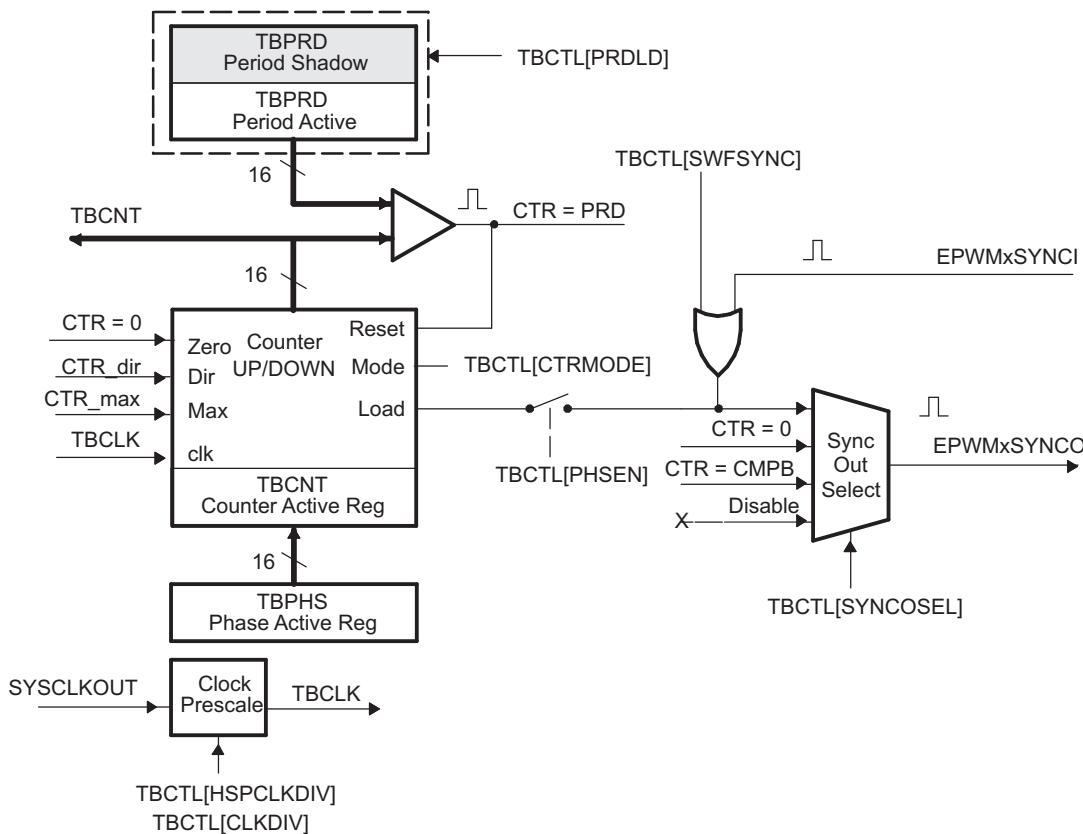
**Table 15-11. Time-Base Submodule Registers**

| Acronym | Register Description                          | Address Offset | Shadowed |
|---------|---|----------------|----------|
| TBCTL   | Time-Base Control Register                    | 0h             | No       |
| TBSTS   | Time-Base Status Register                     | 2h             | No       |
| TBPHSR  | HRPWM extension Phase Register <sup>(1)</sup> | 4h             | No       |
| TBPHS   | Time-Base Phase Register                      | 6h             | No       |
| TBCNT   | Time-Base Counter Register                    | 8h             | No       |
| TBPRD   | Time-Base Period Register                     | Ah             | Yes      |

<sup>(1)</sup> This register is available only on ePWM instances that include the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM, this location is reserved. See [Section 15.1.2](#) to determine which ePWM instances include this feature.

Figure 15-10 shows the critical signals and registers of the time-base submodule. Table 15-12 provides descriptions of the key signals associated with the time-base submodule.

**Figure 15-10. Time-Base Submodule Signals and Registers**



**Table 15-12. Key Time-Base Signals**

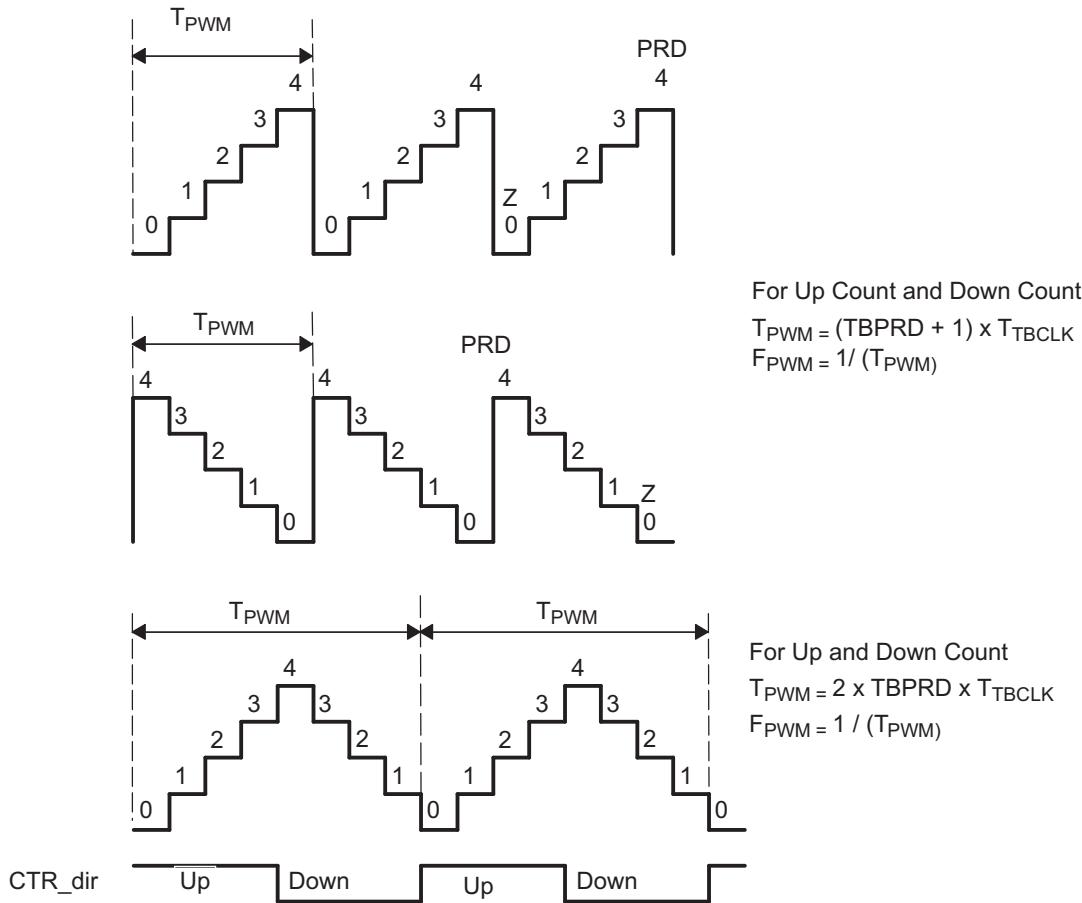
| Signal     | Description  |
|------------|--|
| EPWMxSYNCI | Time-base synchronization input.<br><br>Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module (EPWM1) this signal comes from a device pin. For subsequent ePWM modules this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. See <a href="#">Section 15.2.2.3.3.2</a> for information on the synchronization order of a particular device. |
| EPWMxSYNCO | Time-base synchronization output.<br><br>This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources:<br>1. EPWMxSYNCI (Synchronization input pulse)<br>2. CTR = 0: The time-base counter equal to zero (TBCNT = 0000h).<br>3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCNT = CMPB) register.  |
| CTR = PRD  | Time-base counter equal to the specified period.<br><br>This signal is generated whenever the counter value is equal to the active period register value. That is when TBCNT = TBPRD.  |
| CTR = 0    | Time-base counter equal to zero.<br><br>This signal is generated whenever the counter value is zero. That is when TBCNT equals 0000h.  |
| CTR = CMPB | Time-base counter equal to active counter-compare B register (TBCNT = CMPB).<br><br>This event is generated by the counter-compare submodule and used by the synchronization out logic.  |
| CTR_dir    | Time-base counter direction.<br><br>Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.   |
| CTR_max    | Time-base counter equal max value. (TBCNT = FFFFh)<br><br>Generated event when the TBCNT value reaches its maximum value. This signal is only used only as a status bit.   |
| TBCLK      | Time-base clock.<br><br>This is a prescaled version of the system clock (SYSCLKOUT) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.   |

### 15.2.2.3.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. [Figure 15-11](#) shows the period ( $T_{\text{pwm}}$ ) and frequency ( $F_{\text{pwm}}$ ) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (SYSCLKOUT).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:** In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.
- **Down-Count Mode:** In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

**Figure 15-11. Time-Base Frequency and Period**


#### 15.2.2.3.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register:** The active register controls the hardware and is responsible for actions that the hardware causes or invokes.
- **Shadow Register:** The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

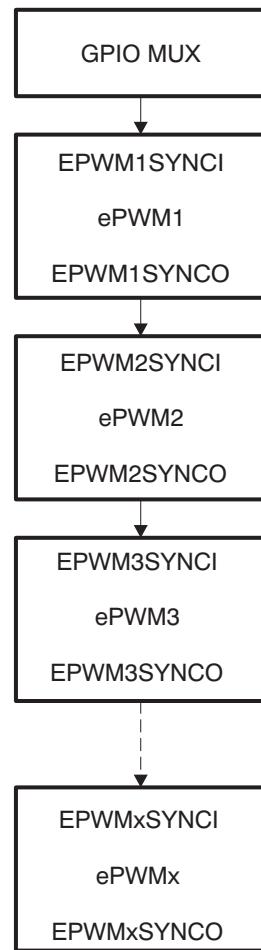
The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDLD] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:** The TBPRD shadow register is enabled when TBCTL[PRDLD] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active)  $\leftarrow$  TBPRD (shadow)) when the time-base counter equals zero (TBCNT = 0000h). By default the TBPRD shadow register is enabled.
- **Time-Base Period Immediate Load Mode:** If immediate load mode is selected (TBCTL[PRDLD] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

### 15.2.2.3.3.2 Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the ePWM modules on a device. Each ePWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (ePWM1) comes from an external pin. The possible synchronization connections for the remaining ePWM modules is shown in [Figure 15-12](#).

**Figure 15-12. Time-Base Counter Synchronization Scheme 1**



Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCNT) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:** The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.
- **Software Forced Synchronization Pulse:** Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The TBPHS bit is ignored in count-up or count-down modes. See [Figure 15-13](#) through [Figure 15-16](#) for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master.

#### **15.2.2.3.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules**

The TBCLKEN bit in the PWMSS\_CTRL register in the Control Module can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. The TBCLKEN bit is part of the chip configuration registers and is described in [Chapter 9](#). When TBCLKEN = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKEN = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the ePWM module clocks.
2. Set TBCLKEN = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKEN = 1.

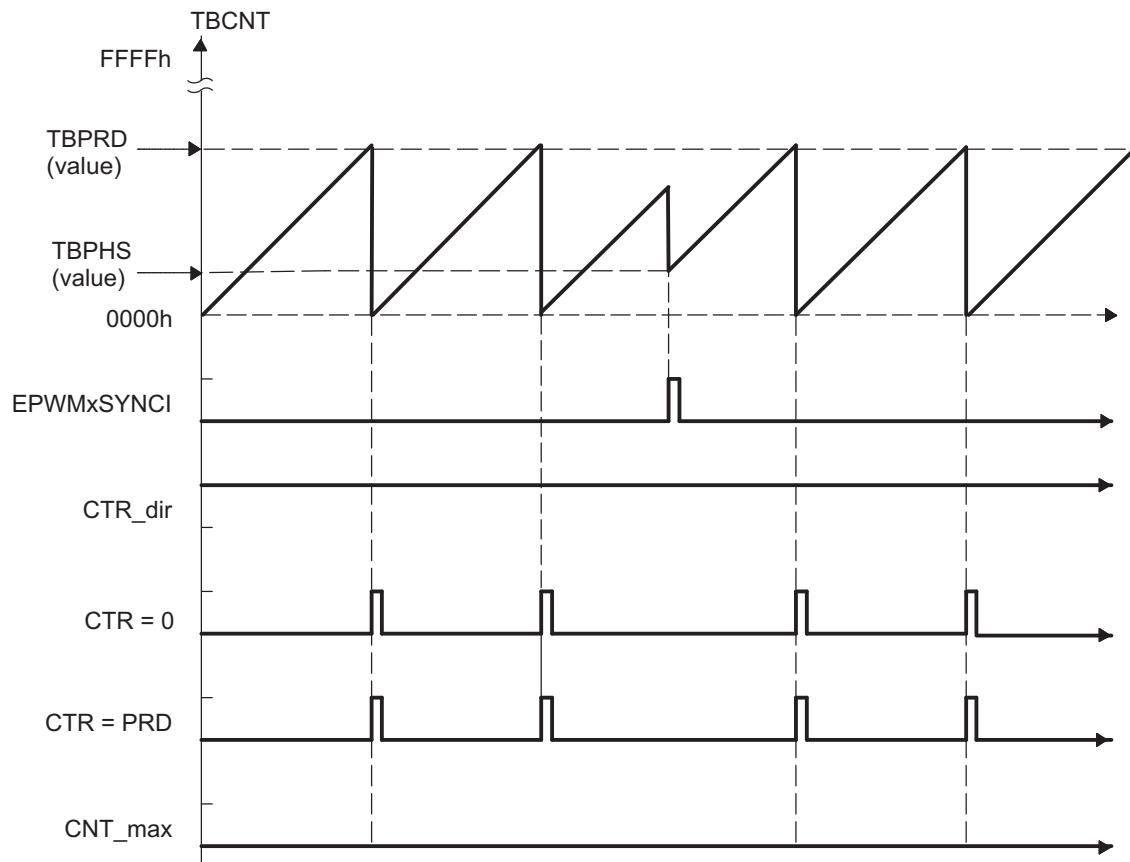
#### **15.2.2.3.5 Time-Base Counter Modes and Timing Waveforms**

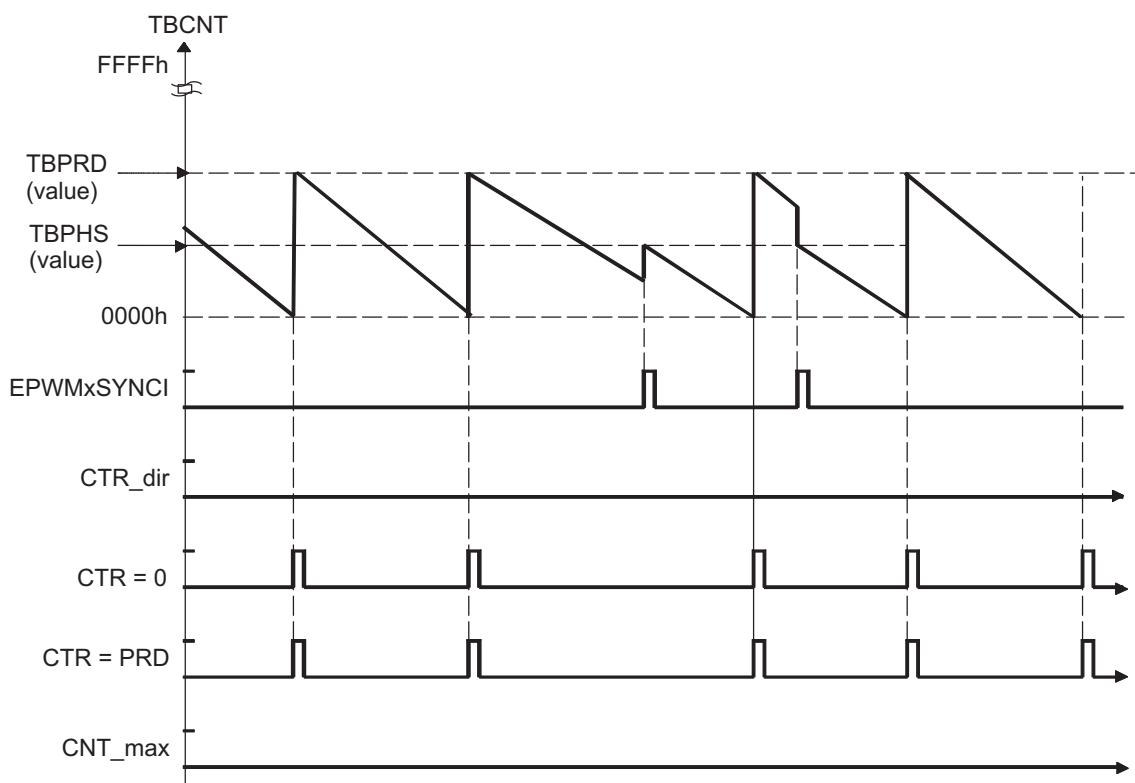
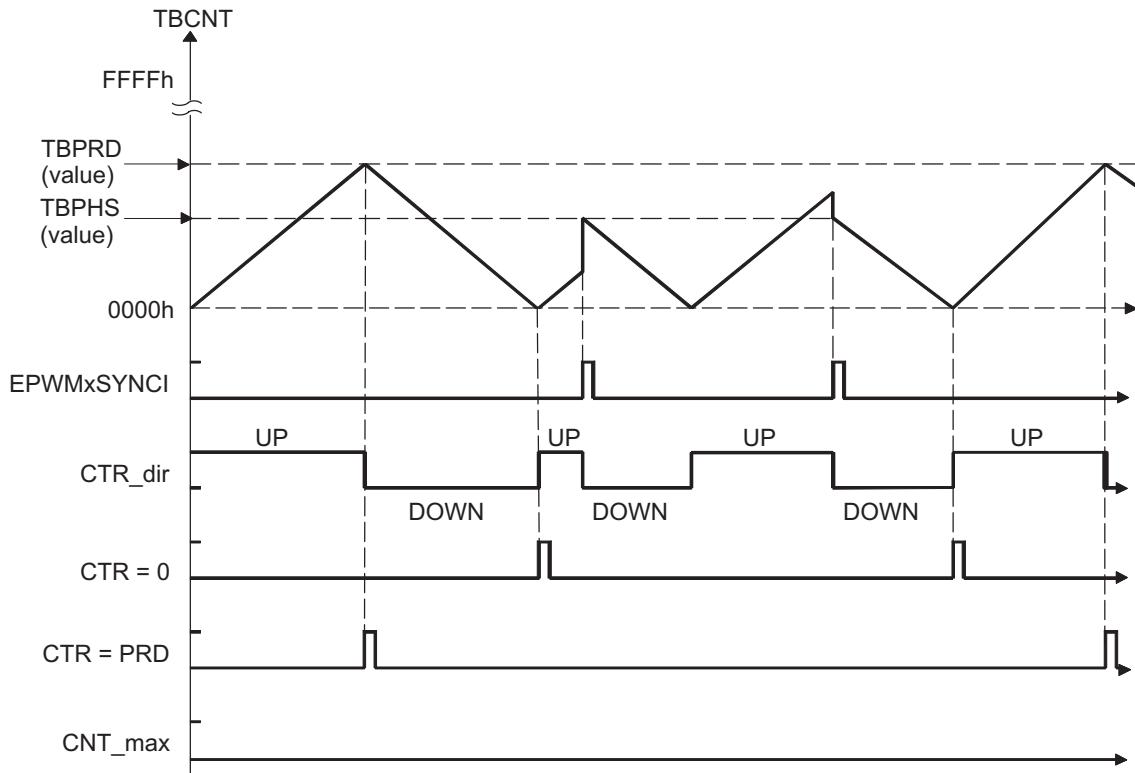
The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical.
- Frozen where the time-base counter is held constant at the current value.

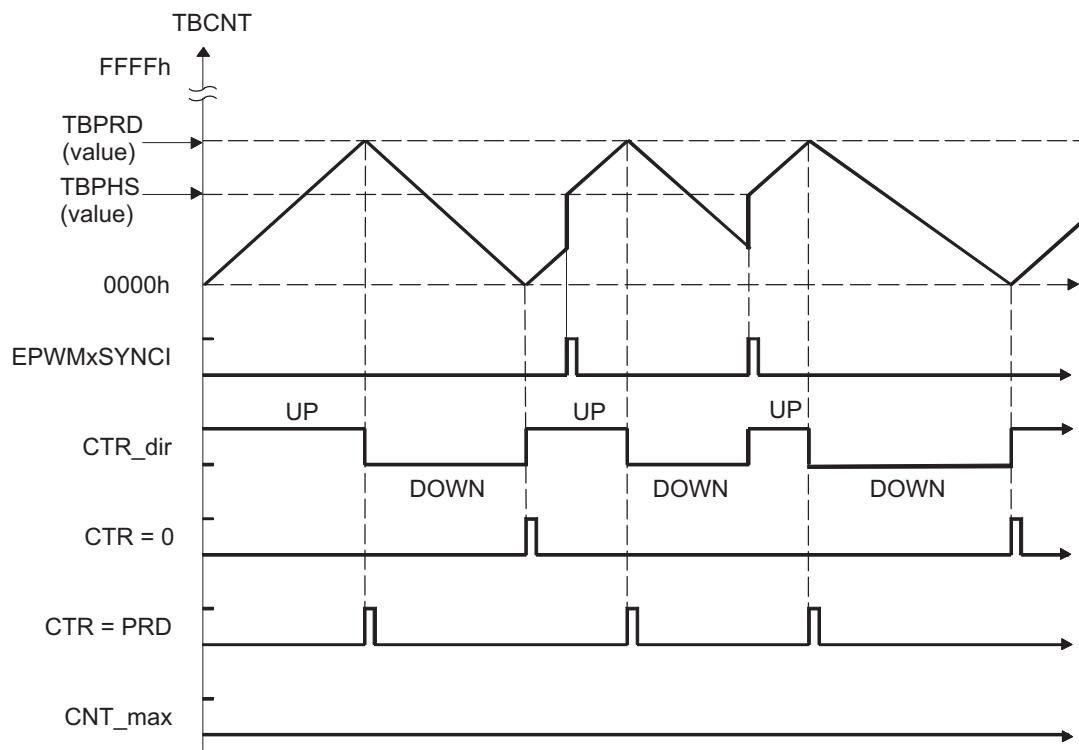
To illustrate the operation of the first three modes, [Figure 15-13](#) to [Figure 15-16](#) show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

Figure 15-13. Time-Base Up-Count Mode Waveforms



**Figure 15-14. Time-Base Down-Count Mode Waveforms**

**Figure 15-15. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down on Synchronization Event**


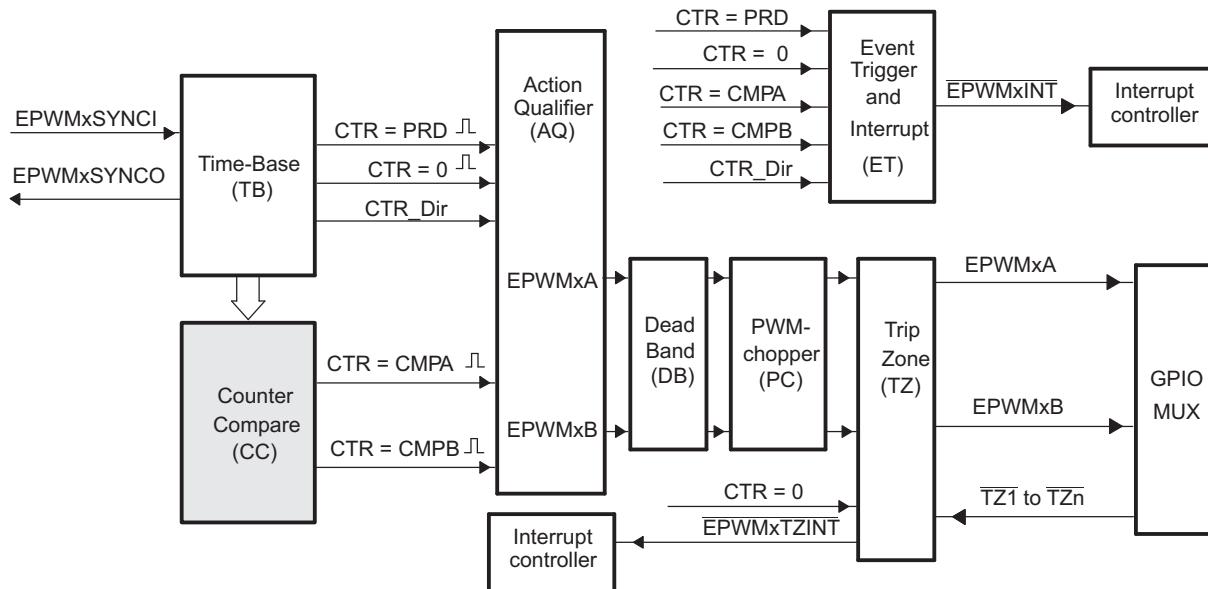
**Figure 15-16. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up on Synchronization Event**



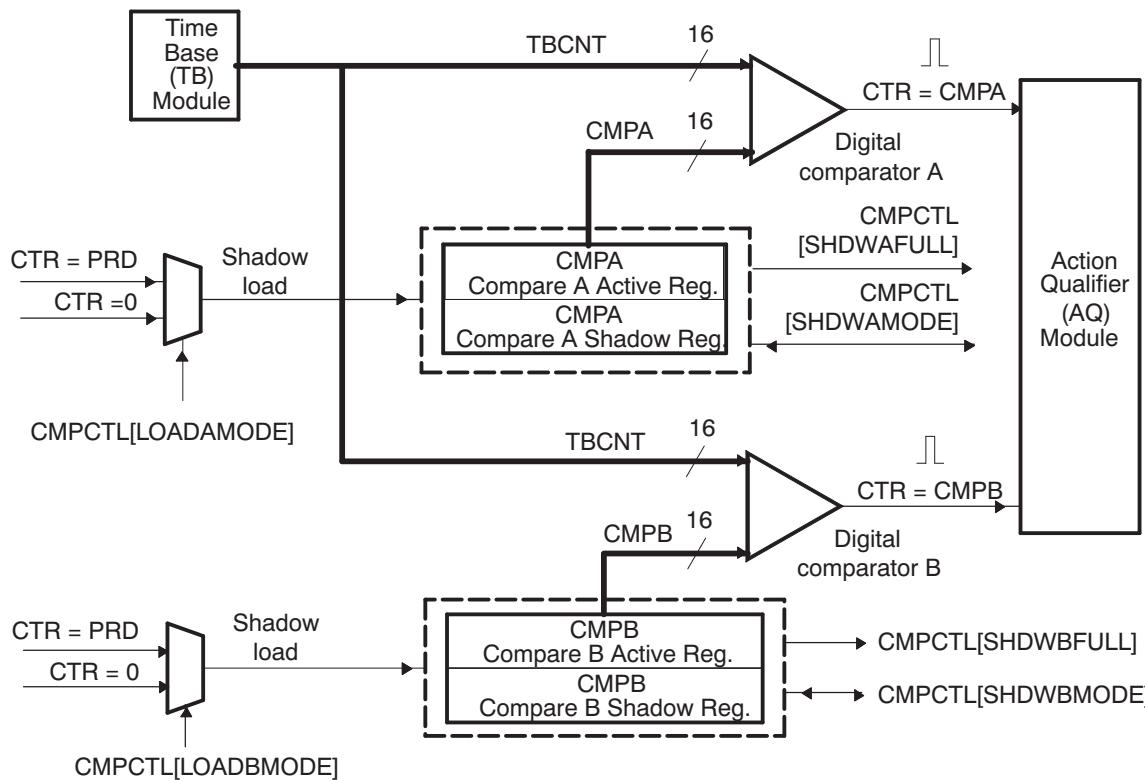
#### 15.2.2.4 Counter-Compare (CC) Submodule

Figure 15-17 illustrates the counter-compare submodule within the ePWM. Figure 15-18 shows the basic structure of the counter-compare submodule.

**Figure 15-17. Counter-Compare Submodule**



**Figure 15-18. Counter-Compare Submodule Signals and Registers**



#### 15.2.2.4.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) and counter-compare B (CMPB) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

The counter-compare submodule:

- Generates events based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCNT = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCNT = CMPB)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

#### 15.2.2.4.2 Controlling and Monitoring the Counter-Compare Submodule

[Table 15-13](#) lists the registers used to control and monitor the counter-compare submodule. [Table 15-14](#) lists the key signals associated with the counter-compare submodule.

**Table 15-13. Counter-Compare Submodule Registers**

| Acronym | Register Description                                      | Address Offset | Shadowed |
|---------|---|----------------|----------|
| CMPCTL  | Counter-Compare Control Register.                         | Eh             | No       |
| CMPAHR  | HRPWM Counter-Compare A Extension Register <sup>(1)</sup> | 10h            | Yes      |
| CMPA    | Counter-Compare A Register                                | 12h            | Yes      |
| CMPB    | Counter-Compare B Register                                | 14h            | Yes      |

<sup>(1)</sup> This register is available only on ePWM modules with the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM, this location is reserved. See [Section 15.1.2](#) to determine which ePWM instances include this feature.

**Table 15-14. Counter-Compare Submodule Key Signals**

| Signal     | Description of Event  | Registers Compared |
|------------|---|--------------------|
| CTR = CMPA | Time-base counter equal to the active counter-compare A value   | TBCNT = CMPA       |
| CTR = CMPB | Time-base counter equal to the active counter-compare B value   | TBCNT = CMPB       |
| CTR = PRD  | Time-base counter equal to the active period.<br>Used to load active counter-compare A and B registers from the shadow register | TBCNT = TBPRD      |
| CTR = 0    | Time-base counter equal to zero.<br>Used to load active counter-compare A and B registers from the shadow register              | TBCNT = 0000h      |

#### **15.2.2.4.3 Operational Highlights for the Counter-Compare Submodule**

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCNT = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCNT = CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle, if the compare value is between 0000h and TBPRD; and occurs once per cycle, if the compare value is equal to 0000h or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to [Section 15.2.2.5.1](#) for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occurs at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

- **Shadow Mode:** The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events:

- CTR = PRD: Time-base counter equal to the period (TBCNT = TBPRD).
- CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)
- Both CTR = PRD and CTR = 0

Which of these three events is specified by the CMPCTL[LOADAMODE] and CMPCTL[LOADBMODE] register bits. Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

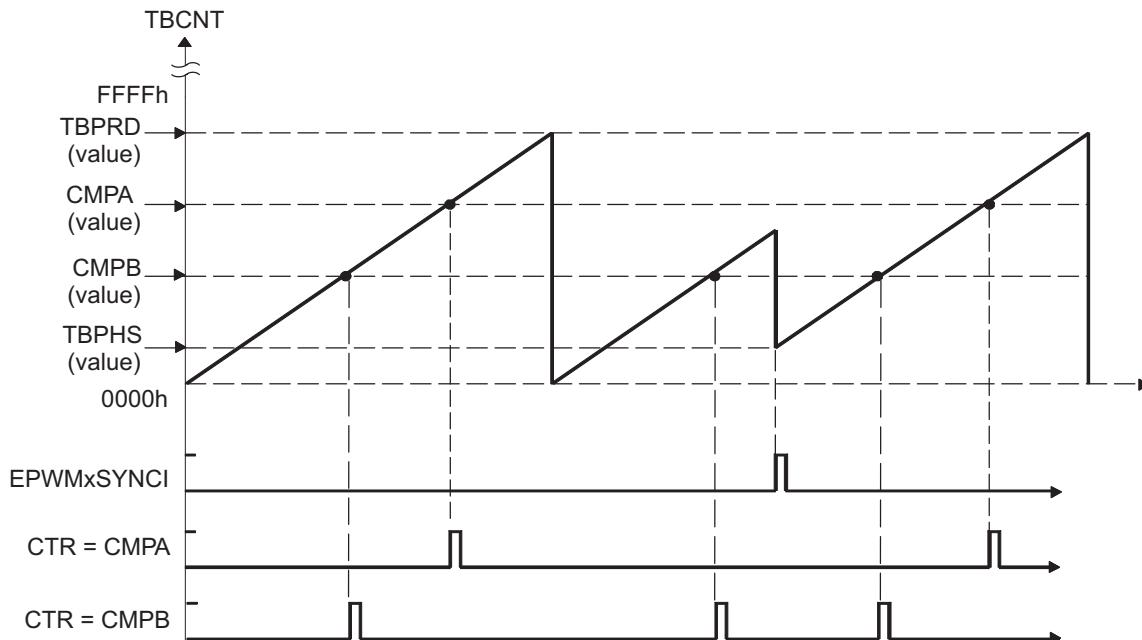
- **Immediate Load Mode:** If immediate load mode is selected (TBCTL[SHADWAMODE] = 1 or TBCTL[SHADWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

#### **15.2.2.4.4 Count Mode Timing Waveforms**

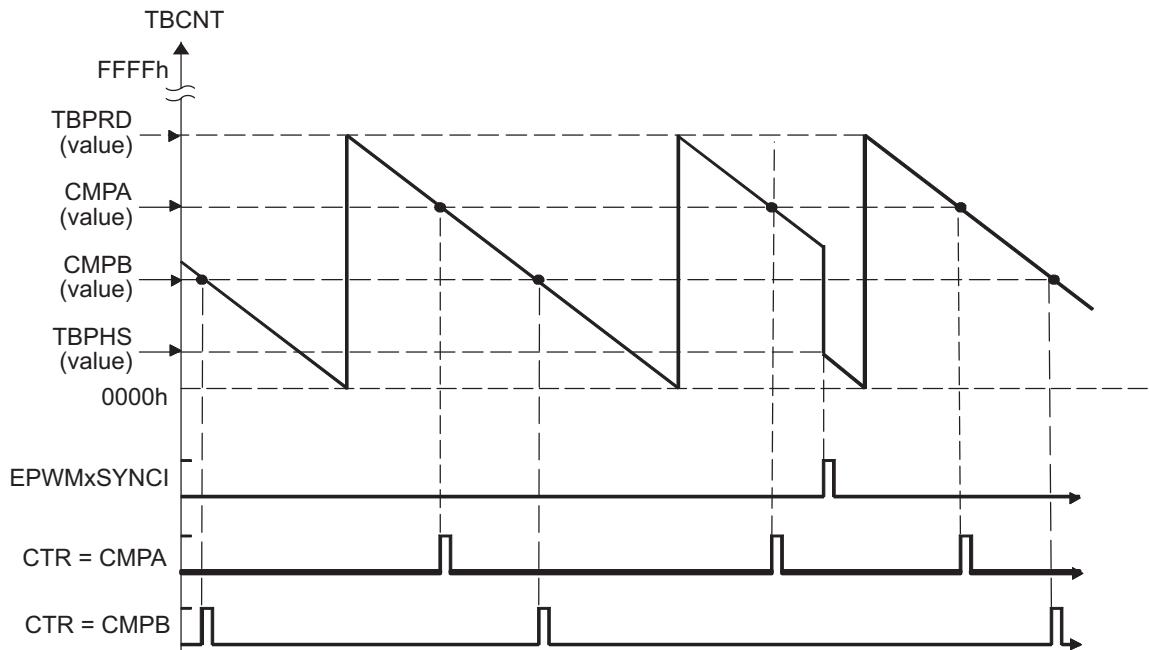
The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

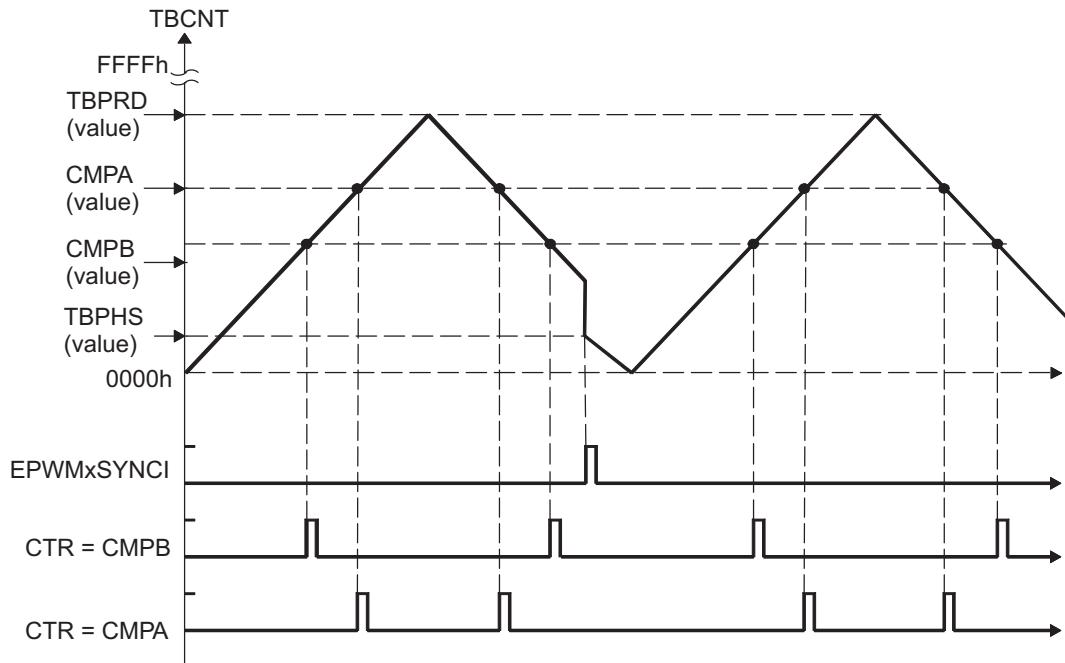
To best illustrate the operation of the first three modes, the timing diagrams in [Figure 15-19](#) to [Figure 15-22](#) show when events are generated and how the EPWMxSYNCI signal interacts.

**Figure 15-19. Counter-Compare Event Waveforms in Up-Count Mode**

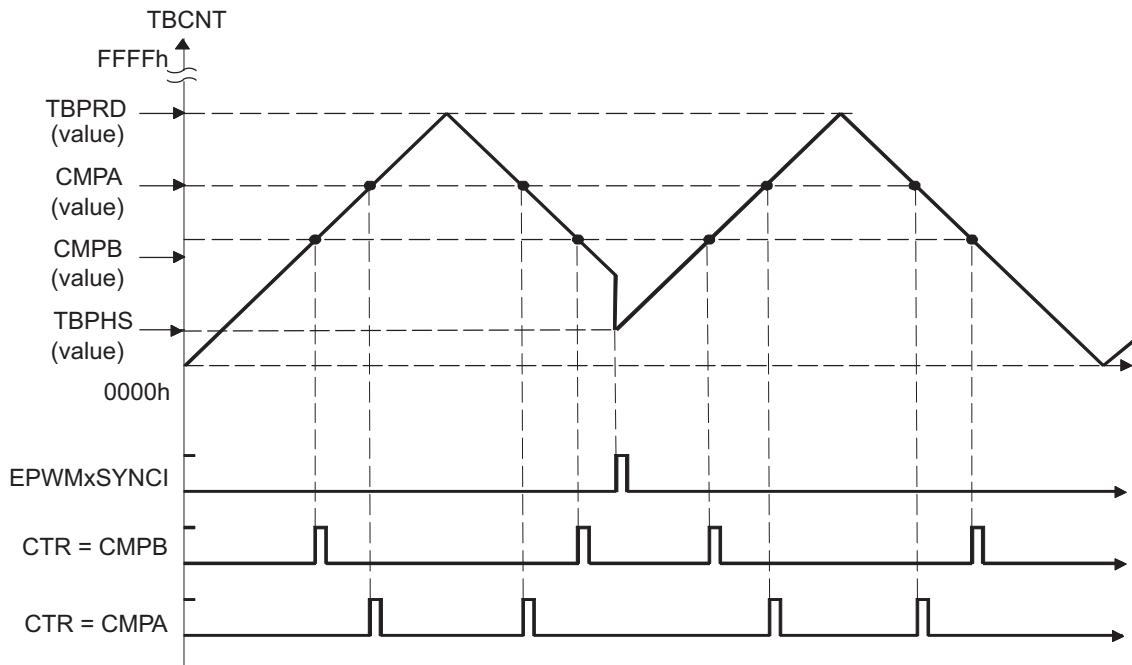
NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCNT count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

**Figure 15-20. Counter-Compare Events in Down-Count Mode**

**Figure 15-21. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down on Synchronization Event**



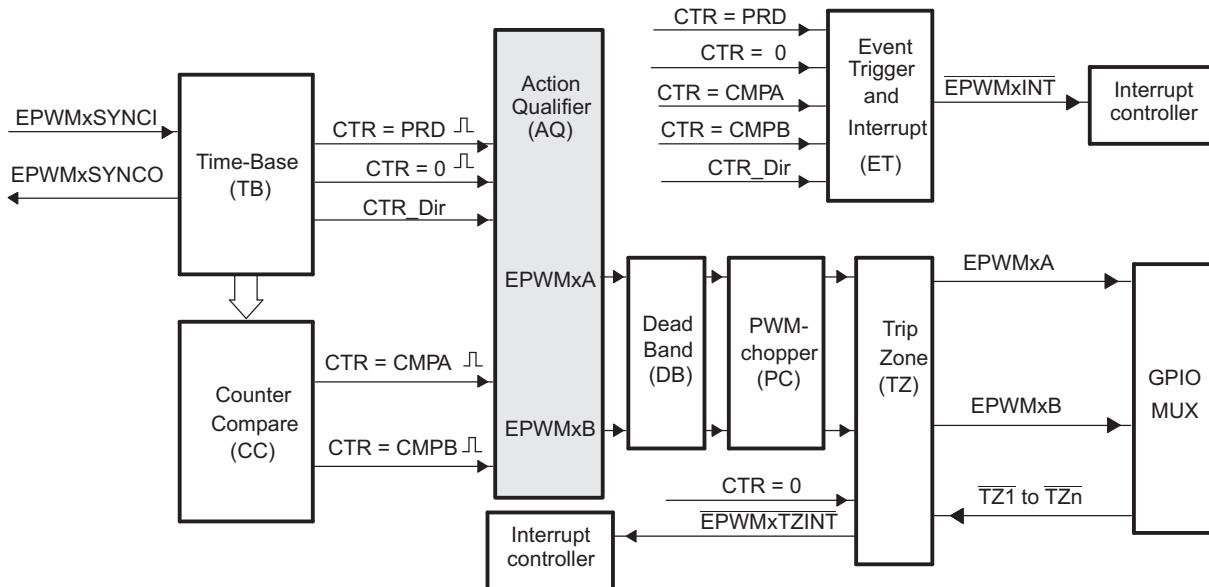
**Figure 15-22. Counter-Compare Events in Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up on Synchronization Event**



### 15.2.2.5 Action-Qualifier (AQ) Submodule

Figure 15-23 shows the action-qualifier (AQ) submodule (see shaded block) in the ePWM system. The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

**Figure 15-23. Action-Qualifier Submodule**



#### 15.2.2.5.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period ( $TBCNT = TBPRD$ )
  - CTR = 0: Time-base counter equal to zero ( $TBCNT = 0000h$ )
  - CTR = CMPA: Time-base counter equal to the counter-compare A register ( $TBCNT = CMPA$ )
  - CTR = CMPB: Time-base counter equal to the counter-compare B register ( $TBCNT = CMPB$ )
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing.

#### 15.2.2.5.2 Controlling and Monitoring the Action-Qualifier Submodule

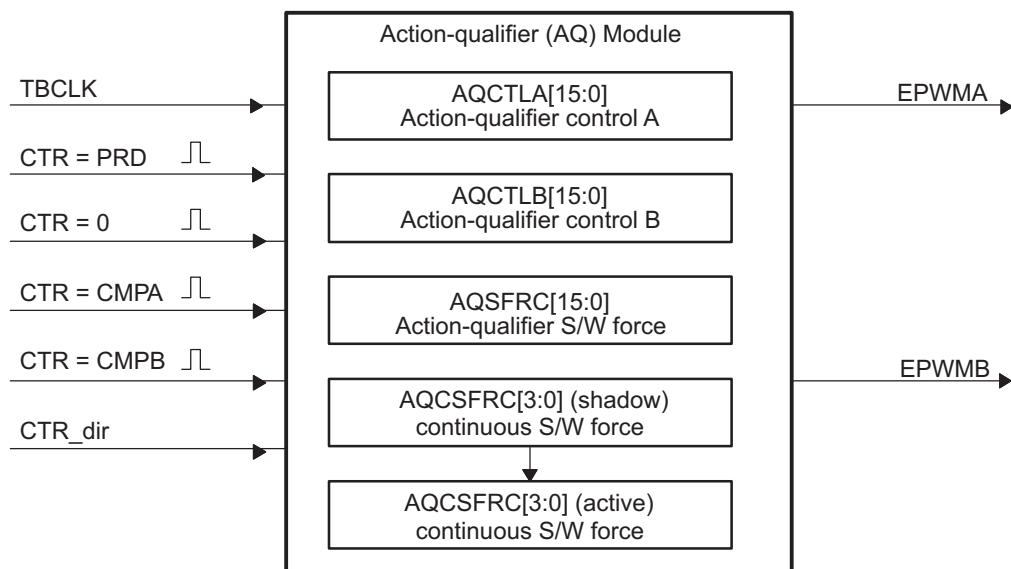
Table 15-15 lists the registers used to control and monitor the action-qualifier submodule.

**Table 15-15. Action-Qualifier Submodule Registers**

| Acronym | Register Description                                    | Address Offset | Shadowed |
|---------|---|----------------|----------|
| AQCTLA  | Action-Qualifier Control Register For Output A (EPWMxA) | 16h            | No       |
| AQCTLB  | Action-Qualifier Control Register For Output B (EPWMxB) | 18h            | No       |
| AQSFR   | Action-Qualifier Software Force Register                | 1Ah            | No       |
| AQCSFR  | Action-Qualifier Continuous Software Force              | 1Ch            | Yes      |

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers shown in [Figure 15-24](#). The possible input events are summarized again in [Table 15-16](#).

**Figure 15-24. Action-Qualifier Submodule Inputs and Outputs**



**Table 15-16. Action-Qualifier Submodule Possible Input Events**

| Signal                | Description                                      | Registers Compared |
|-----------------------|--|--------------------|
| CTR = PRD             | Time-base counter equal to the period value      | TBCNT = TBPRD      |
| CTR = 0               | Time-base counter equal to zero                  | TBCNT = 0000h      |
| CTR = CMPA            | Time-base counter equal to the counter-compare A | TBCNT = CMPA       |
| CTR = CMPB            | Time-base counter equal to the counter-compare B | TBCNT = CMPB       |
| Software forced event | Asynchronous event initiated by software         |                    |

The software forced action is a useful asynchronous event. This control is handled by registers AQSFRC and AQCSFRC.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:** Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:** Set output EPWMxA or EPWMxB to a low level.
- **Toggle:** If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:** Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts. See the event-trigger submodule description in [Section 15.2.2.9](#) for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this chapter use a set of symbolic actions. These symbols are summarized in [Figure 15-25](#). Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

**Figure 15-25. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

| S/W force | TB Counter equals: |        |        |        | Actions    |
|-----------|--------------------|--------|--------|--------|------------|
|           | Zero               | Comp A | Comp B | Period |            |
|           |                    |        |        |        | Do Nothing |
|           |                    |        |        |        | Clear Low  |
|           |                    |        |        |        | Set High   |
|           |                    |        |        |        | Toggle     |

### 15.2.2.5.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in [Table 15-17](#). A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCNT.

**Table 15-17. Action-Qualifier Event Priority for Up-Down-Count Mode**

| Priority Level | Event if TBCNT is Incrementing<br>TBCNT = 0 up to TBCNT = TBPRD | Event if TBCNT is Decrementing<br>TBCNT = TBPRD down to TBCNT = 1 |
|----------------|---|---|
| 1 (Highest)    | Software forced event   | Software forced event   |
| 2              | Counter equals CMPB on up-count (CBU)                           | Counter equals CMPB on down-count (CBD)                           |
| 3              | Counter equals CMPA on up-count (CAU)                           | Counter equals CMPA on down-count (CAD)                           |
| 4              | Counter equals zero   | Counter equals period (TBPRD)                                     |
| 5              | Counter equals CMPB on down-count (CBD) <sup>(1)</sup>          | Counter equals CMPB on up-count (CBU) <sup>(1)</sup>              |
| 6 (Lowest)     | Counter equals CMPA on down-count (CAD) <sup>(1)</sup>          | Counter equals CMPA on up-count (CBU) <sup>(1)</sup>              |

<sup>(1)</sup> To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

[Table 15-18](#) shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

**Table 15-18. Action-Qualifier Event Priority for Up-Count Mode**

| Priority Level | Event                                   |
|----------------|---|
| 1 (Highest)    | Software forced event                   |
| 2              | Counter equal to period (TBPRD)         |
| 3              | Counter equal to CMPB on up-count (CBU) |
| 4              | Counter equal to CMPA on up-count (CAU) |
| 5 (Lowest)     | Counter equal to Zero                   |

[Table 15-19](#) shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

**Table 15-19. Action-Qualifier Event Priority for Down-Count Mode**

| Priority Level | Event                                     |
|----------------|---|
| 1 (Highest)    | Software forced event                     |
| 2              | Counter equal to Zero                     |
| 3              | Counter equal to CMPB on down-count (CBD) |
| 4              | Counter equal to CMPA on down-count (CAD) |
| 5 (Lowest)     | Counter equal to period (TBPRD)           |

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 15-20.

**Table 15-20. Behavior if CMPA/CMPB is Greater than the Period**

| Counter Mode       | Compare on Up-Count Event CAU/CBU   | Compare on Down-Count Event CAU/CBU   |
|--------------------|---|---|
| Up-Count Mode      | If $\text{CMPA}/\text{CMPB} \leq \text{TBPRD}$ period, then the event occurs on a compare match ( $\text{TBCNT} = \text{CMPA}$ or $\text{CMPB}$ ).<br><br>If $\text{CMPA}/\text{CMPB} > \text{TBPRD}$ , then the event will not occur.  | Never occurs.   |
|                    |   |   |
| Down-Count Mode    | Never occurs.   | If $\text{CMPA}/\text{CMPB} < \text{TBPRD}$ , the event will occur on a compare match ( $\text{TBCNT} = \text{CMPA}$ or $\text{CMPB}$ ).<br><br>If $\text{CMPA}/\text{CMPB} \geq \text{TBPRD}$ , the event will occur on a period match ( $\text{TBCNT} = \text{TBPRD}$ ).                        |
|                    |   |   |
| Up-Down-Count Mode | If $\text{CMPA}/\text{CMPB} < \text{TBPRD}$ and the counter is incrementing, the event occurs on a compare match ( $\text{TBCNT} = \text{CMPA}$ or $\text{CMPB}$ ).<br><br>If $\text{CMPA}/\text{CMPB} \geq \text{TBPRD}$ , the event will occur on a period match ( $\text{TBCNT} = \text{TBPRD}$ ). | If $\text{CMPA}/\text{CMPB} < \text{TBPRD}$ and the counter is decrementing, the event occurs on a compare match ( $\text{TBCNT} = \text{CMPA}$ or $\text{CMPB}$ ).<br><br>If $\text{CMPA}/\text{CMPB} \geq \text{TBPRD}$ , the event occurs on a period match ( $\text{TBCNT} = \text{TBPRD}$ ). |
|                    |   |   |

#### 15.2.2.5.4 Waveforms for Common Configurations

**NOTE:** The waveforms in this chapter show the ePWMs behavior for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

**Use up-down-count mode to generate a symmetric PWM:**

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD - 1.

This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

**Use up-down-count mode to generate an asymmetric PWM:**

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

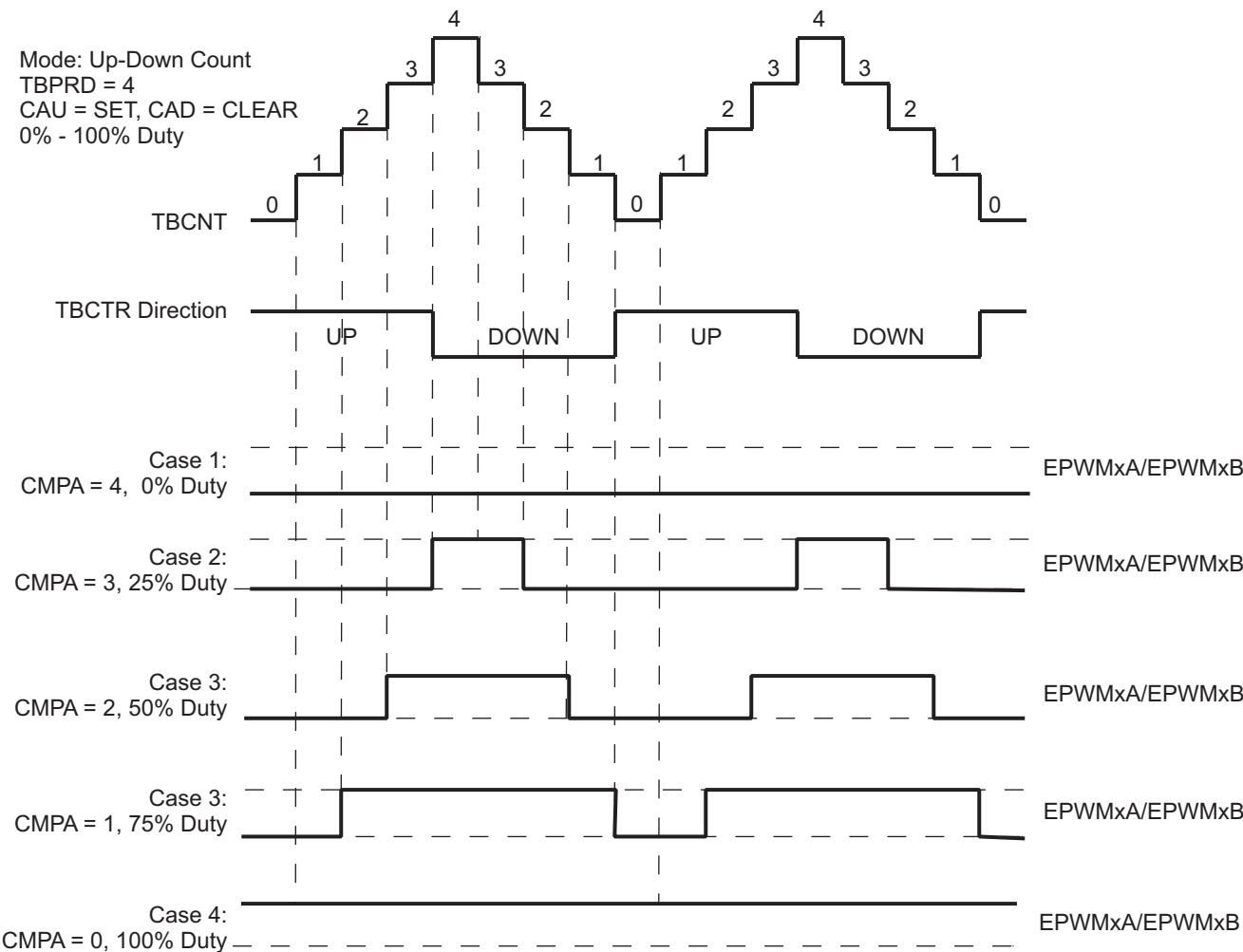
**When using up-count mode to generate an asymmetric PWM:**

- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

**Figure 15-26** shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCNT. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When  $CMPA = 0$ , the PWM signal is low for the entire period giving the 0% duty waveform. When  $CMPA = TBPRD$ , the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load  $CMPA/CMPB$  on zero, then use  $CMPA/CMPB$  values greater than or equal to 1. If you load  $CMPA/CMPB$  on period, then use  $CMPA/CMPB$  values less than or equal to  $TBPRD-1$ . This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

**Figure 15-26. Up-Down-Count Mode Symmetrical Waveform**

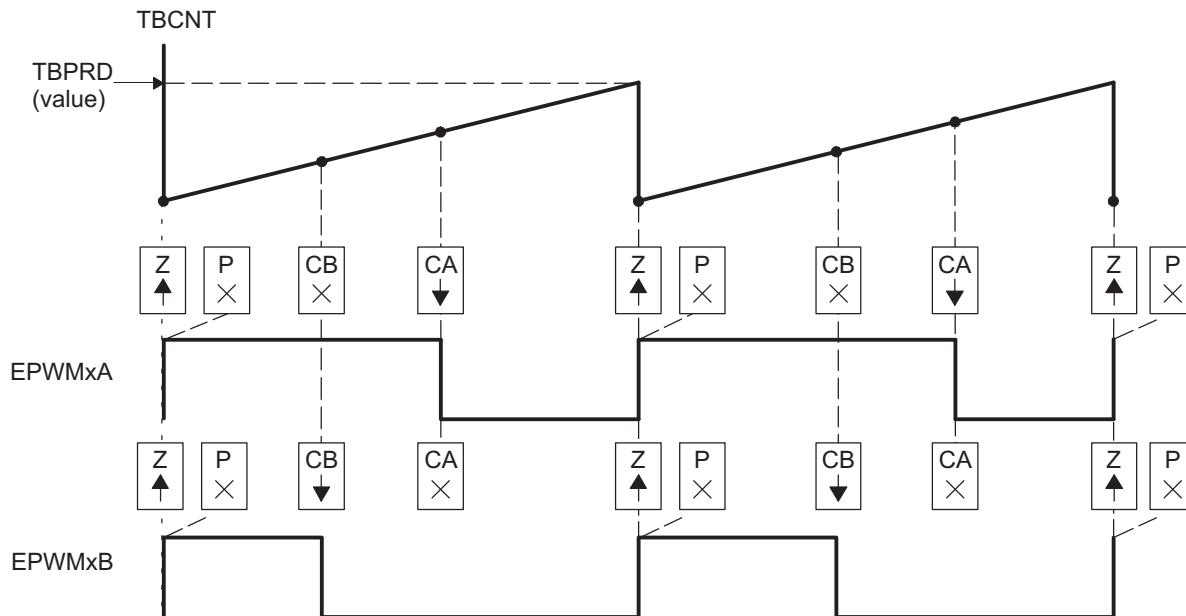


The PWM waveforms in [Figure 15-27](#) through [Figure 15-32](#) show some common action-qualifier configurations. Some conventions used in the figures are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

[Table 15-21](#) and [Table 15-22](#) contains initialization and runtime register configurations for the waveforms in [Figure 15-27](#).

**Figure 15-27. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High**



- (1) PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).
- (4) The "Do Nothing" actions ( X ) are shown for completeness, but will not be shown on subsequent diagrams.
- (5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

**Table 15-21. EPWMx Initialization for Figure 15-27**

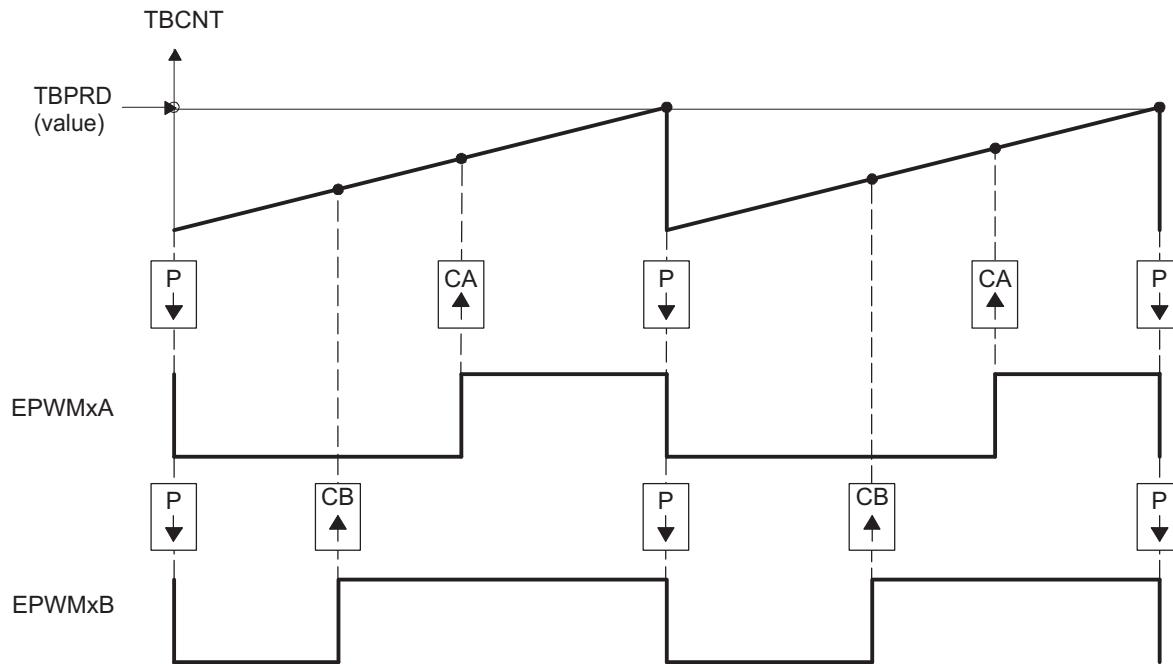
| <b>Register</b> | <b>Bit</b> | <b>Value</b>    | <b>Comments</b>              |
|-----------------|------------|-----------------|------------------------------|
| TBPRD           | TBPRD      | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS           | TBPHS      | 0               | Clear Phase Register to 0    |
| TBCNT           | TBCNT      | 0               | Clear TB counter             |
| TBCTL           | CTRMODE    | TB_UP           |                              |
|                 | PHSEN      | TB_DISABLE      | Phase loading disabled       |
|                 | PRDLD      | TB_SHADOW       |                              |
|                 | SYNCOSEL   | TB_SYNC_DISABLE |                              |
|                 | HSPCLKDIV  | TB_DIV1         | TBCLK = SYSCLK               |
|                 | CLKDIV     | TB_DIV1         |                              |
| CMPA            | CMPA       | 350 (15Eh)      | Compare A = 350 TBCLK counts |
| CMPB            | CMPB       | 200 (C8h)       | Compare B = 200 TBCLK counts |
| CMPCTL          | SHDWAMODE  | CC_SHADOW       |                              |
|                 | SHDWBMODE  | CC_SHADOW       |                              |
|                 | LOADAMODE  | CC_CTR_ZERO     | Load on CTR = 0              |
|                 | LOADBMODE  | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA          | ZRO        | AQ_SET          |                              |
|                 | CAU        | AQ_CLEAR        |                              |
| AQCTLB          | ZRO        | AQ_SET          |                              |
|                 | CBU        | AQ_CLEAR        |                              |

**Table 15-22. EPWMx Run Time Changes for Figure 15-27**

| <b>Register</b> | <b>Bit</b> | <b>Value</b> | <b>Comments</b>               |
|-----------------|------------|--------------|-------------------------------|
| CMPA            | CMPA       | Duty1A       | Adjust duty for output EPWM1A |
| CMPB            | CMPB       | Duty1B       | Adjust duty for output EPWM1B |

Table 15-23 and Table 15-24 contains initialization and runtime register configurations for the waveforms in Figure 15-28.

**Figure 15-28. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low**



- (1) PWM period =  $(TBPRD + 1) \times T_{TBCLK}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- (4) The Do Nothing actions ( X ) are shown for completeness here, but will not be shown on subsequent diagrams.
- (5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCNT wraps from period to 0000h.

**Table 15-23. EPWMx Initialization for Figure 15-28**

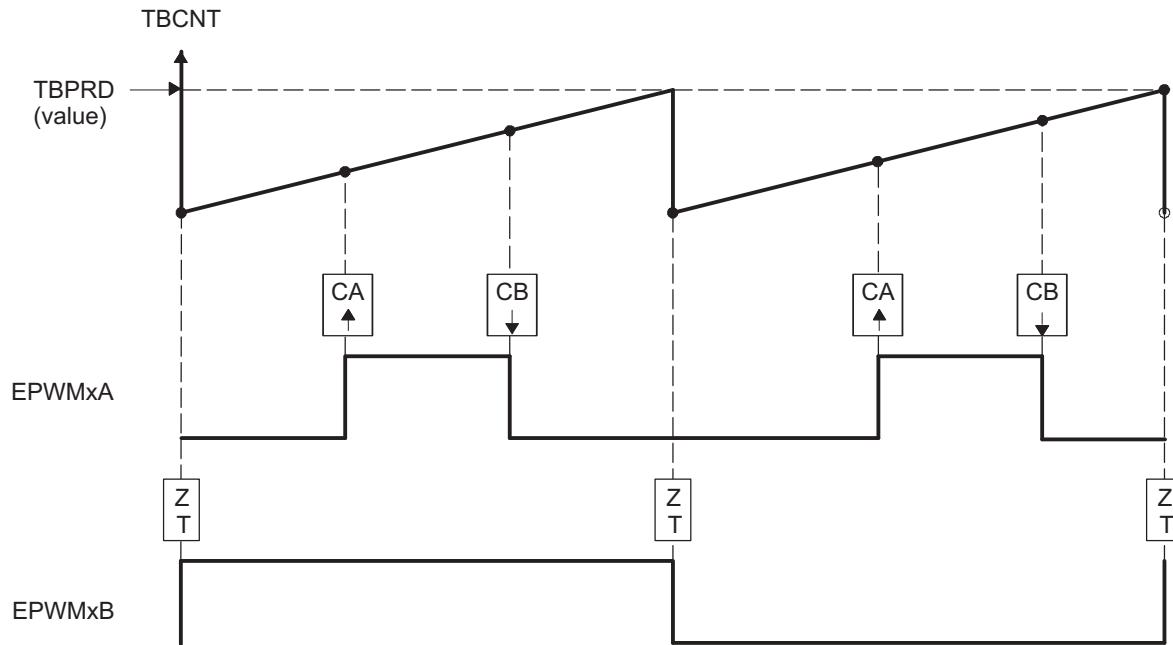
| Register | Bit       | Value           | Comments                     |
|----------|-----------|-----------------|------------------------------|
| TBPRD    | TBPRD     | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0    |
| TBCNT    | TBCNT     | 0               | Clear TB counter             |
| TBCTL    | CTRMODE   | TB_UP           |                              |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled       |
|          | PRDLD     | TB_SHADOW       |                              |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                              |
|          | HSPCLKDIV | TB_DIV1         | TBCLK = SYSCLK               |
|          | CLKDIV    | TB_DIV1         |                              |
| CMPA     | CMPA      | 350 (15Eh)      | Compare A = 350 TBCLK counts |
| CMPB     | CMPB      | 200 (C8h)       | Compare B = 200 TBCLK counts |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                              |
|          | SHDWBMODE | CC_SHADOW       |                              |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0              |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA   | PRD       | AQ_CLEAR        |                              |
|          | CAU       | AQ_SET          |                              |
| AQCTLB   | PRD       | AQ_CLEAR        |                              |
|          | CBU       | AQ_SET          |                              |

**Table 15-24. EPWMx Run Time Changes for Figure 15-28**

| Register | Bit  | Value  | Comments                      |
|----------|------|--------|-------------------------------|
| CMPA     | CMPA | Duty1A | Adjust duty for output EPWM1A |
| CMPB     | CMPB | Duty1B | Adjust duty for output EPWM1B |

Table 15-25 and Table 15-26 contains initialization and runtime register configurations for the waveforms Figure 15-29. Use the code in Example 15-1 to define the headers.

**Figure 15-29. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**



- (1)  $\text{PWM frequency} = 1 / (\text{TBPRD} + 1) \times T_{\text{TBCLK}}$
- (2) Pulse can be placed anywhere within the PWM cycle (0000h - TBPRD)
- (3) High time duty proportional to (CMPB - CMPA)
- (4) EPWMxB can be used to generate a 50% duty square wave with frequency  $= 1/2 \times ((\text{TBPRD} + 1) \times \text{TBCLK})$

**Table 15-25. EPWMx Initialization for Figure 15-29**

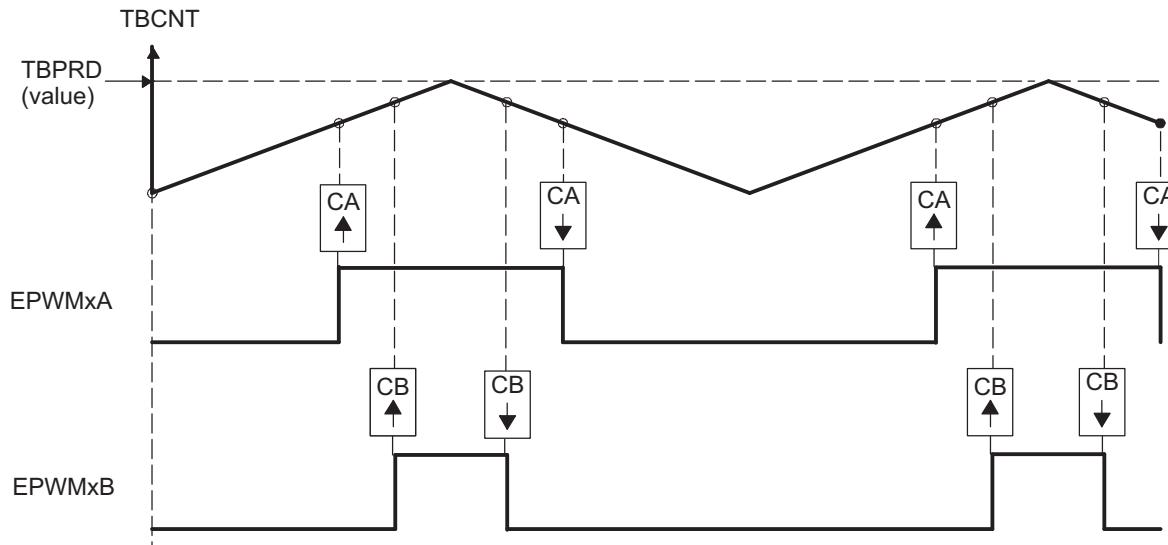
| Register | Bit       | Value           | Comments                     |
|----------|-----------|-----------------|------------------------------|
| TBPRD    | TBPRD     | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0    |
| TBCNT    | TBCNT     | 0               | Clear TB counter             |
| TBCTL    | CTRMODE   | TB_UP           |                              |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled       |
|          | PRDLD     | TB_SHADOW       |                              |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                              |
|          | HSPCLKDIV | TB_DIV1         | TBCLK = SYSCLK               |
|          | CLKDIV    | TB_DIV1         |                              |
| CMPA     | CMPA      | 200 (C8h)       | Compare A = 200 TBCLK counts |
| CMPB     | CMPB      | 400 (190h)      | Compare B = 400 TBCLK counts |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                              |
|          | SHDWBMODE | CC_SHADOW       |                              |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0              |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA   | CAU       | AQ_SET          |                              |
|          | CBU       | AQ_CLEAR        |                              |
| AQCTLB   | ZRO       | AQ_TOGGLE       |                              |

**Table 15-26. EPWMx Run Time Changes for Figure 15-29**

| Register | Bit  | Value    | Comments                      |
|----------|------|----------|-------------------------------|
| CMPA     | CMPA | EdgePosA | Adjust duty for output EPWM1A |
| CMPB     | CMPB | EdgePosB |                               |

Table 15-27 and Table 15-28 contains initialization and runtime register configurations for the waveforms in Figure 15-30. Use the code in Example 15-1 to define the headers.

**Figure 15-30. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low**



- (1) PWM period =  $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).
- (3) Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).
- (4) Outputs EPWMxA and EPWMxB can drive independent power switches

**Table 15-27. EPWMx Initialization for Figure 15-30**

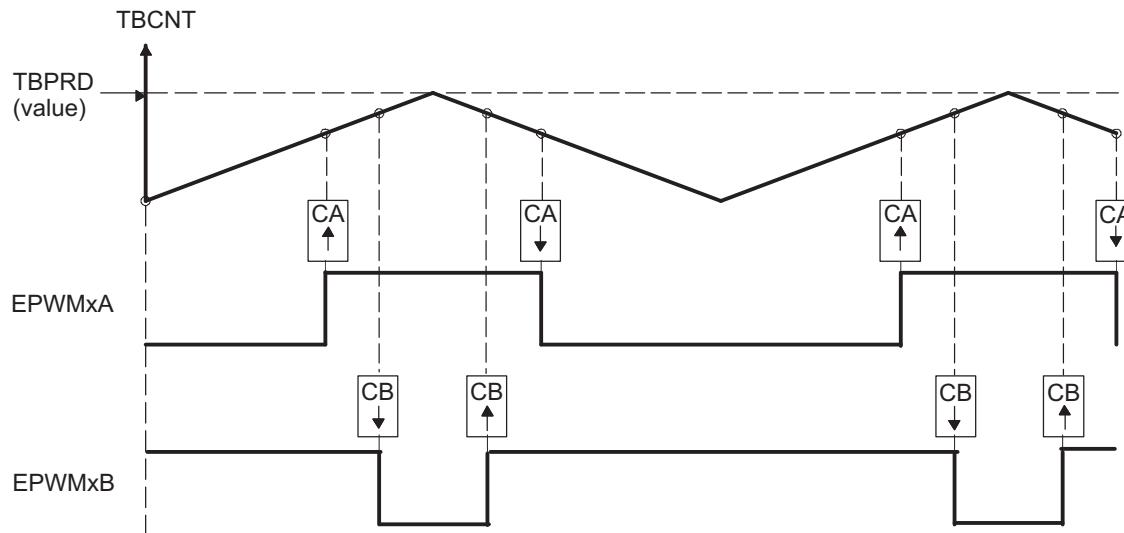
| Register | Bit       | Value           | Comments                     |
|----------|-----------|-----------------|------------------------------|
| TBPRD    | TBPRD     | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0    |
| TBCNT    | TBCNT     | 0               | Clear TB counter             |
| TBCTL    | CTRMODE   | TB_UPDOWN       |                              |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled       |
|          | PRDLD     | TB_SHADOW       |                              |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                              |
|          | HSPCLKDIV | TB_DIV1         | TBCLK = SYSCLK               |
|          | CLKDIV    | TB_DIV1         |                              |
| CMPA     | CMPA      | 400 (190h)      | Compare A = 400 TBCLK counts |
| CMPB     | CMPB      | 500 (1F4h)      | Compare B = 500 TBCLK counts |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                              |
|          | SHDWBMODE | CC_SHADOW       |                              |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0              |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA   | CAU       | AQ_SET          |                              |
|          | CAD       | AQ_CLEAR        |                              |
| AQCTLB   | CBU       | AQ_SET          |                              |
|          | CBD       | AQ_CLEAR        |                              |

**Table 15-28. EPWMx Run Time Changes for Figure 15-30**

| Register | Bit  | Value  | Comments                      |
|----------|------|--------|-------------------------------|
| CMPA     | CMPA | Duty1A | Adjust duty for output EPWM1A |
| CMPB     | CMPB | Duty1B | Adjust duty for output EPWM1B |

Table 15-29 and Table 15-30 contains initialization and runtime register configurations for the waveforms in Figure 15-31. Use the code in Example 15-1 to define the headers.

**Figure 15-31. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary**



- (1) PWM period =  $2 \times \text{TBPRD} \times T_{\text{TBCLK}}$
- (2) Duty modulation for EPWMxA is set by CMPA, and is active low, i.e., low time duty proportional to CMPA
- (3) Duty modulation for EPWMxB is set by CMPB and is active high, i.e., high time duty proportional to CMPB
- (4) Outputs EPWMx can drive upper/lower (complementary) power switches
- (5) Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

**Table 15-29. EPWMx Initialization for Figure 15-31**

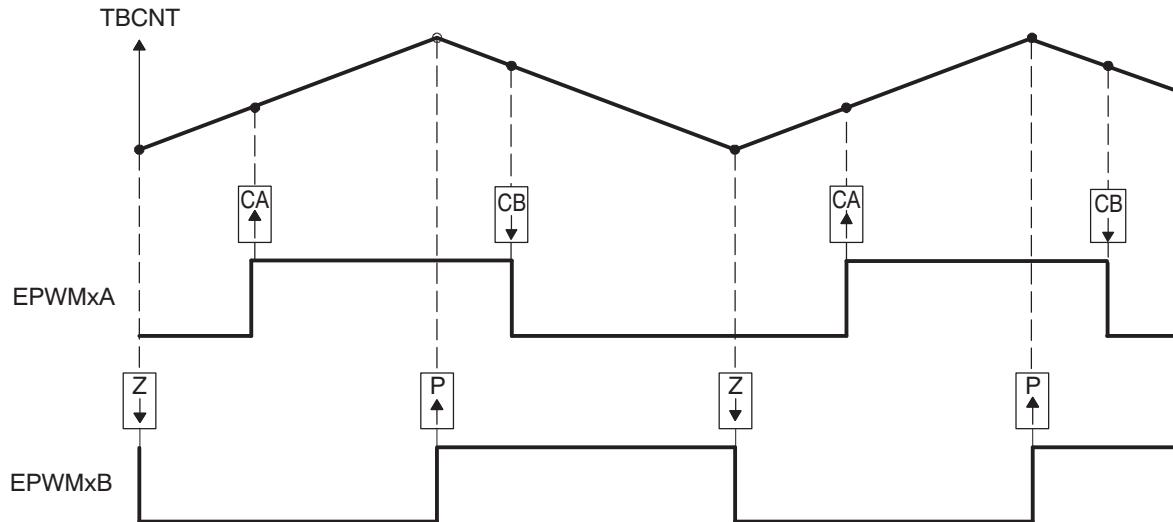
| <b>Register</b> | <b>Bit</b> | <b>Value</b>    | <b>Comments</b>              |
|-----------------|------------|-----------------|------------------------------|
| TBPRD           | TBPRD      | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS           | TBPHS      | 0               | Clear Phase Register to 0    |
| TBCNT           | TBCNT      | 0               | Clear TB counter             |
| TBCTL           | CTRMODE    | TB_UPDOWN       |                              |
|                 | PHSEN      | TB_DISABLE      | Phase loading disabled       |
|                 | PRDLD      | TB_SHADOW       |                              |
|                 | SYNCOSEL   | TB_SYNC_DISABLE |                              |
|                 | HSPCLKDIV  | TB_DIV1         | TBCLK = SYSCLK               |
|                 | CLKDIV     | TB_DIV1         |                              |
| CMPA            | CMPA       | 350 (15Eh)      | Compare A = 350 TBCLK counts |
| CMPB            | CMPB       | 400 (190h)      | Compare B = 400 TBCLK counts |
| CMPCTL          | SHDWAMODE  | CC_SHADOW       |                              |
|                 | SHDWBMODE  | CC_SHADOW       |                              |
|                 | LOADAMODE  | CC_CTR_ZERO     | Load on CTR = 0              |
|                 | LOADBMODE  | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA          | CAU        | AQ_SET          |                              |
|                 | CAD        | AQ_CLEAR        |                              |
| AQCTLB          | CBU        | AQ_CLEAR        |                              |
|                 | CBD        | AQ_SET          |                              |

**Table 15-30. EPWMx Run Time Changes for Figure 15-31**

| <b>Register</b> | <b>Bit</b> | <b>Value</b> | <b>Comments</b>               |
|-----------------|------------|--------------|-------------------------------|
| CMPA            | CMPA       | Duty1A       | Adjust duty for output EPWM1A |
| CMPB            | CMPB       | Duty1B       | Adjust duty for output EPWM1B |

Table 15-31 and Table 15-32 contains initialization and runtime register configurations for the waveforms in Figure 15-32. Use the code in Example 15-1 to define the headers.

**Figure 15-32. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low**



- (1) PWM period =  $2 \times \text{TBPRD} \times \text{TBCLK}$
- (2) Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
- (3) Duty modulation for EPWMxA is set by CMPA and CMPB.
- (4) Low time duty for EPWMxA is proportional to  $(\text{CMPA} + \text{CMPB})$ .
- (5) To change this example to active high, CMPA and CMPB actions need to be inverted (i.e., Set ! Clear and Clear Set).
- (6) Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

**Table 15-31. EPWMx Initialization for Figure 15-32**

| Register | Bit       | Value           | Comments                     |
|----------|-----------|-----------------|------------------------------|
| TBPRD    | TBPRD     | 600 (258h)      | Period = 601 TBCLK counts    |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0    |
| TBCNT    | TBCNT     | 0               | Clear TB counter             |
| TBCTL    | CTRMODE   | TB_UPDOWN       |                              |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled       |
|          | PRDLD     | TB_SHADOW       |                              |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                              |
|          | HSPCLKDIV | TB_DIV1         | TBCLK = SYSCLK               |
|          | CLKDIV    | TB_DIV1         |                              |
| CMPA     | CMPA      | 250 (FAh)       | Compare A = 250 TBCLK counts |
| CMPB     | CMPB      | 450 (1C2h)      | Compare B = 450 TBCLK counts |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                              |
|          | SHDWBMODE | CC_SHADOW       |                              |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0              |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0              |
| AQCTLA   | CAU       | AQ_SET          |                              |
|          | CBD       | AQ_CLEAR        |                              |
| AQCTLB   | ZRO       | AQ_CLEAR        |                              |
|          | PRD       | AQ_SET          |                              |

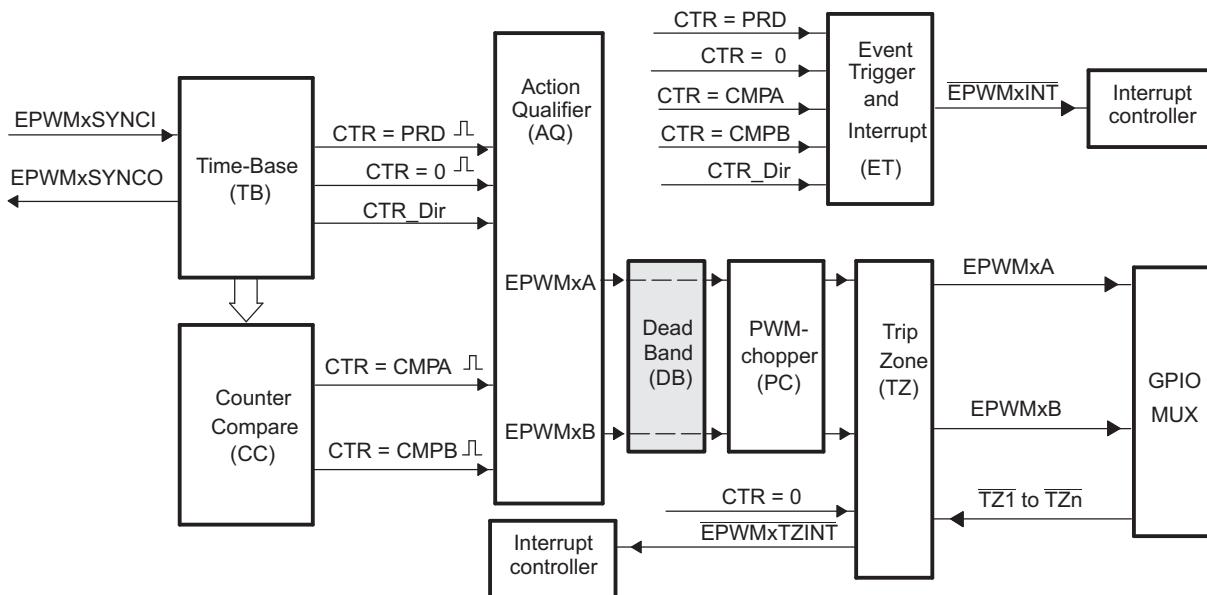
**Table 15-32. EPWMx Run Time Changes for Figure 15-32**

| Register | Bit  | Value    | Comments                      |
|----------|------|----------|-------------------------------|
| CMPA     | CMPA | EdgePosA | Adjust duty for output EPWM1A |
| CMPB     | CMPB | EdgePosB |                               |

### 15.2.2.6 Dead-Band Generator (DB) Submodule

Figure 15-33 illustrates the dead-band generator submodule within the ePWM module.

**Figure 15-33. Dead-Band Generator Submodule**



#### 15.2.2.6.1 Purpose of the Dead-Band Submodule

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band generator submodule should be used.

The key functions of the dead-band generator submodule are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

#### 15.2.2.6.2 Controlling and Monitoring the Dead-Band Submodule

The dead-band generator submodule operation is controlled and monitored via the following registers:

**Table 15-33. Dead-Band Generator Submodule Registers**

| Acronym | Register Description                        | Address Offset | Shadowed |
|---------|---|----------------|----------|
| DBCTL   | Dead-Band Control Register                  | 1Eh            | No       |
| DBRED   | Dead-Band Rising Edge Delay Count Register  | 20h            | No       |
| DBFED   | Dead-Band Falling Edge Delay Count Register | 22h            | No       |

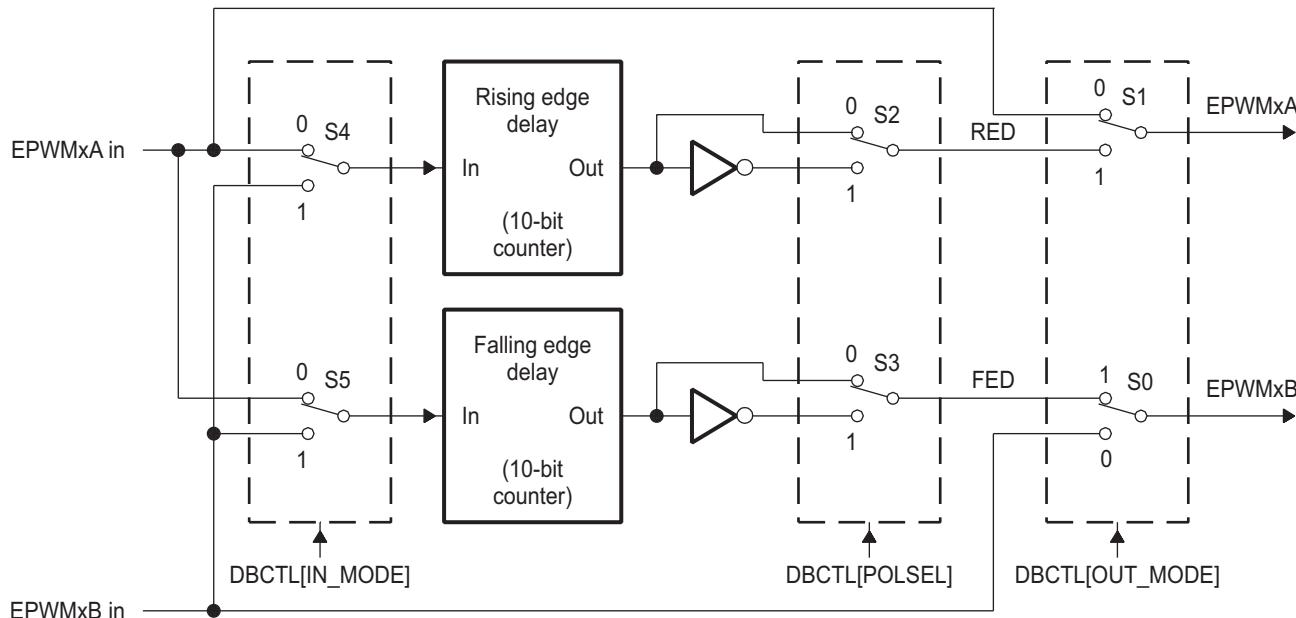
### 15.2.2.6.3 Operational Highlights for the Dead-Band Generator Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in [Figure 15-34](#).

- **Input Source Selection:** The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the DBCTL[IN\_MODE] control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:
  - EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  - EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  - EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  - EPWMxB In is the source for both falling-edge and rising-edge delay.
- **Output Mode Control:** The output mode is configured by way of the DBCTL[OUT\_MODE] bits. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.
- **Polarity Control:** The polarity control (DBCTL[POLSEL]) allows you to specify whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

**Figure 15-34. Configuration Options for the Dead-Band Generator Submodule**



Although all combinations are supported, not all are typical usage modes. [Table 15-34](#) lists some classical dead-band configurations. These modes assume that the DBCTL[IN\_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in [Table 15-34](#) fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)** Allows you to fully disable the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings** These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in [Figure 15-35](#). Note that to generate equivalent waveforms to [Figure 15-35](#), configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay** Finally the last two entries in [Table 15-34](#) show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

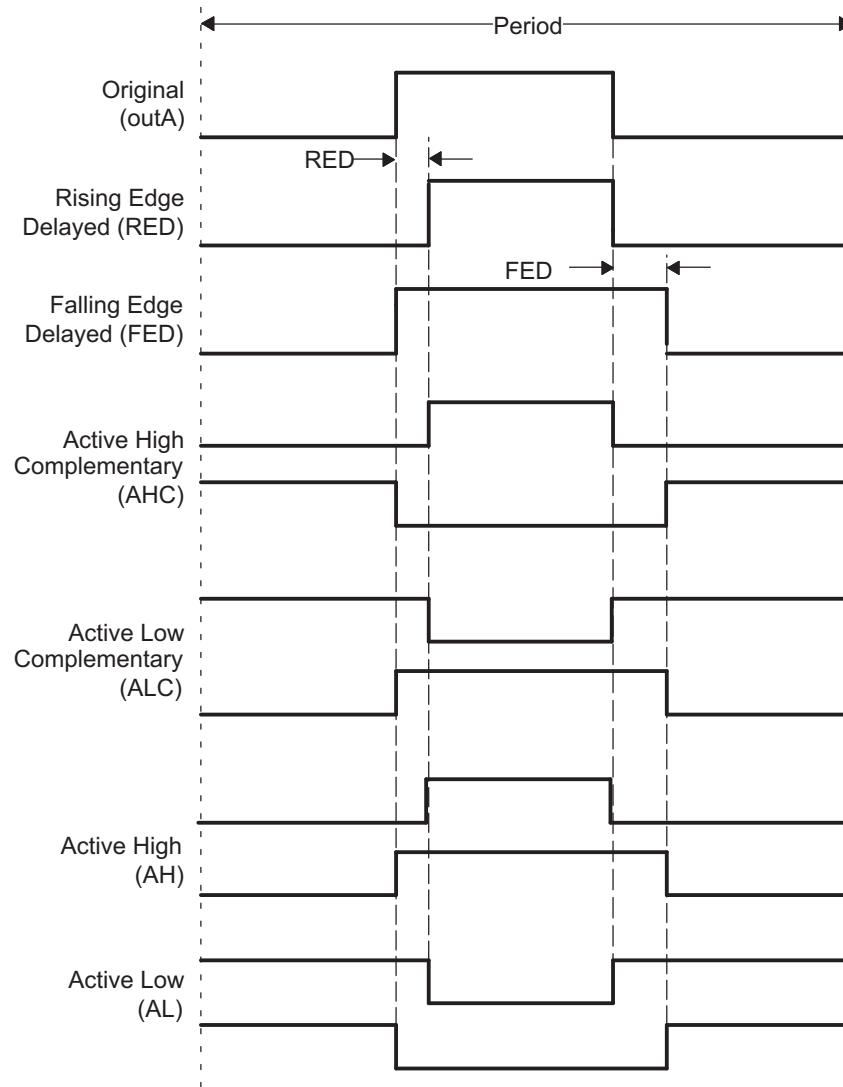
**Table 15-34. Classical Dead-Band Operating Modes**

| Mode | Mode Description <sup>(1)</sup>   | DBCTL[POLSEL] |        | DBCTL[OUT_MODE] |    |
|------|---|---------------|--------|-----------------|----|
|      |   | S3            | S2     | S1              | S0 |
| 1    | EPWMxA and EPWMxB Passed Through (No Delay)   | x             | x      | 0               | 0  |
| 2    | Active High Complementary (AHC)   | 1             | 0      | 1               | 1  |
| 3    | Active Low Complementary (ALC)  | 0             | 1      | 1               | 1  |
| 4    | Active High (AH)  | 0             | 0      | 1               | 1  |
| 5    | Active Low (AL)   | 1             | 1      | 1               | 1  |
| 6    | EPWMxA Out = EPWMxA In (No Delay)<br>EPWMxB Out = EPWMxA In with Falling Edge Delay   | 0 or 1        | 0 or 1 | 0               | 1  |
| 7    | EPWMxA Out = EPWMxA In with Rising Edge Delay<br>EPWMxB Out = EPWMxB In with No Delay | 0 or 1        | 0 or 1 | 1               | 0  |

<sup>(1)</sup> These are classical dead-band modes and assume that DBCTL[IN\_MODE] = 0,0. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN\_MODE configuration.

Figure 15-35 shows waveforms for typical cases where  $0\% < \text{duty} < 100\%$ .

**Figure 15-35. Dead-Band Waveforms for Typical Cases ( $0\% < \text{Duty} < 100\%$ )**



The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formula to calculate falling-edge-delay and rising-edge-delay are:

$$\text{FED} = \text{DBFED} \times T_{\text{TBCLK}}$$

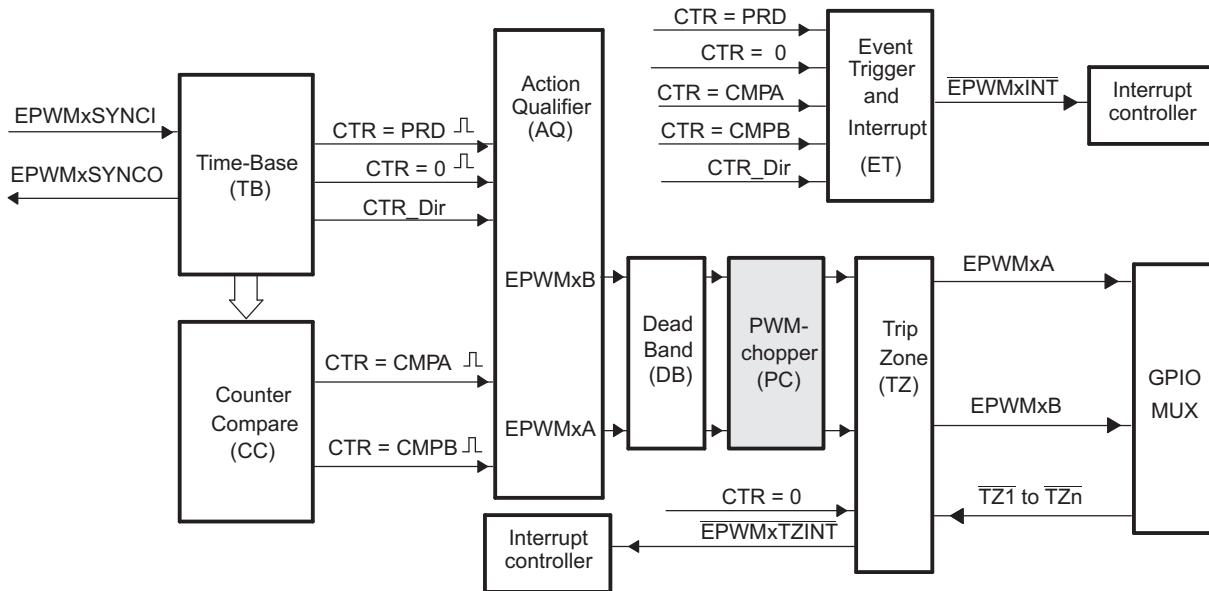
$$\text{RED} = \text{DBRED} \times T_{\text{TBCLK}}$$

Where  $T_{\text{TBCLK}}$  is the period of TBCLK, the prescaled version of SYSCLKOUT.

### 15.2.2.7 PWM-Chopper (PC) Submodule

**Figure 15-36** illustrates the PWM-chopper (PC) submodule within the ePWM module. The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

**Figure 15-36. PWM-Chopper Submodule**



#### 15.2.2.7.1 Purpose of the PWM-Chopper Submodule

The key functions of the PWM-chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

#### 15.2.2.7.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the register in [Table 15-35](#).

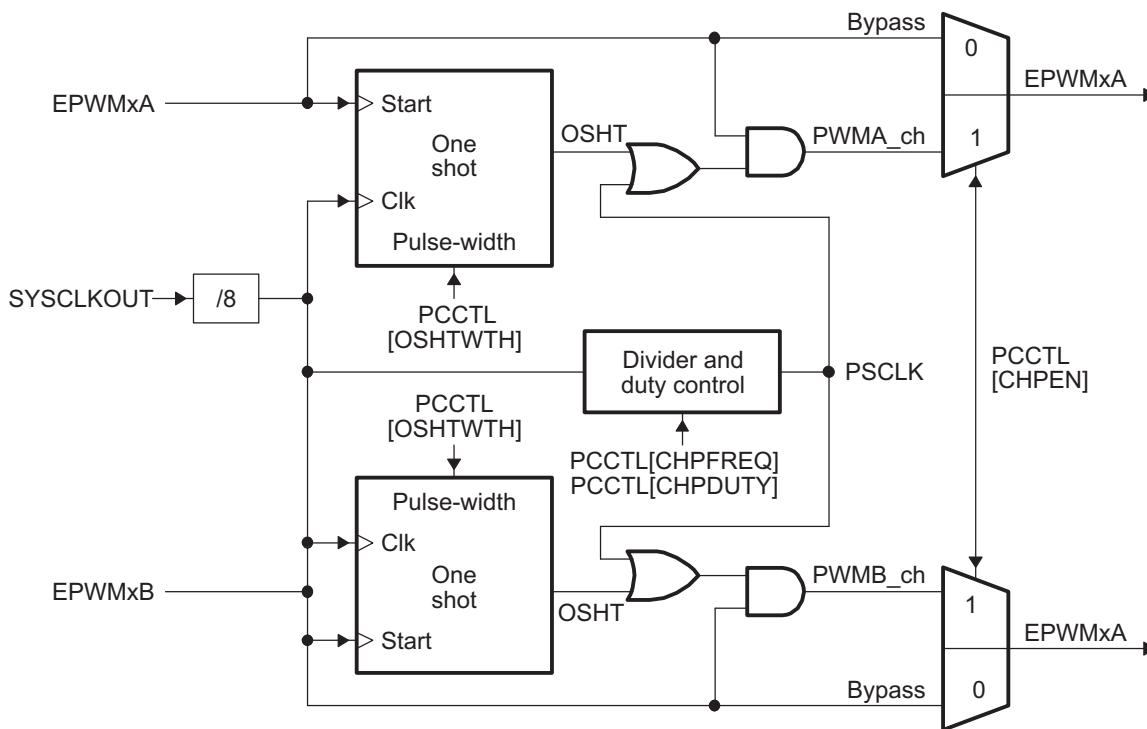
**Table 15-35. PWM-Chopper Submodule Registers**

| Acronym | Register Description         | Address Offset | Shadowed |
|---------|------------------------------|----------------|----------|
| PCCTL   | PWM-chopper Control Register | 3Ch            | No       |

### 15.2.2.7.3 Operational Highlights for the PWM-Chopper Submodule

Figure 15-37 shows the operational details of the PWM-chopper submodule. The carrier clock is derived from SYSCLKOUT. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

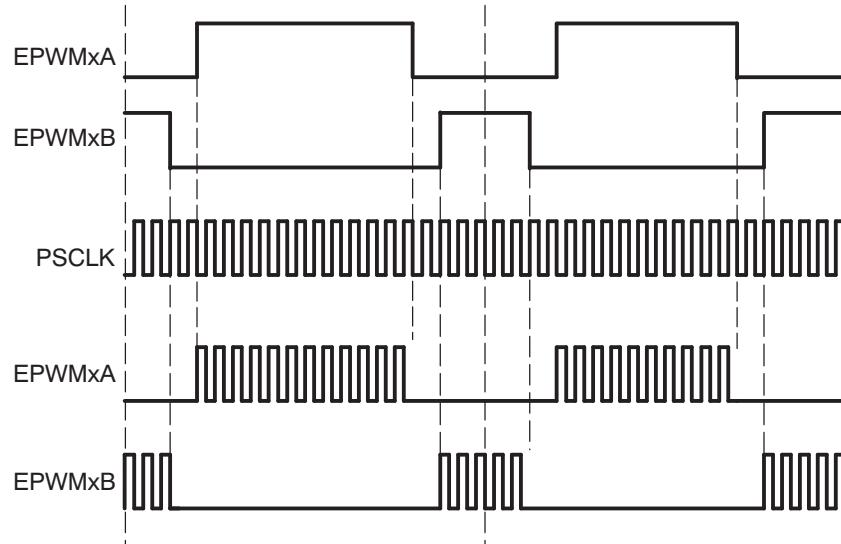
**Figure 15-37. PWM-Chopper Submodule Signals and Registers**



#### 15.2.2.7.4 Waveforms

Figure 15-38 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

**Figure 15-38. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only**



##### 15.2.2.7.4.1 One-Shot Pulse

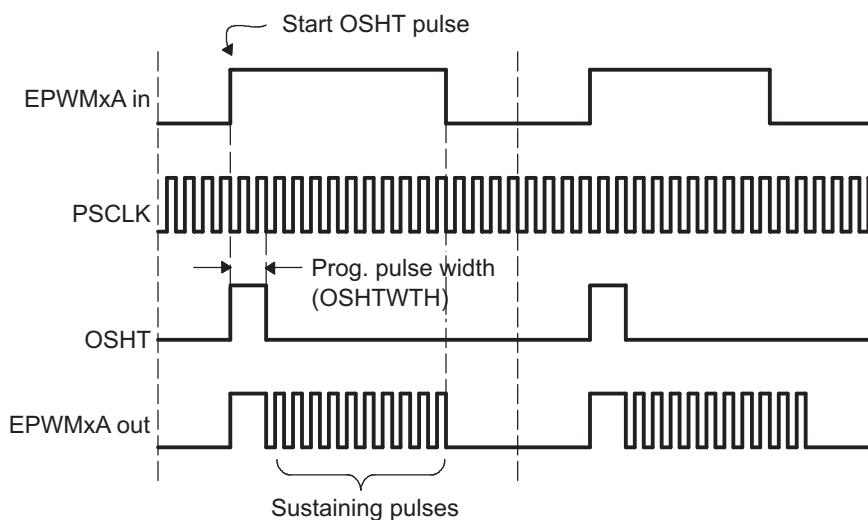
The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1\text{st} \text{pulse}} = T_{\text{SYSCLKOUT}} \times 8 \times \text{OSHTWTH}$$

Where  $T_{\text{SYSCLKOUT}}$  is the period of the system clock (SYSCLKOUT) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 15-39 shows the first and subsequent sustaining pulses.

**Figure 15-39. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

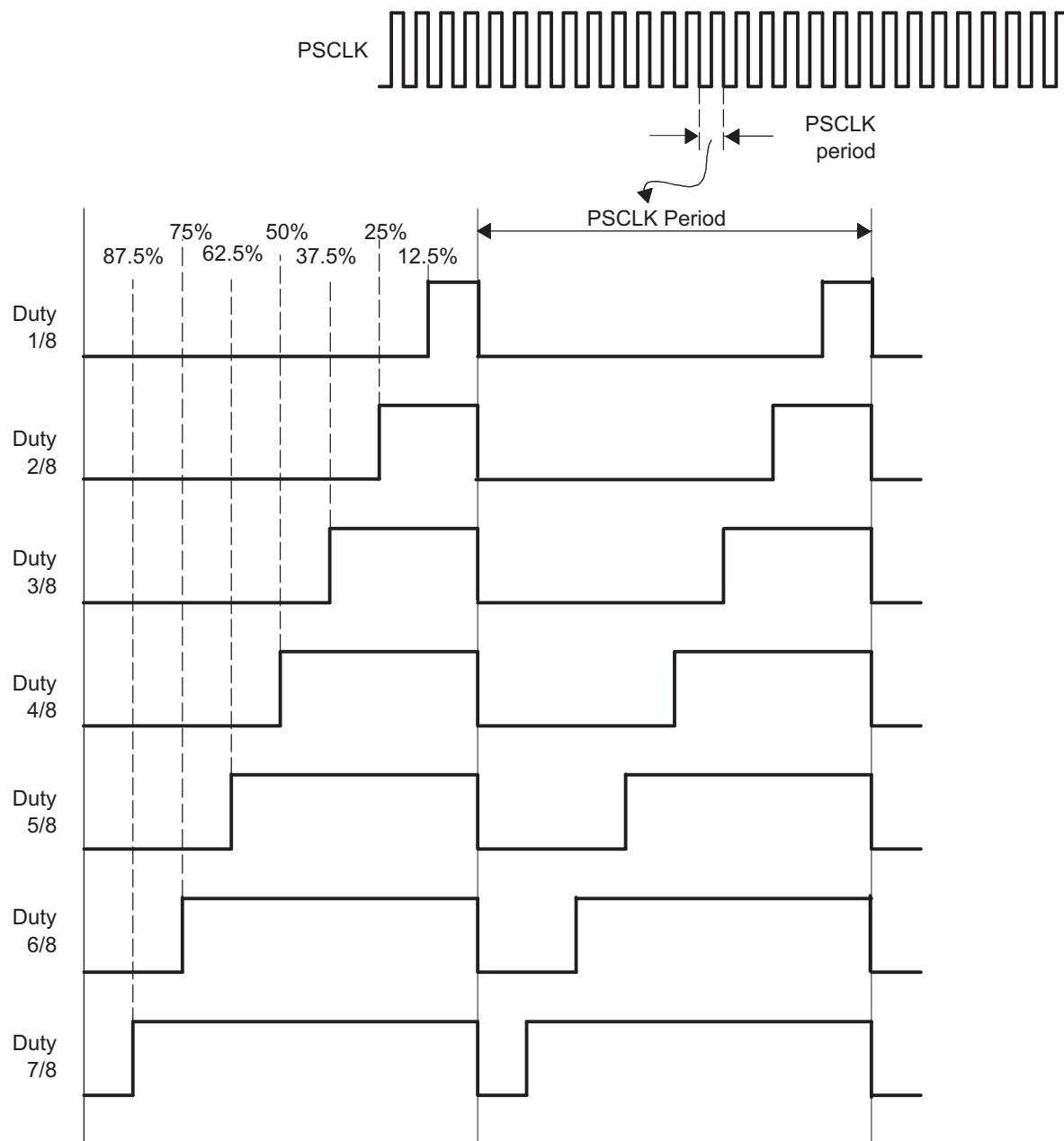


#### 15.2.2.7.4.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

[Figure 15-40](#) shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

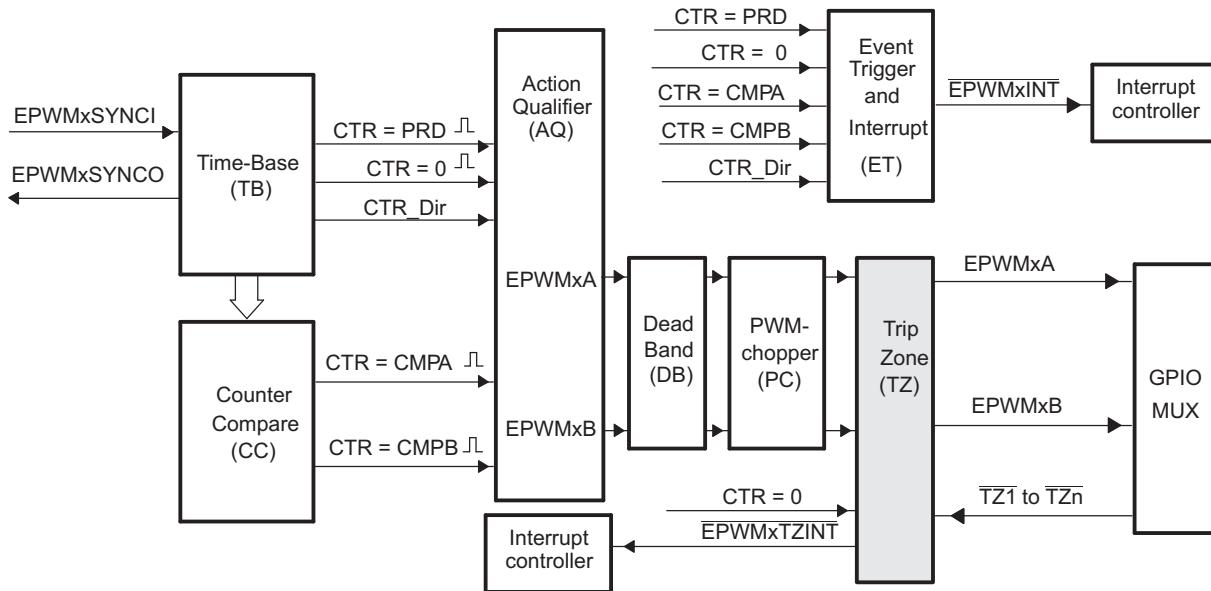
**Figure 15-40. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**



### 15.2.2.8 Trip-Zone (TZ) Submodule

Figure 15-41 shows how the trip-zone (TZ) submodule fits within the ePWM module. Each ePWM module is connected to every  $\overline{TZ}$  signal that are sourced from the GPIO MUX. These signals indicates external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur. See Section 15.1.2 to determine the number of trip-zone pins available for the device.

**Figure 15-41. Trip-Zone Submodule**



#### 15.2.2.8.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs  $\overline{TZ1}$  to  $\overline{TZn}$  can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

### 15.2.2.8.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

**Table 15-36. Trip-Zone Submodule Registers**

| Acronym | Register Description                | Address Offset | Shadowed |
|---------|-------------------------------------|----------------|----------|
| TZSEL   | Trip-Zone Select Register           | 24h            | No       |
| TZCTL   | Trip-Zone Control Register          | 28h            | No       |
| TZEINT  | Trip-Zone Enable Interrupt Register | 2Ah            | No       |
| TZFLG   | Trip-Zone Flag Register             | 2Ch            | No       |
| TZCLR   | Trip-Zone Clear Register            | 2Eh            | No       |
| TZFCR   | Trip-Zone Force Register            | 30h            | No       |

### 15.2.2.8.3 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals at pin  $\overline{TZ1}$  to  $\overline{TZn}$  is an active-low input signal. When the pin goes low, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signal may or may not be synchronized to the system clock (SYSCLKOUT). A minimum of 1 SYSCLKOUT low pulse on the  $\overline{TZ_n}$  inputs is sufficient to trigger a fault condition in the ePWM module. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on the  $\overline{TZn}$  inputs.

The  $\overline{TZ_n}$  input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a ePWM module. The configuration is determined by the TZSEL[CBCn] and TZSEL[OSHTn] bits (where n corresponds to the trip pin) respectively.

- **Cycle-by-Cycle (CBC):** When a cycle-by-cycle trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 15-37](#) lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register.

The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] flag bit will remain set until it is manually cleared by writing to the TZCLR[CBC] bit. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] bit is cleared, then it will again be immediately set.

- **One-Shot (OSHT):** When a one-shot trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. [Table 15-37](#) lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMxTZINT interrupt is generated if it is enabled in the TZEINT register. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL[TZA] and TZCTL[TZB] register bits. One of four possible actions, shown in [Table 15-37](#), can be taken on a trip event.

**Table 15-37. Possible Actions On a Trip Event**

| TZCTL[TZA]<br>and/or<br>TZCTL[TZB] | EPWMx <sub>A</sub><br>and/or<br>EPWMx <sub>B</sub> | Comment                                      |
|------------------------------------|--|--|
| 0                                  | High-Impedance                                     | Tripped                                      |
| 1h                                 | Force to High State                                | Tripped                                      |
| 2h                                 | Force to Low State                                 | Tripped                                      |
| 3h                                 | No Change  | Do Nothing. No change is made to the output. |

**Example 15-2. Trip-Zone Configurations****Scenario A:**

A one-shot trip event on  $\overline{\text{TZ1}}$  pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{\text{TZ}}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{\text{TZ}}$  as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

**Scenario B:**

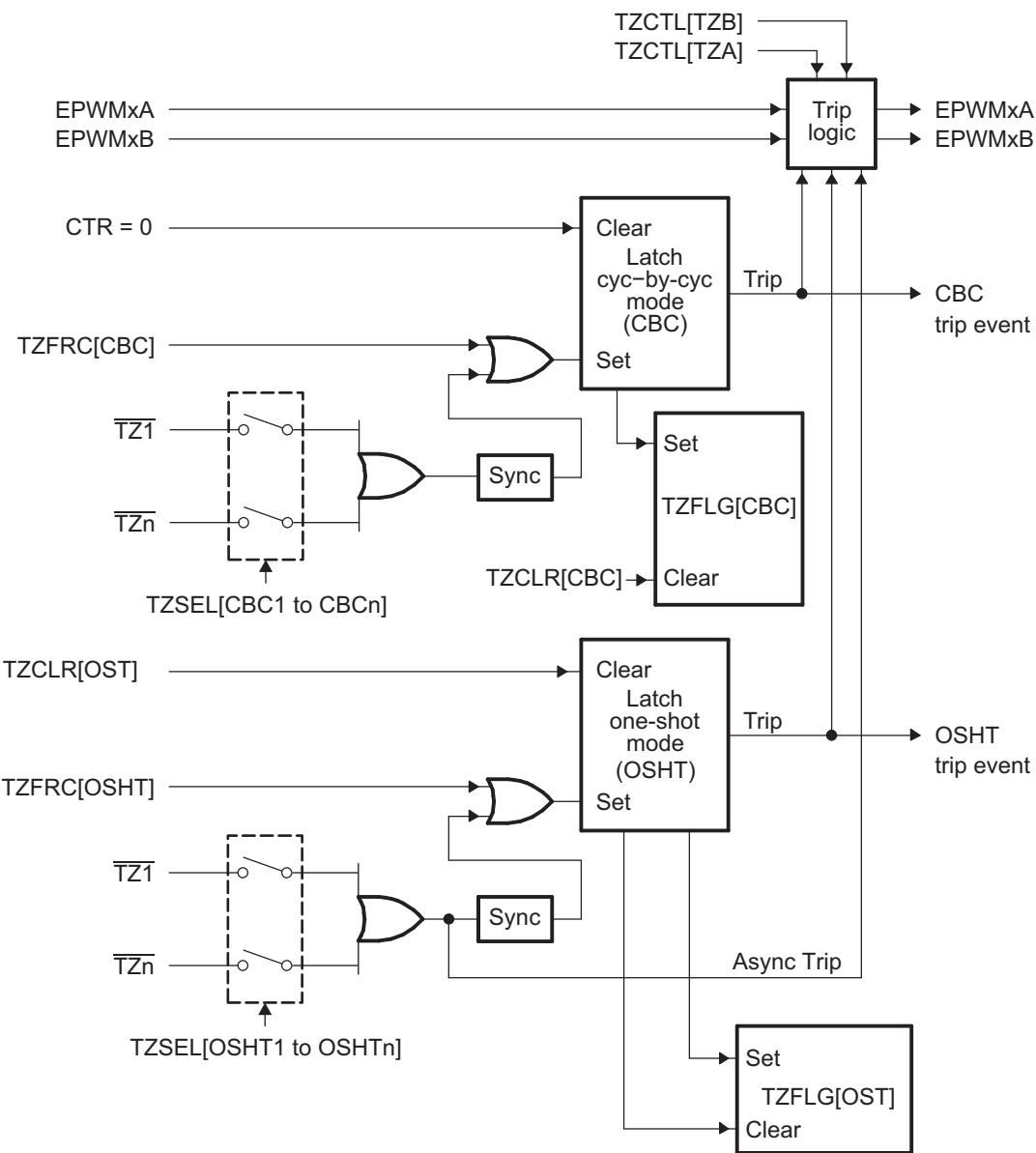
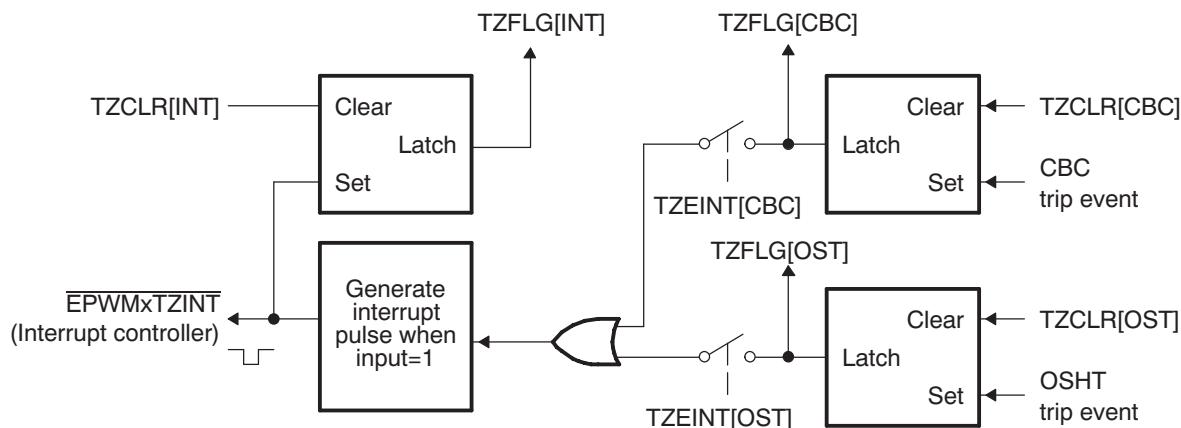
A cycle-by-cycle event on  $\overline{\text{TZ5}}$  pulls both EPWM1A, EPWM1B low.

A one-shot event on  $\overline{\text{TZ1}}$  or  $\overline{\text{TZ6}}$  puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables  $\overline{\text{TZ5}}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables  $\overline{\text{TZ1}}$  as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables  $\overline{\text{TZ6}}$  as a one-shot event source for ePWM1
  - TZCTL[TZA] = 0: EPWM1A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM1B will ignore the trip event.

**15.2.2.8.4 Generating Trip Event Interrupts**

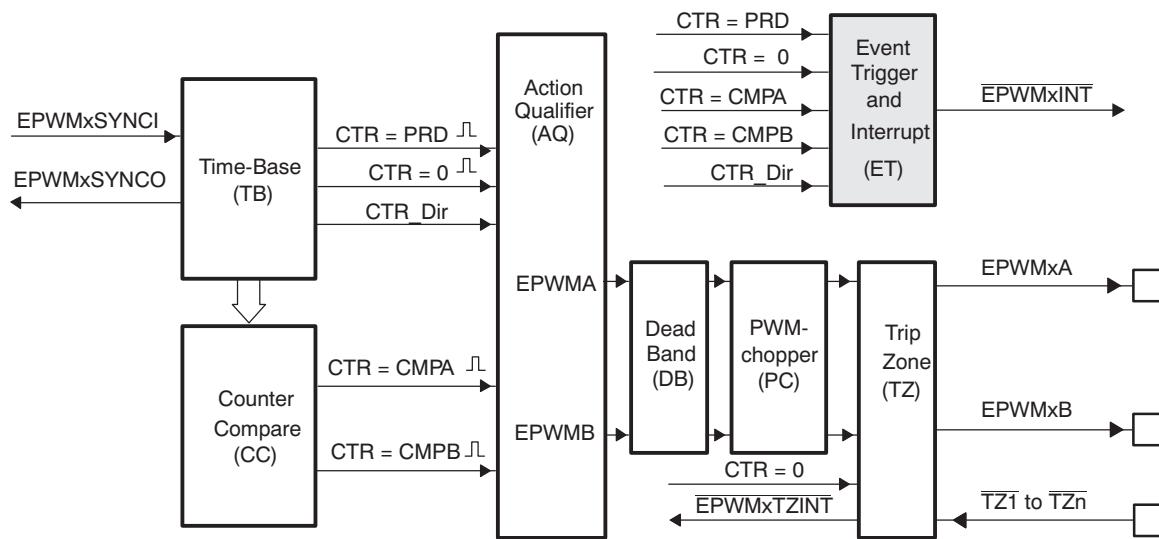
Figure 15-42 and Figure 15-43 illustrate the trip-zone submodule control and interrupt logic, respectively.

**Figure 15-42. Trip-Zone Submodule Mode Control Logic**

**Figure 15-43. Trip-Zone Submodule Interrupt Logic**


### 15.2.2.9 Event-Trigger (ET) Submodule

Figure 15-44 shows the event-trigger (ET) submodule in the ePWM system. The event-trigger submodule manages the events generated by the time-base submodule and the counter-compare submodule to generate an interrupt to the CPU.

**Figure 15-44. Event-Trigger Submodule**



#### 15.2.2.9.1 Purpose of the Event-Trigger Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base and counter-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags

#### 15.2.2.9.2 Controlling and Monitoring the Event-Trigger Submodule

The key registers used to configure the event-trigger submodule are shown in Table 15-38:

**Table 15-38. Event-Trigger Submodule Registers**

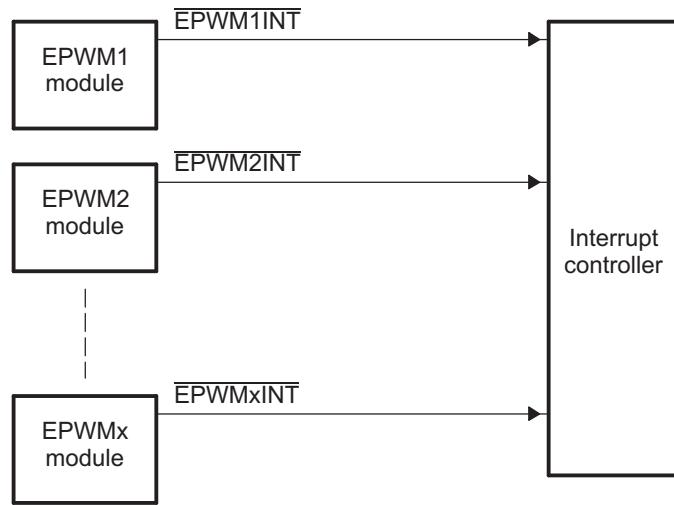
| Acronym | Register Description             | Address Offset | Shadowed |
|---------|----------------------------------|----------------|----------|
| ETSEL   | Event-Trigger Selection Register | 32h            | No       |
| ETPS    | Event-Trigger Prescale Register  | 34h            | No       |
| ETFLG   | Event-Trigger Flag Register      | 36h            | No       |
| ETCLR   | Event-Trigger Clear Register     | 38h            | No       |
| ETFRC   | Event-Trigger Force Register     | 3Ah            | No       |

### 15.2.2.9.3 Operational Overview of the Event-Trigger Submodule

The following sections describe the event-trigger submodule's operational highlights.

Each ePWM module has one interrupt request line connected to the interrupt controller as shown in Figure 15-45.

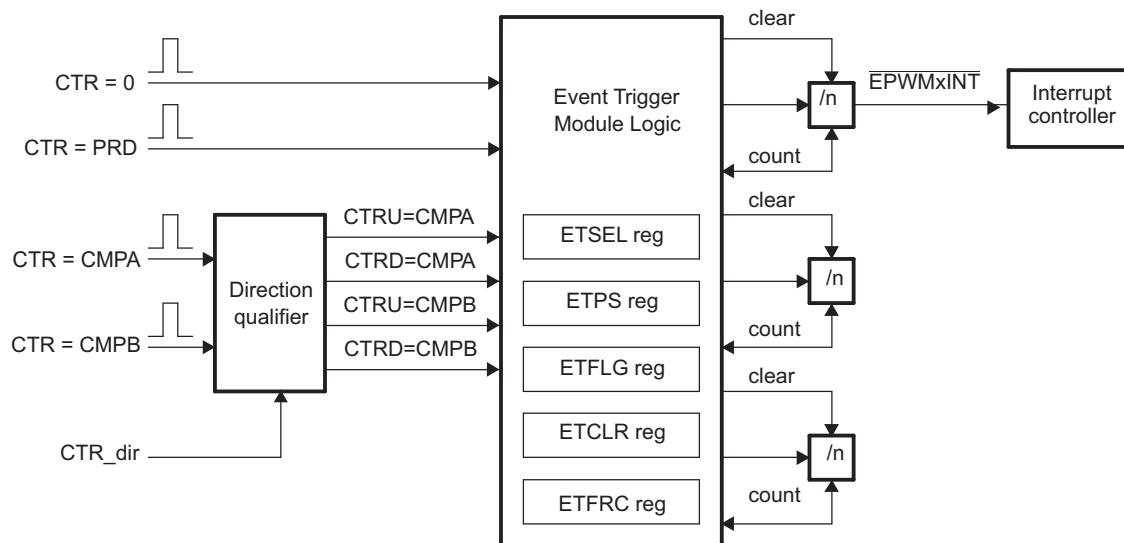
**Figure 15-45. Event-Trigger Submodule Inter-Connectivity to Interrupt Controller**



The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 15-46) and can be configured to prescale these events before issuing an interrupt request. The event-trigger prescaling logic can issue interrupt requests at:

- Every event
- Every second event
- Every third event

**Figure 15-46. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**



- ETSEL—This selects which of the possible events will trigger an interrupt.
- ETPS—This programs the event prescaling options previously mentioned.
- ETFLG—These are flag bits indicating status of the selected and prescaled events.
- ETCLR—These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC—These bits allow software forcing of an event. Useful for debugging or software intervention.

A more detailed look at how the various register bits interact with the Interrupt is shown in [Figure 15-47](#).

[Figure 15-47](#) shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

An interrupt cannot be generated on every fourth or more events.

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCNT = 0000h).
- Time-base counter equal to period (TBCNT = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.

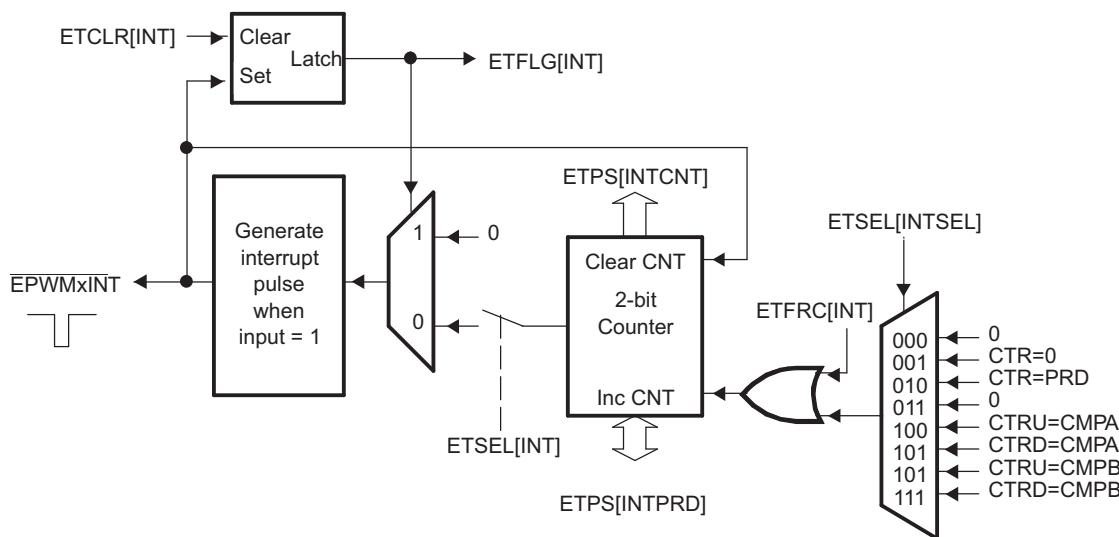
The number of events that have occurred can be read from the interrupt event counter (ETPS[INTCNT]) register bits. That is, when the specified event occurs the ETPS[INTCNT] bits are incremented until they reach the value specified by ETPS[INTPRD]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the interrupt controller.

When ETPS[INTCNT] reaches ETPS[INTPRD], one of the following behaviors will occur:

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCTN = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored.

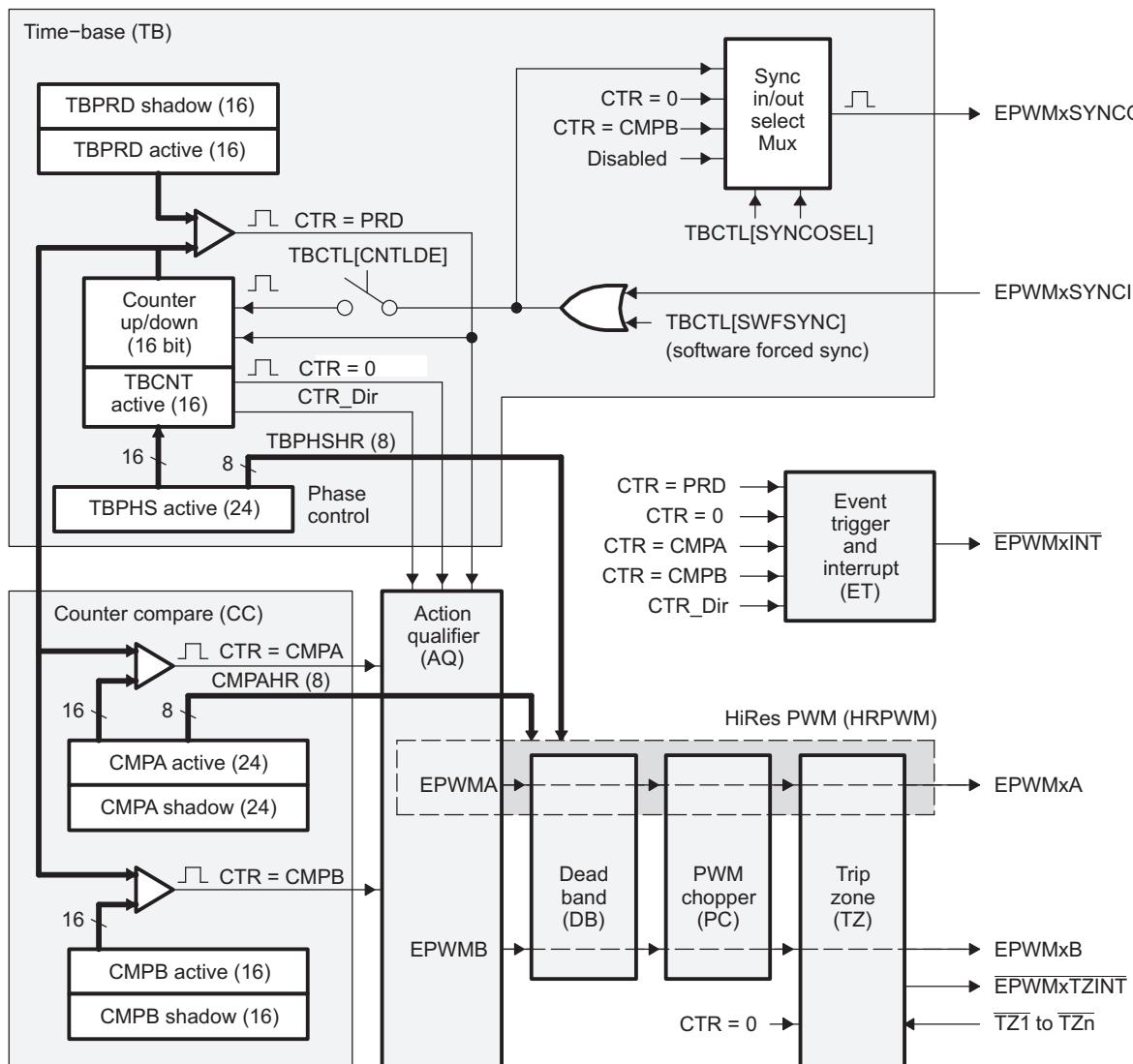
**Figure 15-47. Event-Trigger Interrupt Generator**



### 15.2.2.10 High-Resolution PWM (HRPWM) Submodule

Figure 15-48 shows the high-resolution PWM (HRPWM) submodule in the ePWM system. Some devices include the high-resolution PWM submodule, see [Section 15.1.2](#) to determine which ePWM instances include this feature.

**Figure 15-48. HRPWM System Interface**



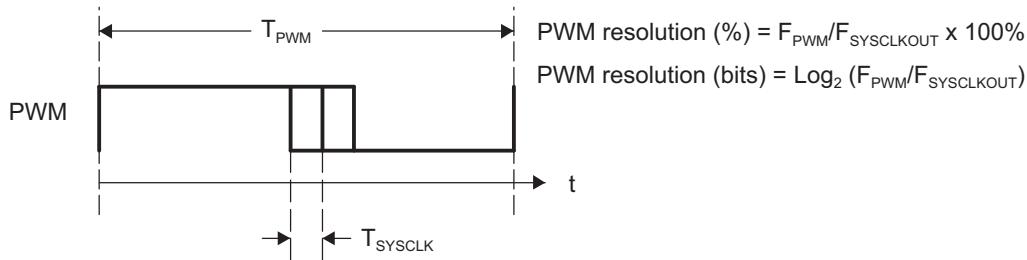
### 15.2.2.10.1 Purpose of the High-Resolution PWM Submodule

The enhanced high-resolution pulse-width modulator (eHRPWM) extends the time resolution capabilities of the conventionally derived digital pulse-width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer time granularity control or edge positioning using extensions to the Compare A and Phase registers
- Implemented using the A signal path of PWM, that is, on the EPWMxA output. EPWMxB output has conventional PWM capabilities

The ePWM peripheral is used to perform a function that is mathematically equivalent to a digital-to-analog converter (DAC). As shown in [Figure 15-49](#), the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

**Figure 15-49. Resolution Calculations for Conventionally Generated PWM**



If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, [Table 15-39](#) shows resolution in bits for various PWM frequencies. [Table 15-39](#) values assume a MEP step size of 180 ps. See your device-specific data manual for typical and maximum performance specifications for the MEP.

**Table 15-39. Resolution for PWM and HRPWM**

| PWM Frequency (kHz) | Regular Resolution (PWM) |     | High Resolution (HRPWM) |       |
|---------------------|--------------------------|-----|-------------------------|-------|
|                     | Bits                     | %   | Bits                    | %     |
| 20                  | 12.3                     | 0.0 | 18.1                    | 0.000 |
| 50                  | 11.0                     | 0.0 | 16.8                    | 0.001 |
| 100                 | 10.0                     | 0.1 | 15.8                    | 0.002 |
| 150                 | 9.4                      | 0.2 | 15.2                    | 0.003 |
| 200                 | 9.0                      | 0.2 | 14.8                    | 0.004 |
| 250                 | 8.6                      | 0.3 | 14.4                    | 0.005 |
| 500                 | 7.6                      | 0.5 | 13.8                    | 0.007 |
| 1000                | 6.6                      | 1.0 | 12.4                    | 0.018 |
| 1500                | 6.1                      | 1.5 | 11.9                    | 0.027 |
| 2000                | 5.6                      | 2.0 | 11.4                    | 0.036 |

Although each application may differ, typical low-frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high-frequency PWM requirements of power conversion topologies such as:

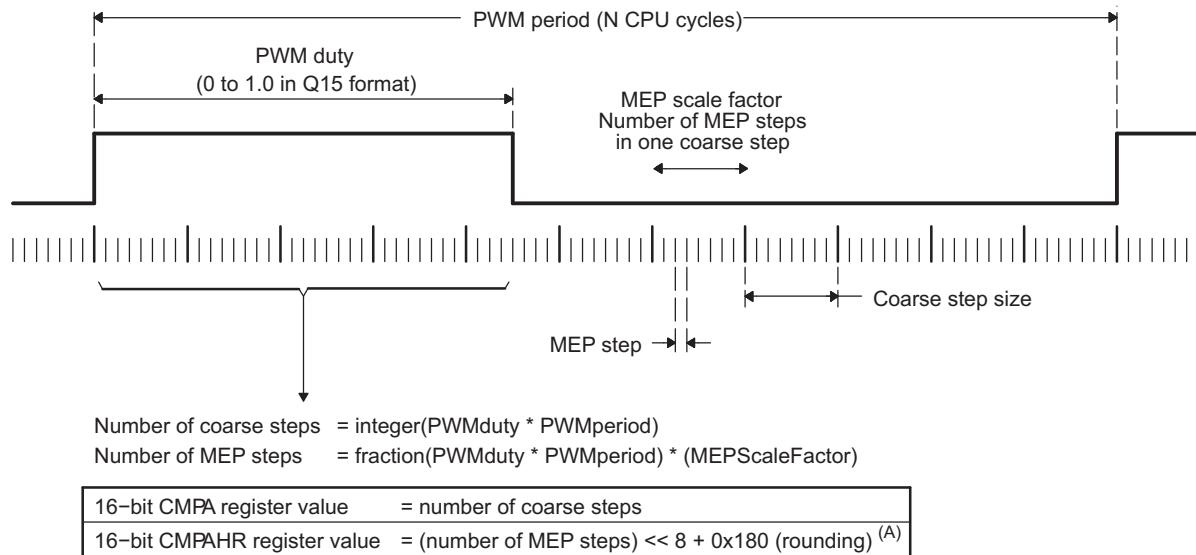
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

### 15.2.2.10.2 Architecture of the High-Resolution PWM Submodule

The HRPWM is based on micro edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by sub-dividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions.

[Figure 15-50](#) shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR).

**Figure 15-50. Operating Logic Using MEP**



A For MEP range and rounding adjustment.

To generate an HRPWM waveform, configure the TBM, CCM, and AQM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the TBM, CCM, and AQM registers to extend edge resolution, and should be configured accordingly. Although many programming combinations are possible, only a few are needed and practical.

### 15.2.2.10.3 Controlling and Monitoring the High-Resolution PWM Submodule

The MEP of the HRPWM is controlled by two extension registers, each 8-bits wide. These two HRPWM registers are concatenated with the 16-bit TBPHS and CMPA registers used to control PWM operation.

- TBPHSHR - Time-Base Phase High-Resolution Register
- CMPAHR - Counter-Compare A High-Resolution Register

[Table 15-40](#) lists the registers used to control and monitor the high-resolution PWM submodule.

**Table 15-40. HRPWM Submodule Registers**

| Acronym | Register Description               | Address Offset | Shadowed |
|---------|------------------------------------|----------------|----------|
| TBPHSHR | Extension Register for HRPWM Phase | 4h             | No       |
| CMPAHR  | Extension Register for HRPWM Duty  | 10h            | Yes      |
| HRCNFG  | HRPWM Configuration Register       | 1040h          | No       |

#### 15.2.2.10.4 Configuring the High-Resolution PWM Submodule

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register located at offset address 1040h. This register provides configuration options for the following key operating modes:

- **Edge Mode:** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE), or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control, while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge.
- **Control Mode:** The MEP is programmed to be controlled either from the CMPAHR register (duty cycle control) or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR register. BE control mode should be used with TBPHSHR register.
- **Shadow Mode:** This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR register and should be chosen to be the same as the regular load option for the CMPA register. If TBPHSHR is used, then this option has no effect.

#### 15.2.2.10.5 Operational Highlights for the High-Resolution PWM Submodule

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps, each of which has a time resolution on the order of 150 ps. The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. [Table 15-41](#) shows the typical range of operating frequencies supported by the HRPWM.

**Table 15-41. Relationship Between MEP Steps, PWM Frequency and Resolution**

| System (MHz) | MEP Steps Per SYSCLKOUT <sup>(1)</sup> <sup>(2)</sup> <sup>(3)</sup> | PWM Minimum (Hz) <sup>(4)</sup> | PWM Maximum (MHz) | Resolution at Maximum (Bits) <sup>(5)</sup> |
|--------------|--|---------------------------------|-------------------|---|
| 50.0         | 111  | 763                             | 2.50              | 11.1  |
| 60.0         | 93   | 916                             | 3.00              | 10.9  |
| 70.0         | 79   | 1068                            | 3.50              | 10.6  |
| 80.0         | 69   | 1221                            | 4.00              | 10.4  |
| 90.0         | 62   | 1373                            | 4.50              | 10.3  |
| 100.0        | 56   | 1526                            | 5.00              | 10.1  |

<sup>(1)</sup> System frequency = SYSCLKOUT, that is, CPU clock. TBCLK = SYSCLKOUT

<sup>(2)</sup> Table data based on a MEP time resolution of 180 ps (this is an example value)

<sup>(3)</sup> MEP steps applied =  $T_{SYSCLKOUT}/180$  ps in this example.

<sup>(4)</sup> PWM minimum frequency is based on a maximum period value, TBPRD = 65 535. PWM mode is asymmetrical up-count.

<sup>(5)</sup> Resolution in bits is given for the maximum PWM frequency stated.

### 15.2.2.10.5.1 Edge Positioning

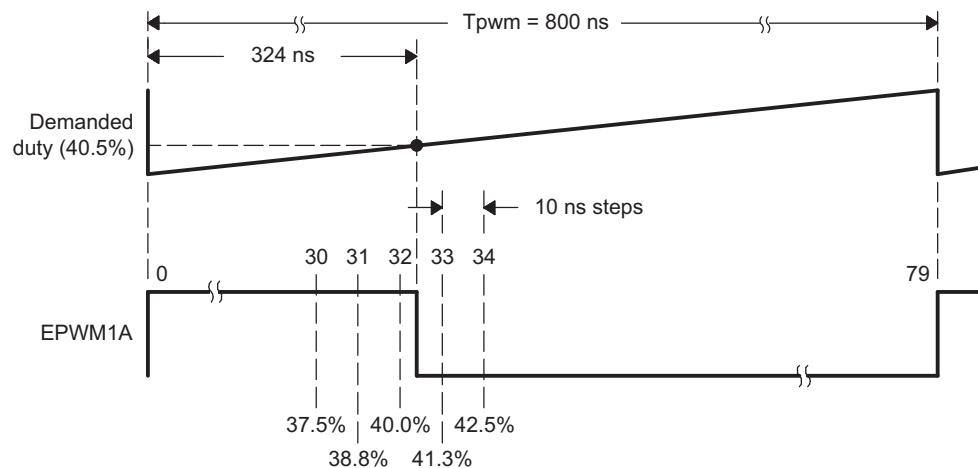
In a typical power control loop (switch modes, digital motor control (DMC), uninterruptible power supply (UPS)), a digital controller (PID, 2pole/2zero, lag/lead, etc.) issues a duty command, usually expressed in per unit or percentage terms.

In the following example, assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on-time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. In [Figure 15-51](#), a compare value of 32 counts (duty = 40%) is the closest to 40.5% that you can attain. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in [Table 15-42](#).

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns.

[Table 15-42](#) shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ns.

**Figure 15-51. Required PWM Waveform for a Requested Duty = 40.5%**



**Table 15-42. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

| CMPA<br>(count)<br><small>(1) (2) (3)</small> | DUTY<br>(%) | High Time<br>(ns) | CMPA<br>(count) | CMPAHR<br>(count) | Duty<br>(%) | High Time<br>(ns) |
|---|-------------|-------------------|-----------------|-------------------|-------------|-------------------|
| 28  | 35.0        | 280               | 32              | 18                | 40.405      | 323.24            |
| 29  | 36.3        | 290               | 32              | 19                | 40.428      | 323.42            |
| 30  | 37.5        | 300               | 32              | 20                | 40.450      | 323.60            |
| 31  | 38.8        | 310               | 32              | 21                | 40.473      | 323.78            |
| 32  | 40.0        | 320               | 32              | 22                | 40.495      | 323.96            |
| 33  | 41.3        | 330               | 32              | 23                | 40.518      | 324.14            |
| 34  | 42.5        | 340               | 32              | 24                | 40.540      | 324.32            |
|   |             |                   | 32              | 25                | 40.563      | 324.50            |
| Required                                      |             |                   | 32              | 26                | 40.585      | 324.68            |
| 32.40   | 40.5        | 324               | 32              | 27                | 40.608      | 324.86            |

<sup>(1)</sup> System clock, SYSCLKOUT and TBCLK = 100 MHz, 10 ns

<sup>(2)</sup> For a PWM Period register value of 80 counts, PWM Period =  $80 \times 10 \text{ ns} = 800 \text{ ns}$ , PWM frequency =  $1/800 \text{ ns} = 1.25 \text{ MHz}$

<sup>(3)</sup> Assumed MEP step size for the above example = 180 ps

### 15.2.2.10.5.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard (CMPA) and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

Assumptions for this example:

|   |                       |
|---|-----------------------|
| System clock, SYSCLKOUT   | = 10 ns (100 MHz)     |
| PWM frequency   | = 1.25 MHz (1/800 ns) |
| Required PWM duty cycle, <b>PWMDuty</b>   | = 0.405 (40.5%)       |
| PWM period in terms of coarse steps,<br><b>PWMperiod</b> (800 ns/10 ns)                               | = 80                  |
| Number of MEP steps per coarse step at<br>180 ps (10 ns/180 ps), <b>MEP_SF</b>                        | = 55                  |
| Value to keep CMPAHR within the range<br>of 1-255 and fractional rounding constant<br>(default value) | = 180h                |

#### Step 1: Percentage Integer Duty value conversion for CMPA register

|                     |  |
|---------------------|--|
| CMPA register value | = int( <b>PWMDuty</b> × <b>PWMperiod</b> ); int means integer part |
|                     | = int(0.405 × 80)  |
|                     | = int(32.4)  |
| CMPA register value | = 32 (20h)   |

#### Step 2: Fractional value conversion for CMPAHR register

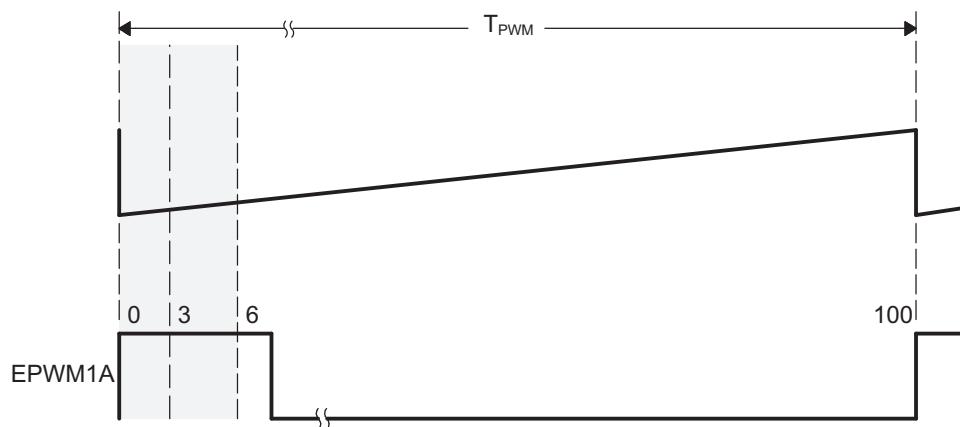
|                       |   |
|-----------------------|---|
| CMPAHR register value | = (frac( <b>PWMDuty</b> × <b>PWMperiod</b> ) × <b>MEP_SF</b> ) << 8 +<br>180h; frac means fractional part |
|                       | = (frac(32.4) × 55 <<8) + 180h; Shift is to move the<br>value as CMPAHR high byte                         |
|                       | = ((0.4 × 55) <<8) + 180h   |
|                       | = (22 <<8) + 180h   |
|                       | = 22 × 256 + 180h ; Shifting left by 8 is the same<br>multiplying by 256.                                 |
|                       | = 5632 + 180h   |
|                       | = 1600h + 180h  |
| CMPAHR value          | = 1780h; CMPAHR value = 1700h, lower 8 bits will be<br>ignored by hardware.                               |

### 15.2.2.10.5.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational 3 SYSCLK cycles after the period starts.

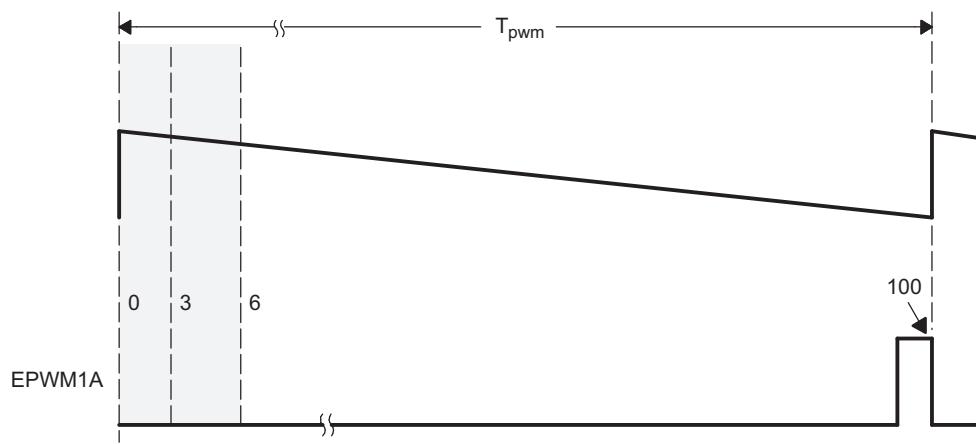
Duty cycle range limitations are illustrated in [Figure 15-52](#). This limitation imposes a lower duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. Although for the first 3 or 6 cycles, the HRPWM capabilities are not available, regular PWM duty control is still fully operational down to 0% duty. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle.

**Figure 15-52. Low % Duty Cycle Range Limitation Example When PWM Frequency = 1 MHz**



If the application demands HRPWM operation in the low percent duty cycle region, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP. This is illustrated in [Figure 15-53](#). In this case low percent duty limitation is no longer an issue.

**Figure 15-53. High % Duty Cycle Range Limitation Example when PWM Frequency = 1 MHz**



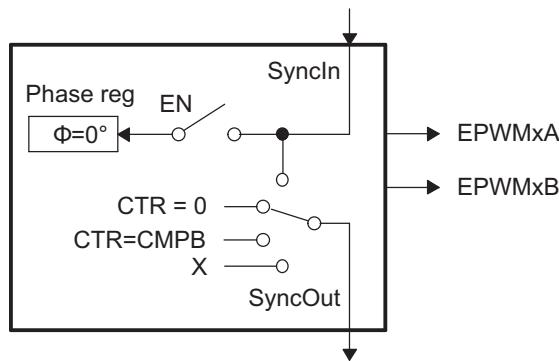
### 15.2.3 Use Cases

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

#### 15.2.3.1 Overview of Multiple Modules

Previously in this user's guide, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in [Figure 15-54](#). This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

**Figure 15-54. Simplified ePWM Module**



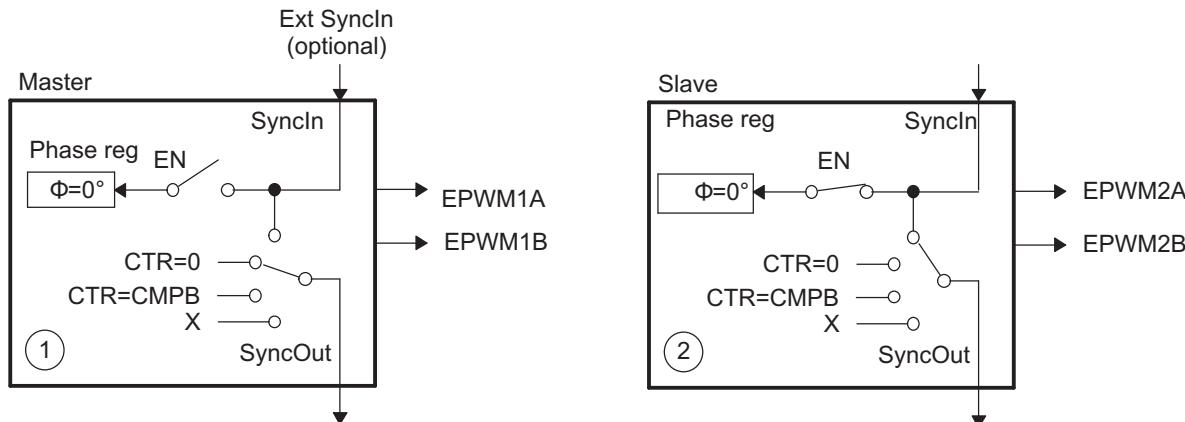
### 15.2.3.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it, i.e., via the enable switch. Although various combinations are possible, the two most common—master module and slave module modes—are shown in [Figure 15-55](#).

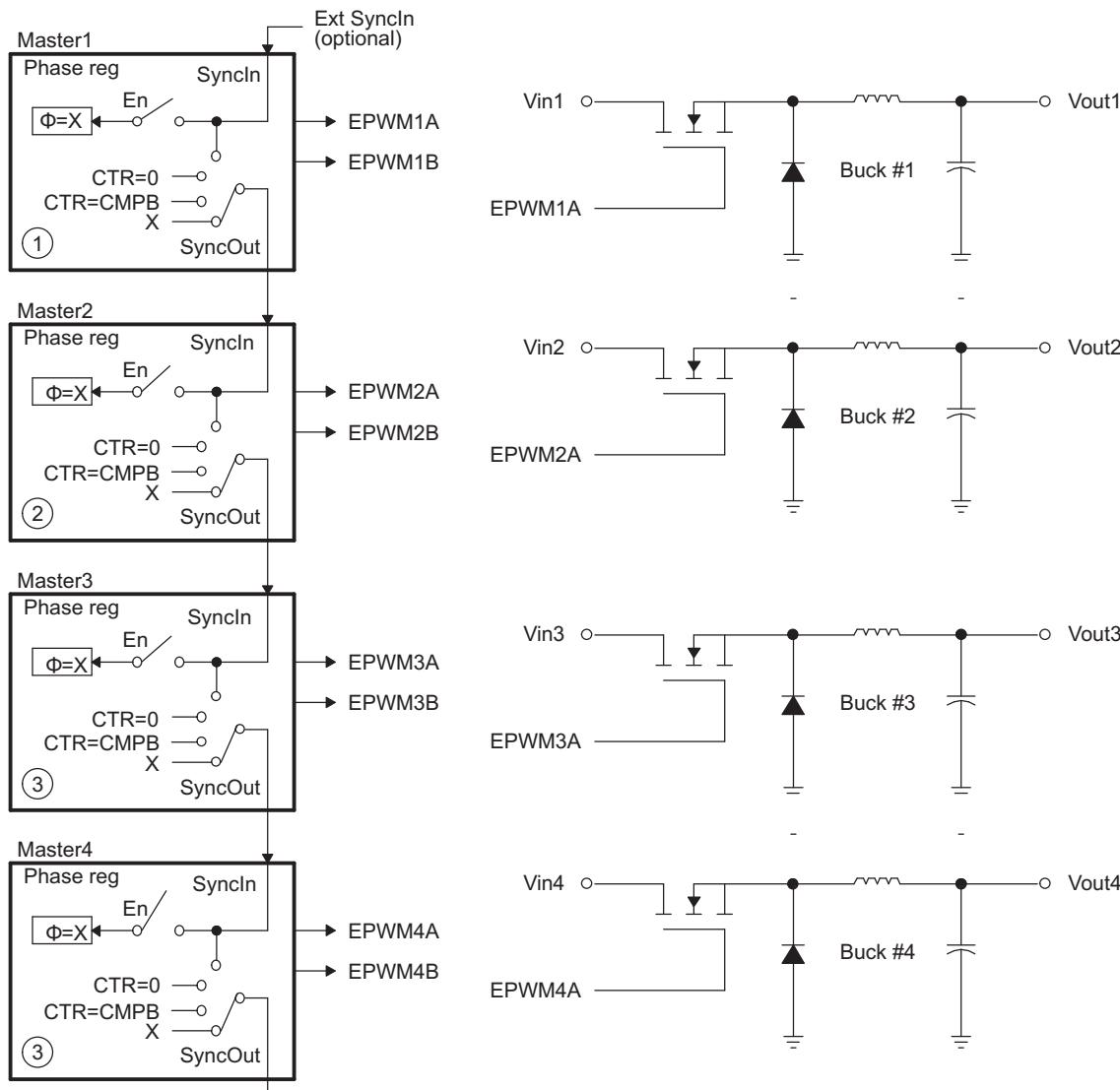
**Figure 15-55. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave**



### 15.2.3.3 Controlling Multiple Buck Converters With Independent Frequencies

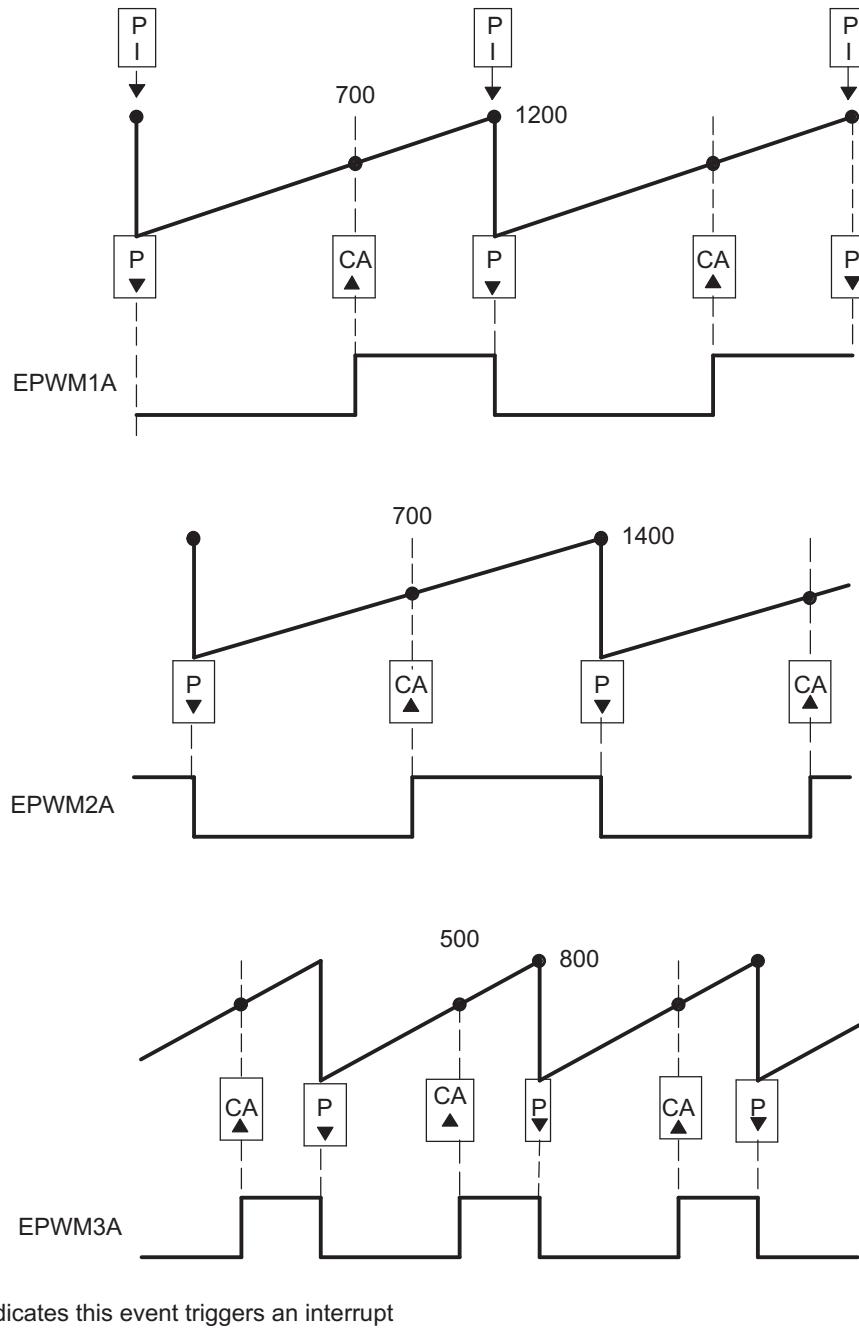
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. [Figure 15-56](#) shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. [Figure 15-57](#) shows the waveforms generated by the setup shown in [Figure 15-56](#); note that only three waveforms are shown, although there are four stages.

**Figure 15-56. Control of Four Buck Stages. Here  $F_{\text{PWM}1} \neq F_{\text{PWM}2} \neq F_{\text{PWM}3} \neq F_{\text{PWM}4}$**



NOTE:  $\Theta = X$  indicates value in phase register is a "don't care"

**Figure 15-57. Buck Waveforms for Figure 15-56 (Note: Only three bucks shown here)**



**Table 15-43. EPWM1 Initialization for Figure 15-57**

| Register | Bit       | Value           | Comments                   |
|----------|-----------|-----------------|----------------------------|
| TBPRD    | TBPRD     | 1200 (4B0h)     | Period = 1201 TBCLK counts |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UP           |                            |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled     |
|          | PRDLD     | TB_SHADOW       |                            |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                            |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                            |
|          | SHDWBMODE | CC_SHADOW       |                            |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0            |
| AQCTLA   | PRD       | AQ_CLEAR        |                            |
|          | CAU       | AQ_SET          |                            |

**Table 15-44. EPWM2 Initialization for Figure 15-57**

| Register | Bit       | Value           | Comments                   |
|----------|-----------|-----------------|----------------------------|
| TBPRD    | TBPRD     | 1400 (578h)     | Period = 1401 TBCLK counts |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UP           |                            |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled     |
|          | PRDLD     | TB_SHADOW       |                            |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                            |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                            |
|          | SHDWBMODE | CC_SHADOW       |                            |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0            |
| AQCTLA   | PRD       | AQ_CLEAR        |                            |
|          | CAU       | AQ_SET          |                            |

**Table 15-45. EPWM3 Initialization for Figure 15-57**

| Register | Bit       | Value           | Comments                  |
|----------|-----------|-----------------|---------------------------|
| TBPRD    | TBPRD     | 800 (320h)      | Period = 801 TBCLK counts |
| TBPHS    | TBPHS     | 0               | Clear Phase Register to 0 |
| TBCTL    | CTRMODE   | TB_UP           |                           |
|          | PHSEN     | TB_DISABLE      | Phase loading disabled    |
|          | PRDLD     | TB_SHADOW       |                           |
|          | SYNCOSEL  | TB_SYNC_DISABLE |                           |
| CMPCTL   | SHDWAMODE | CC_SHADOW       |                           |
|          | SHDWBMODE | CC_SHADOW       |                           |
|          | LOADAMODE | CC_CTR_ZERO     | Load on CTR = 0           |
|          | LOADBMODE | CC_CTR_ZERO     | Load on CTR = 0           |
| AQCTLA   | PRD       | AQ_CLEAR        |                           |
|          | CAU       | AQ_SET          |                           |

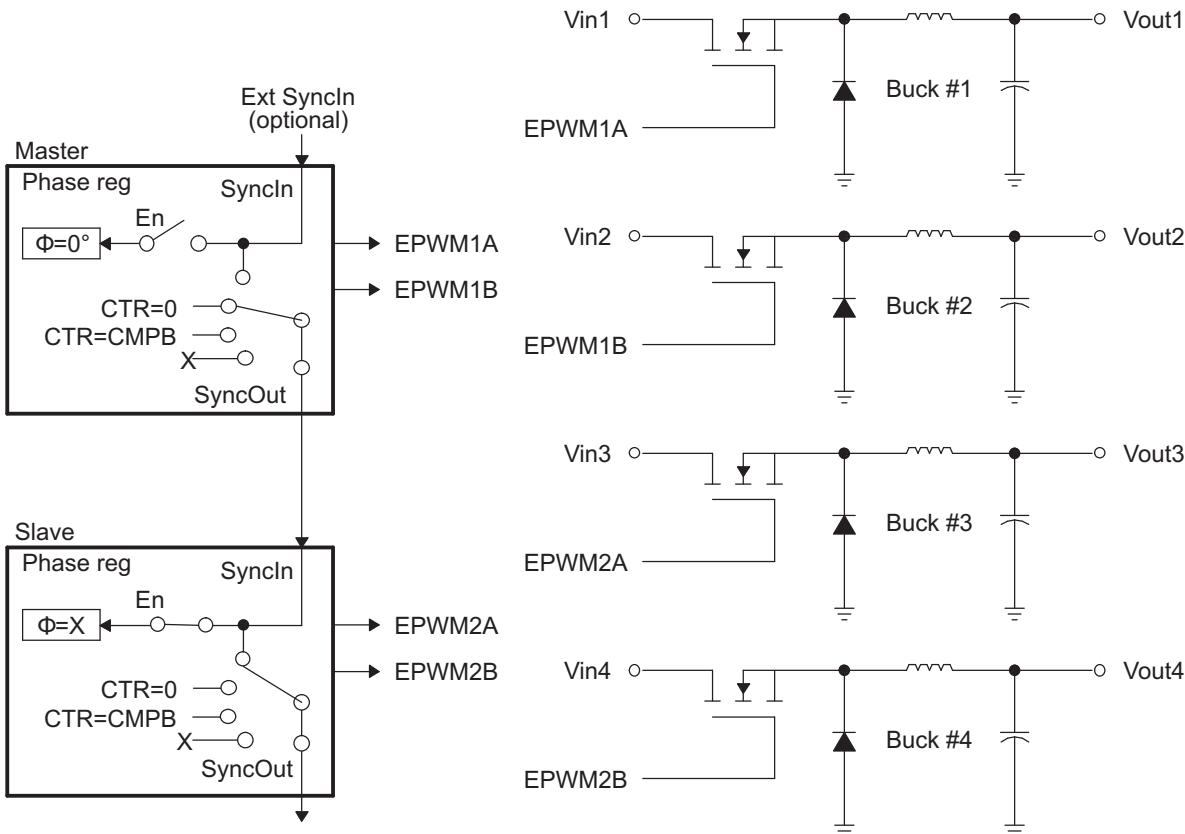
**Example 15-3. Configuration for Example in Figure 15-57**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 700;           // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 700;           // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 500;           // adjust duty for output EPWM3A
```

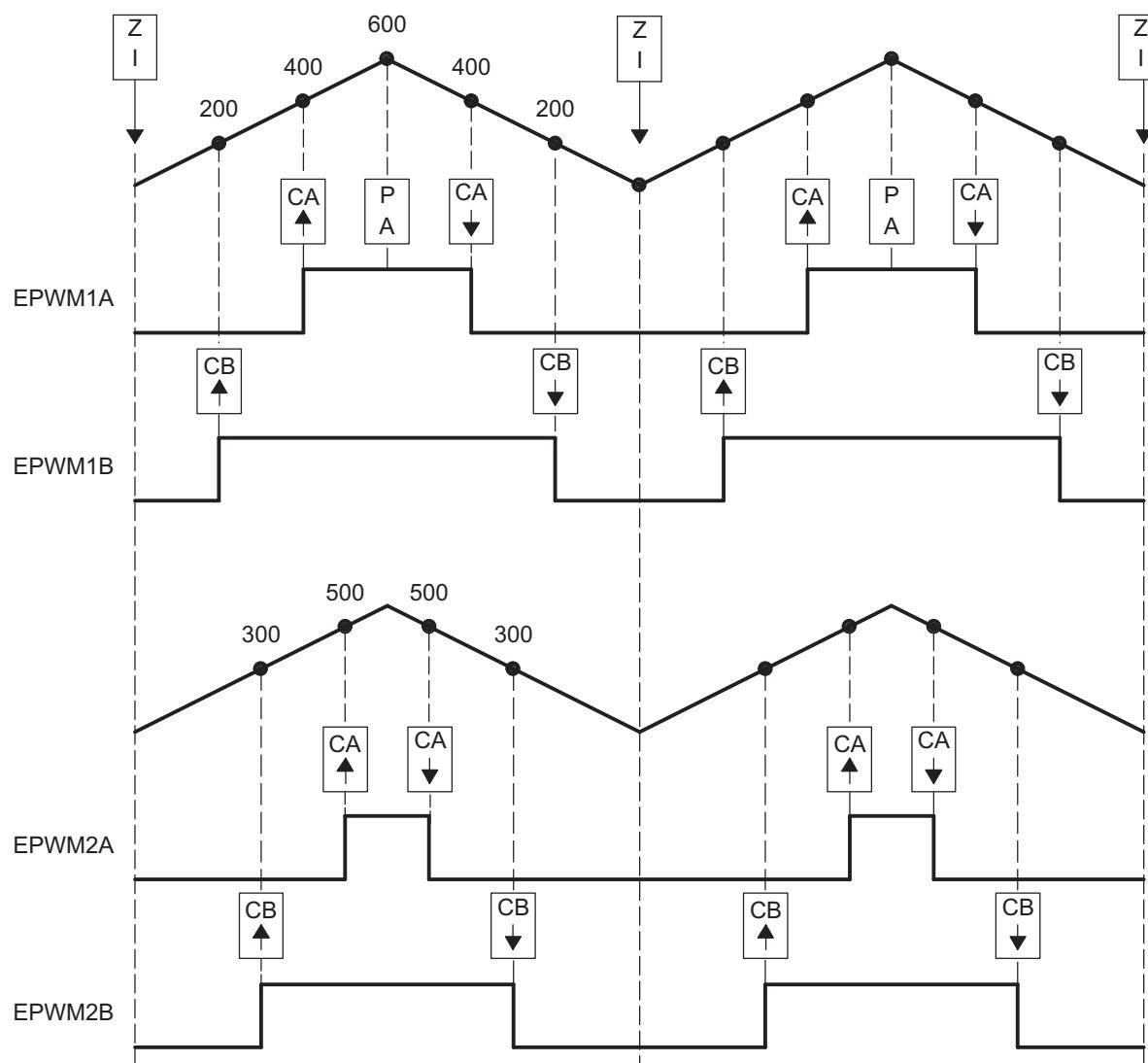
**15.2.3.4 Controlling Multiple Buck Converters With Same Frequencies**

If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. [Figure 15-58](#) shows such a configuration; [Figure 15-59](#) shows the waveforms generated by the configuration.

**Figure 15-58. Control of Four Buck Stages. (Note:  $F_{\text{PWM2}} = N \times F_{\text{PWM1}}$ )**



**Figure 15-59. Buck Waveforms for Figure 15-58 (Note:  $F_{\text{PWM}2} = F_{\text{PWM}1}$ )**



**Table 15-46. EPWM1 Initialization for Figure 15-58**

| Register | Bit       | Value       | Comments                   |
|----------|-----------|-------------|----------------------------|
| TBPRD    | TBPRD     | 600 (258h)  | Period = 1200 TBCLK counts |
| TBPHS    | TBPHS     | 0           | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN   |                            |
|          | PHSEN     | TB_DISABLE  | Phase loading disabled     |
|          | PRDLDD    | TB_SHADOW   |                            |
|          | SYNCOSEL  | TB_CTR_ZERO | Sync down-stream module    |
| CMPCTL   | SHDWAMODE | CC_SHADOW   |                            |
|          | SHDWBMODE | CC_SHADOW   |                            |
|          | LOADAMODE | CC_CTR_ZERO | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO | Load on CTR = 0            |
| AQCTLA   | CAU       | AQ_SET      | Set actions for EPWM1A     |
|          | CAD       | AQ_CLEAR    |                            |
| AQCTLB   | CBU       | AQ_SET      | Set actions for EPWM1B     |
|          | CBD       | AQ_CLEAR    |                            |

**Table 15-47. EPWM2 Initialization for Figure 15-58**

| Register | Bit       | Value       | Comments                   |
|----------|-----------|-------------|----------------------------|
| TBPRD    | TBPRD     | 600 (258h)  | Period = 1200 TBCLK counts |
| TBPHS    | TBPHS     | 0           | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN   |                            |
|          | PHSEN     | TB_ENABLE   | Phase loading enabled      |
|          | PRDLDD    | TB_SHADOW   |                            |
|          | SYNCOSEL  | TB_SYNC_IN  | Sync flow-through          |
| CMPCTL   | SHDWAMODE | CC_SHADOW   |                            |
|          | SHDWBMODE | CC_SHADOW   |                            |
|          | LOADAMODE | CC_CTR_ZERO | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO | Load on CTR = 0            |
| AQCTLA   | CAU       | AQ_SET      | Set actions for EPWM2A     |
|          | CAD       | AQ_CLEAR    |                            |
| AQCTLB   | CBU       | AQ_SET      | Set actions for EPWM2B     |
|          | CBD       | AQ_CLEAR    |                            |

**Example 15-4. Code Snippet for Configuration in Figure 15-58**

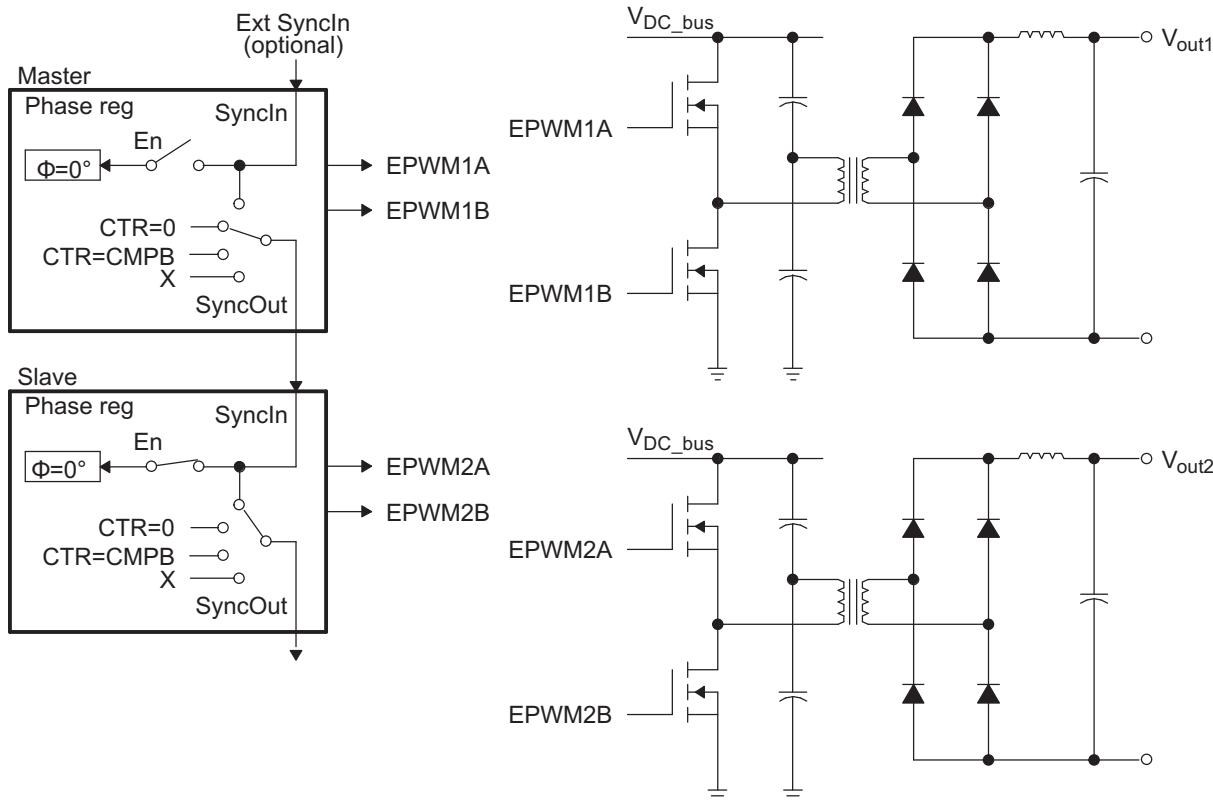
```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 400;           // adjust duty for output EPWM1A
EPwm1Regs.CMPB = 200;                     // adjust duty for output EPWM1B
EPwm2Regs.CMPA.half.CMPA = 500;           // adjust duty for output EPWM2A
EPwm2Regs.CMPB = 300;                     // adjust duty for output EPWM2B
```

### 15.2.3.5 Controlling Multiple Half H-Bridge (HHB) Converters

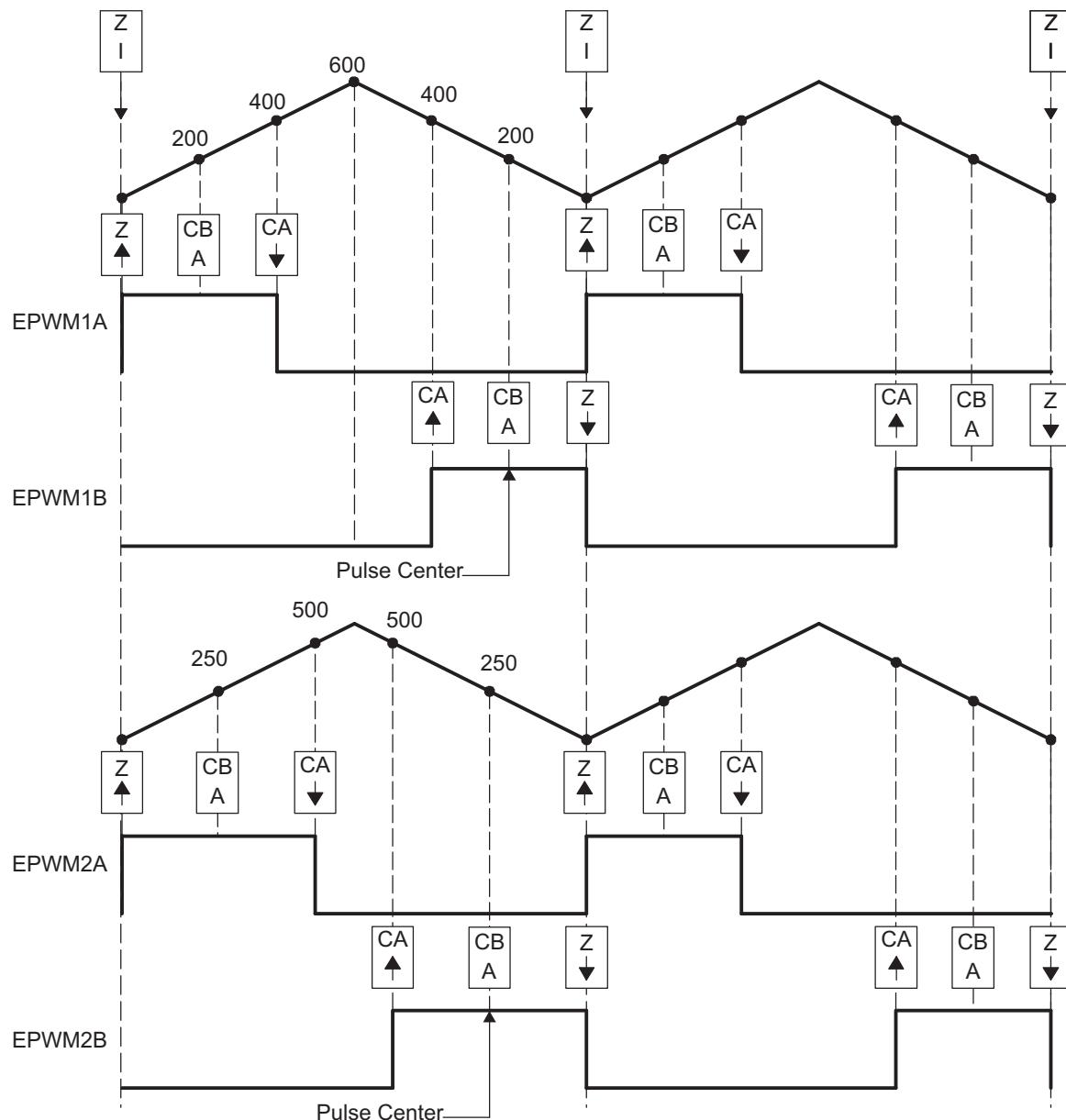
Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 15-60 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 15-61 shows the waveforms generated by the configuration shown in Figure 15-60.

Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

**Figure 15-60. Control of Two Half-H Bridge Stages ( $F_{\text{PWM}2} = N \times F_{\text{PWM}1}$ )**



**Figure 15-61. Half-H Bridge Waveforms for Figure 15-60 (Note: Here  $F_{\text{PWM}2} = F_{\text{PWM}1}$ )**



**Table 15-48. EPWM1 Initialization for Figure 15-60**

| Register | Bit       | Value       | Comments                   |
|----------|-----------|-------------|----------------------------|
| TBPRD    | TBPRD     | 600 (258h)  | Period = 1200 TBCLK counts |
| TBPHS    | TBPHS     | 0           | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN   |                            |
|          | PHSEN     | TB_DISABLE  | Phase loading disabled     |
|          | PRDLDD    | TB_SHADOW   |                            |
|          | SYNCOSEL  | TB_CTR_ZERO | Sync down-stream module    |
| CMPCTL   | SHDWAMODE | CC_SHADOW   |                            |
|          | SHDWBMODE | CC_SHADOW   |                            |
|          | LOADAMODE | CC_CTR_ZERO | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO | Load on CTR = 0            |
| AQCTLA   | ZRO       | AQ_SET      | Set actions for EPWM1A     |
|          | CAU       | AQ_CLEAR    |                            |
| AQCTLB   | ZRO       | AQ_CLEAR    | Set actions for EPWM1B     |
|          | CAD       | AQ_SET      |                            |

**Table 15-49. EPWM2 Initialization for Figure 15-60**

| Register | Bit       | Value       | Comments                   |
|----------|-----------|-------------|----------------------------|
| TBPRD    | TBPRD     | 600 (258h)  | Period = 1200 TBCLK counts |
| TBPHS    | TBPHS     | 0           | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN   |                            |
|          | PHSEN     | TB_ENABLE   | Phase loading enabled      |
|          | PRDLDD    | TB_SHADOW   |                            |
|          | SYNCOSEL  | TB_SYNC_IN  | Sync flow-through          |
| CMPCTL   | SHDWAMODE | CC_SHADOW   |                            |
|          | SHDWBMODE | CC_SHADOW   |                            |
|          | LOADAMODE | CC_CTR_ZERO | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO | Load on CTR = 0            |
| AQCTLA   | ZRO       | AQ_SET      | Set actions for EPWM2A     |
|          | CAU       | AQ_CLEAR    |                            |
| AQCTLB   | ZRO       | AQ_CLEAR    | Set actions for EPWM2B     |
|          | CAD       | AQ_SET      |                            |

**Example 15-5. Code Snippet for Configuration in Figure 15-60**

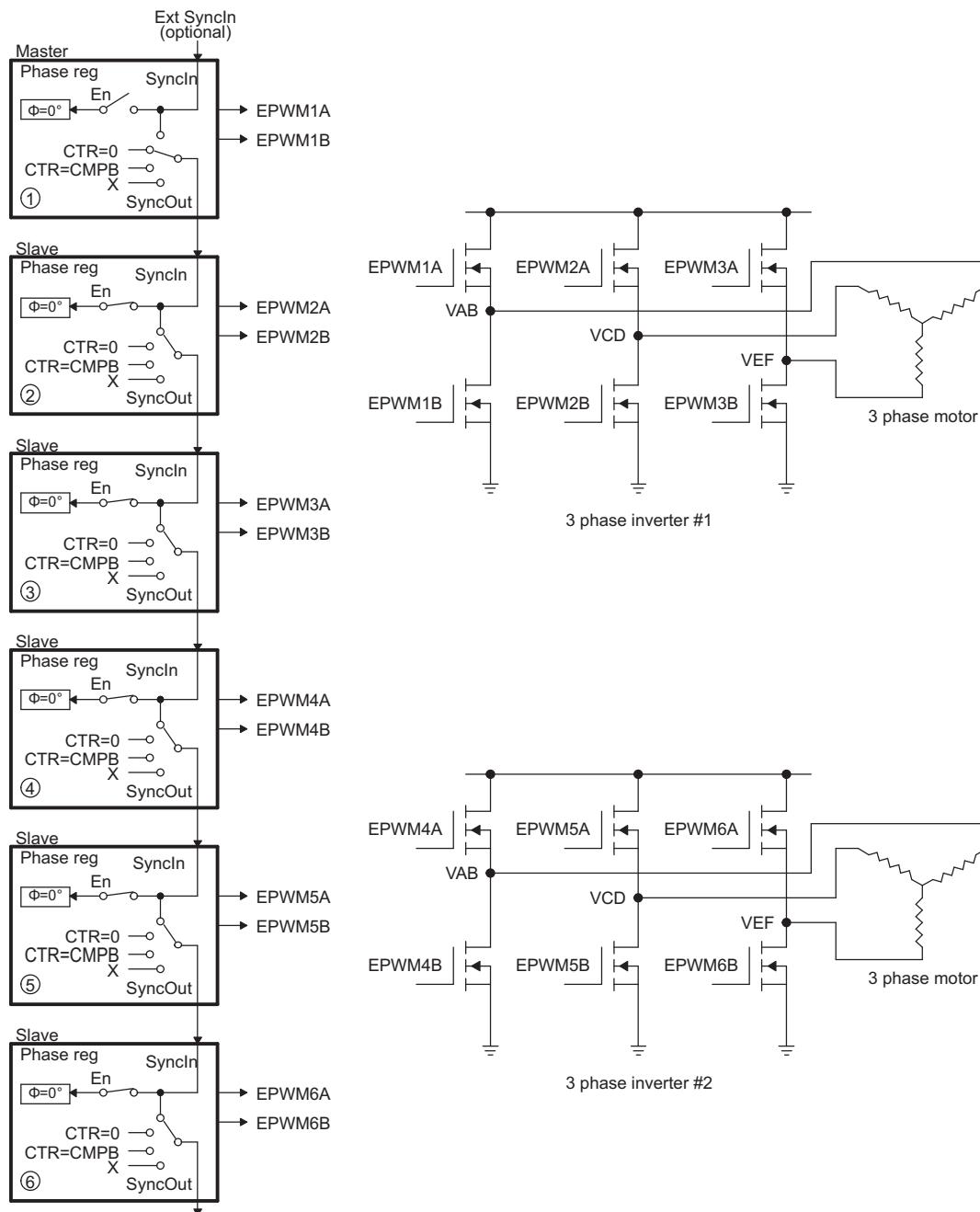
```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 400; // adjust duty for output EPWM1A
EPwm1Regs.CMPB = 200;           // adjust duty for output EPWM1B
EPwm2Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM2A
EPwm2Regs.CMPB = 250;           // adjust duty for output EPWM2B
```

### 15.2.3.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

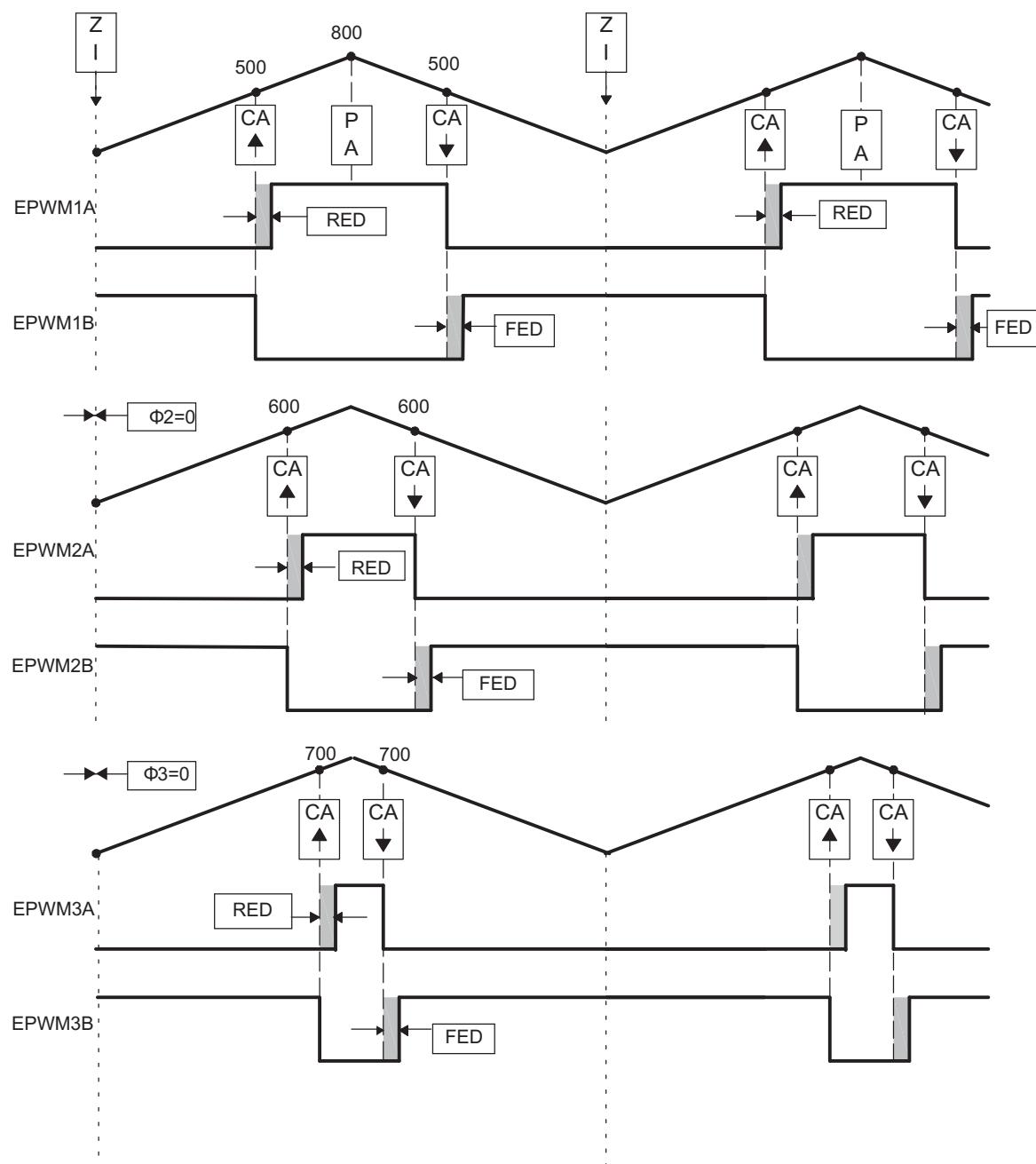
The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. [Figure 15-62](#) shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in [Figure 15-62](#)), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

**Figure 15-62. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control**



**Figure 15-63. 3-Phase Inverter Waveforms for Figure 15-62 (Only One Inverter Shown)**



**Table 15-50. EPWM1 Initialization for Figure 15-62**

| Register | Bit       | Value          | Comments                   |
|----------|-----------|----------------|----------------------------|
| TBPRD    | TBPRD     | 800 (320h)     | Period = 1600 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN      |                            |
|          | PHSEN     | TB_DISABLE     | Phase loading disabled     |
|          | PRDLDD    | TB_SHADOW      |                            |
|          | SYNCOSEL  | TB_CTR_ZERO    | Sync down-stream module    |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                            |
|          | SHDWBMODE | CC_SHADOW      |                            |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0            |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM1A     |
|          | CAD       | AQ_CLEAR       |                            |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary    |
| DBFED    | DBFED     | 50             | FED = 50 TBCLKs            |
|          | DBRED     | 50             | RED = 50 TBCLKs            |

**Table 15-51. EPWM2 Initialization for Figure 15-62**

| Register | Bit       | Value          | Comments                   |
|----------|-----------|----------------|----------------------------|
| TBPRD    | TBPRD     | 800 (320h)     | Period = 1600 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN      |                            |
|          | PHSEN     | TB_ENABLE      | Slave module               |
|          | PRDLDD    | TB_SHADOW      |                            |
|          | SYNCOSEL  | TB_SYNC_IN     | Sync flow-through          |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                            |
|          | SHDWBMODE | CC_SHADOW      |                            |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0            |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM2A     |
|          | CAD       | AQ_CLEAR       |                            |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary    |
| DBFED    | DBFED     | 50             | FED = 50 TBCLKs            |
|          | DBRED     | 50             | RED = 50 TBCLKs            |

**Table 15-52. EPWM3 Initialization for Figure 15-62**

| Register | Bit       | Value          | Comments                   |
|----------|-----------|----------------|----------------------------|
| TBPRD    | TBPRD     | 800 (320h)     | Period = 1600 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UPDOWN      |                            |
|          | PHSEN     | TB_ENABLE      | Slave module               |
|          | PRDLDD    | TB_SHADOW      |                            |
|          | SYNCOSEL  | TB_SYNC_IN     | Sync flow-through          |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                            |
|          | SHDWBMODE | CC_SHADOW      |                            |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0            |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM3A     |
|          | CAD       | AQ_CLEAR       |                            |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary    |
| DBFED    | DBFED     | 50             | FED = 50 TBCLKs            |
|          | DBRED     | 50             | RED = 50 TBCLKs            |

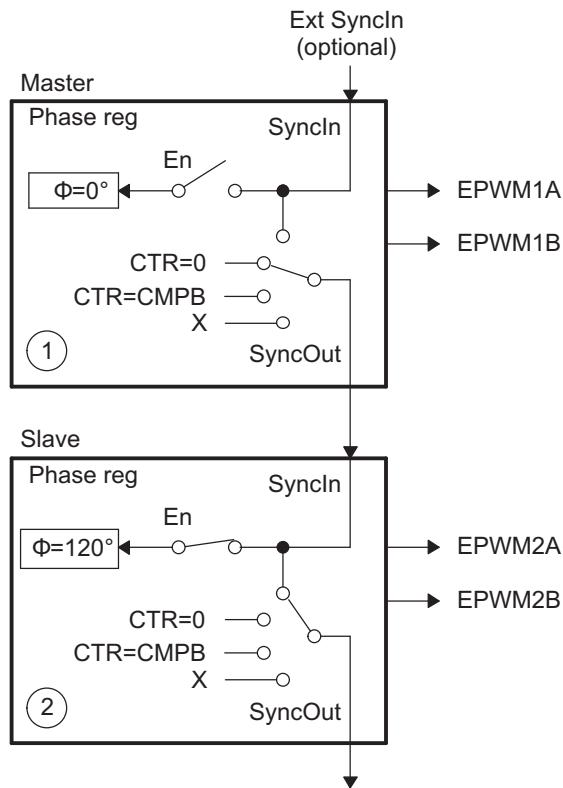
**Example 15-6. Code Snippet for Configuration in Figure 15-62**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 600; // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 700; // adjust duty for output EPWM3A
```

### 15.2.3.7 Practical Applications Using Phase Control Between PWM Modules

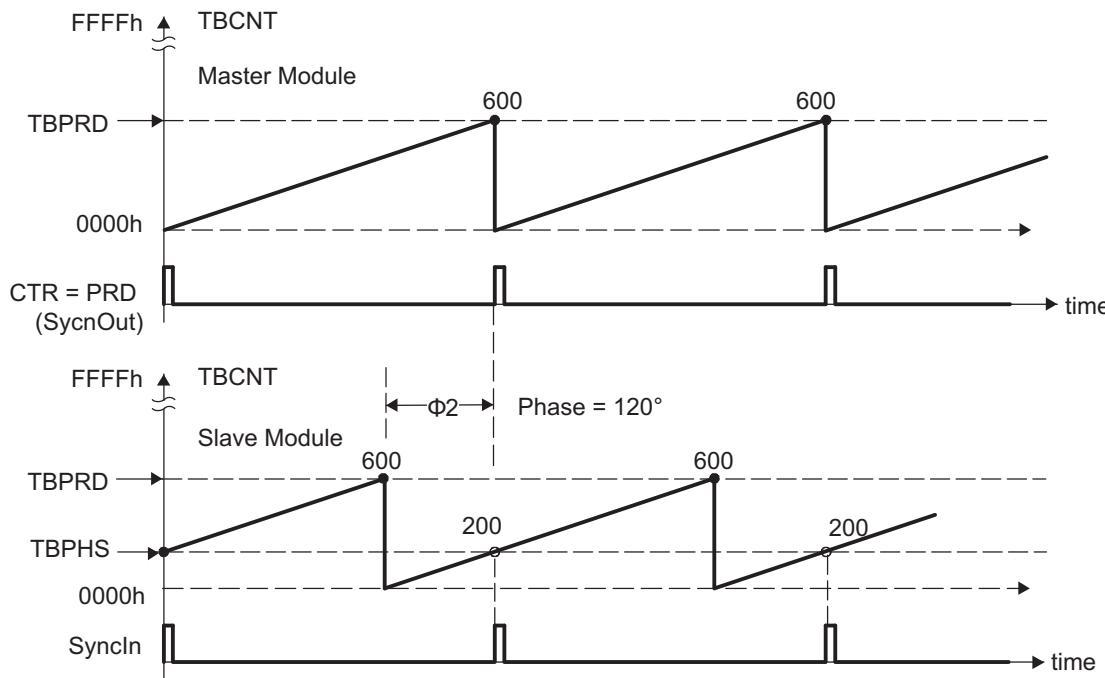
So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCNT register. To illustrate this concept, [Figure 15-64](#) shows a master and slave module with a phase relationship of 120°, that is, the slave leads the master.

**Figure 15-64. Configuring Two PWM Modules for Phase Control**



[Figure 15-65](#) shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 ( $200/600 \times 360^\circ = 120^\circ$ ). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCNT register so the slave time-base is always leading the master's time-base by 120°.

**Figure 15-65. Timing Waveforms Associated With Phase Control Between 2 Modules**



#### 15.2.3.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in [Figure 15-66](#). This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be  $F = 120^\circ$ . This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

$$\text{TBPHS}(N,M) = (\text{TBPRD}/N) \times (M - 1)$$

Where:

N = number of phases

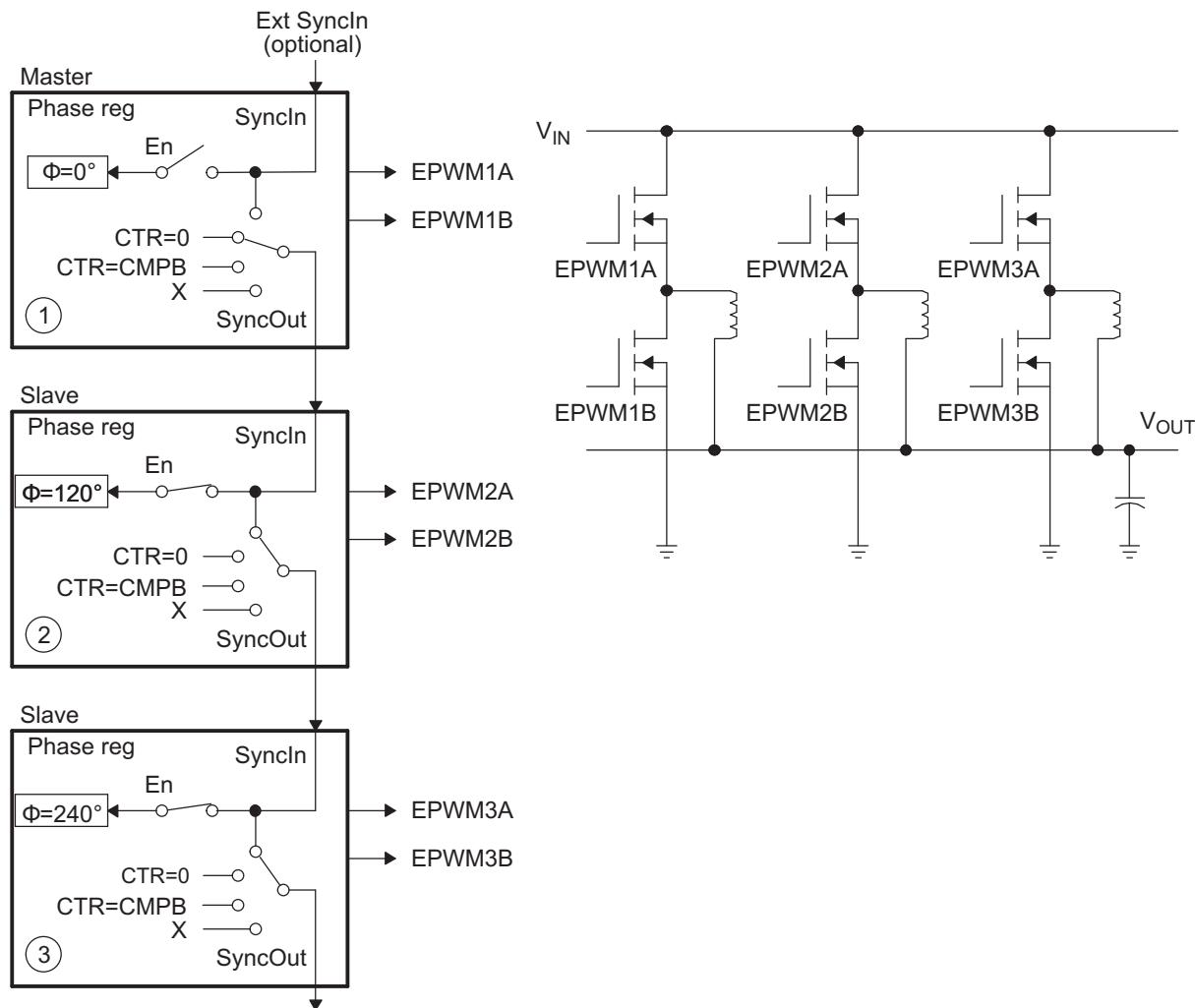
M = PWM module number

For example, for the 3-phase case ( $N = 3$ ),  $\text{TBPRD} = 600$ ,

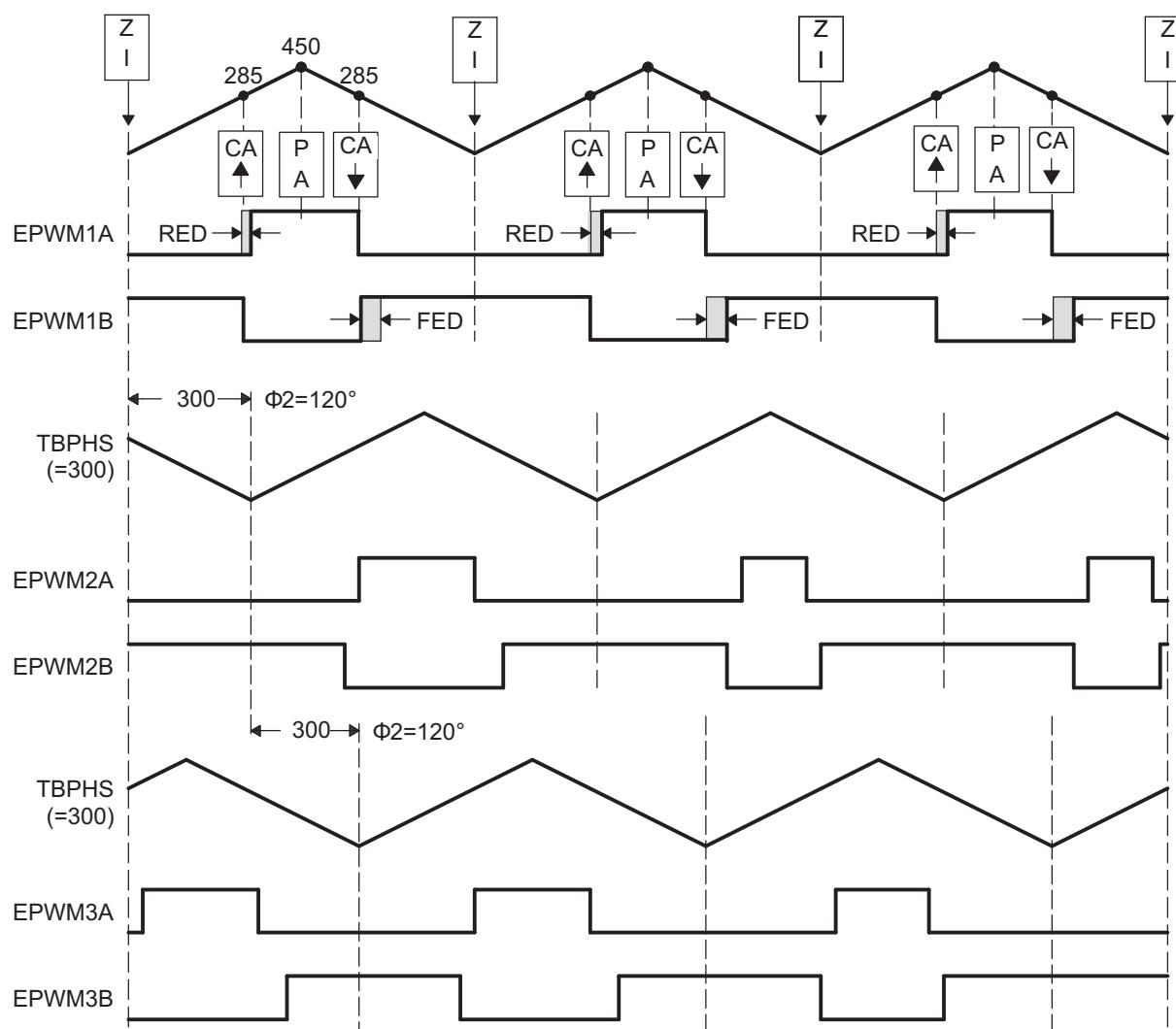
$$\text{TBPHS}(3,2) = (600/3) \times (2 - 1) = 200 \times 1 = 200 \text{ (Phase value for Slave module 2)}$$

$$\text{TBPHS}(3,3) = (600/3) \times (3 - 1) = 200 \times 2 = 400 \text{ (Phase value for Slave module 3)}$$

[Figure 15-67](#) shows the waveforms for the configuration in [Figure 15-66](#).

**Figure 15-66. Control of a 3-Phase Interleaved DC/DC Converter**


**Figure 15-67. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 15-66**



**Table 15-53. EPWM1 Initialization for Figure 15-66**

| Register | Bit       | Value          | Comments                  |
|----------|-----------|----------------|---------------------------|
| TBPRD    | TBPRD     | 450 (1C2h)     | Period = 900 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0 |
| TBCTL    | CTRMODE   | TB_UPDOWN      |                           |
|          | PHSEN     | TB_DISABLE     | Phase loading disabled    |
|          | PRDLDD    | TB_SHADOW      |                           |
|          | SYNCOSEL  | TB_CTR_ZERO    | Sync down-stream module   |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                           |
|          | SHDWBMODE | CC_SHADOW      |                           |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0           |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0           |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM1A    |
|          | CAD       | AQ_CLEAR       |                           |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module   |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary   |
| DBFED    | DBFED     | 20             | FED = 20 TBCLKs           |
|          | DBRED     | 20             | RED = 20 TBCLKs           |

**Table 15-54. EPWM2 Initialization for Figure 15-66**

| Register | Bit       | Value          | Comments                                   |
|----------|-----------|----------------|--|
| TBPRD    | TBPRD     | 450 (1C2h)     | Period = 900 TBCLK counts                  |
| TBPHS    | TBPHS     | 300            | Phase = $(300/900) \times 360 = 120^\circ$ |
| TBCTL    | CTRMODE   | TB_UPDOWN      |  |
|          | PHSEN     | TB_ENABLE      | Slave module                               |
|          | PRDLDD    | TB_SHADOW      |  |
|          | SYNCOSEL  | TB_SYNC_IN     | Sync flow-through                          |
|          | PHSDIR    | TB_DOWN        | Count DOWN on sync                         |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |  |
|          | SHDWBMODE | CC_SHADOW      |  |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0                            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0                            |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM2A                     |
|          | CAD       | AQ_CLEAR       |  |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module                    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary                    |
| DBFED    | DBFED     | 20             | FED = 20 TBCLKs                            |
|          | DBRED     | 20             | RED = 20 TBCLKs                            |

**Table 15-55. EPWM3 Initialization for Figure 15-66**

| Register | Bit       | Value          | Comments                       |
|----------|-----------|----------------|--------------------------------|
| TBPRD    | TBPRD     | 450 (1C2h)     | Period = 900 TBCLK counts      |
| TBPHS    | TBPHS     | 300            | Phase = (300/900) × 360 = 120° |
| TBCTL    | CTRMODE   | TB_UPDOWN      |                                |
|          | PHSEN     | TB_ENABLE      | Slave module                   |
|          | PRDLDD    | TB_SHADOW      |                                |
|          | SYNCOSEL  | TB_SYNC_IN     | Sync flow-through              |
|          | PHSDIR    | TB_UP          | Count UP on sync               |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                                |
|          | SHDWBMODE | CC_SHADOW      |                                |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0                |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0                |
| AQCTLA   | CAU       | AQ_SET         | Set actions for EPWM3A         |
|          | CAD       | AQ_CLEAR       |                                |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module        |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary        |
| DBFED    | DBFED     | 20             | FED = 20 TBCLKs                |
|          | DBRED     | 20             | RED = 20 TBCLKs                |

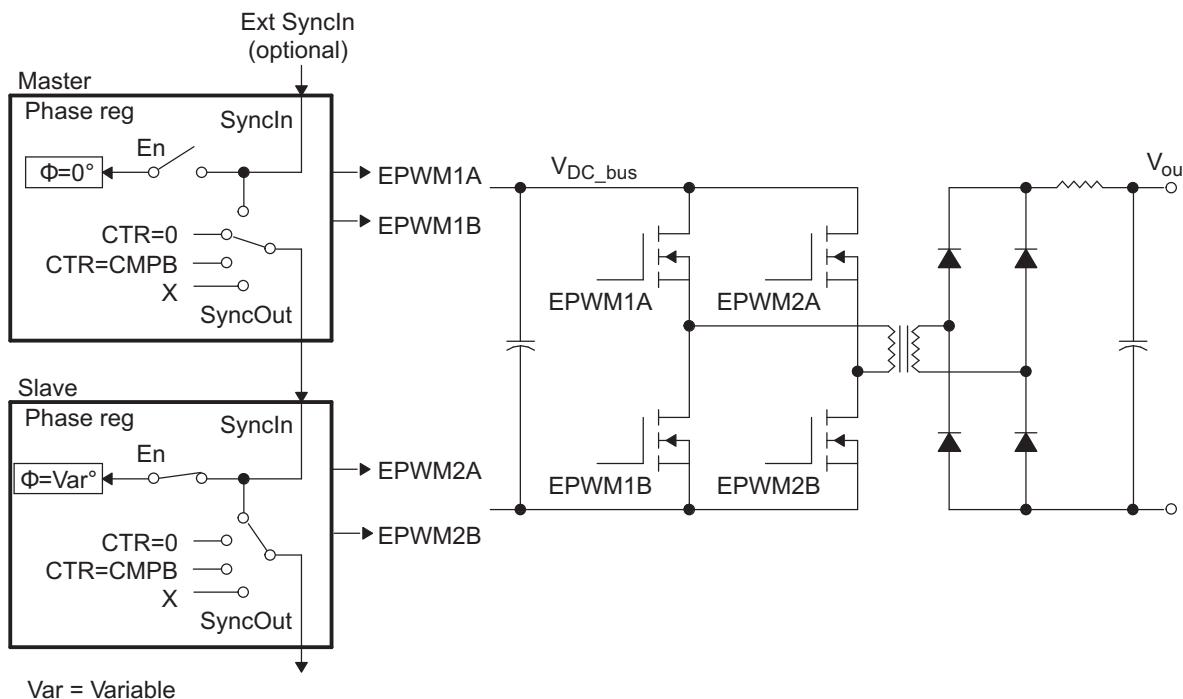
**Example 15-7. Code Snippet for Configuration in Figure 15-66**

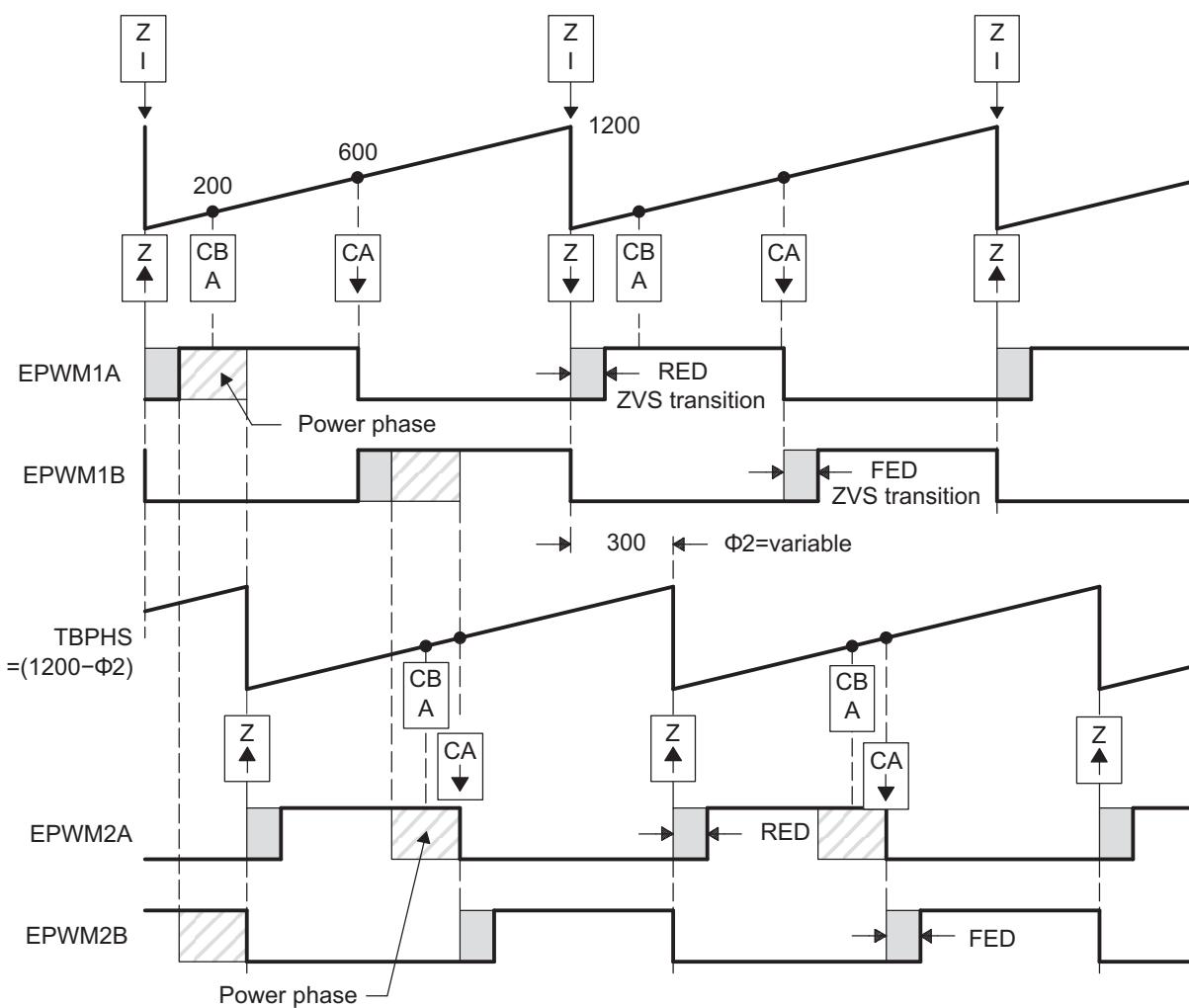
```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm1Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 285;           // adjust duty for output EPWM3A
```

### 15.2.3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in [Figure 15-68](#) assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge*. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. [Figure 15-69](#) shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

**Figure 15-68. Controlling a Full-H Bridge Stage ( $F_{\text{PWM}2} = F_{\text{PWM}1}$ )**



**Figure 15-69. ZVS Full-H Bridge Waveforms**


**Table 15-56. EPWM1 Initialization for Figure 15-68**

| Register | Bit       | Value          | Comments                   |
|----------|-----------|----------------|----------------------------|
| TBPRD    | TBPRD     | 1200 (4B0h)    | Period = 1201 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UP          |                            |
|          | PHSEN     | TB_DISABLE     | Phase loading disabled     |
|          | PRDLDD    | TB_SHADOW      |                            |
|          | SYNCOSEL  | TB_CTR_ZERO    | Sync down-stream module    |
| CMPA     | CMPA      | 600 (258h)     | Set 50% duty for EPWM1A    |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                            |
|          | SHDWBMODE | CC_SHADOW      |                            |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0            |
| AQCTLA   | ZRO       | AQ_SET         | Set actions for EPWM1A     |
|          | CAU       | AQ_CLEAR       |                            |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary    |
| DBFED    | DBFED     | 50             | FED = 50 TBCLKs            |
|          | DBRED     | 70             | RED = 70 TBCLKs            |

**Table 15-57. EPWM2 Initialization for Figure 15-68**

| Register | Bit       | Value          | Comments                   |
|----------|-----------|----------------|----------------------------|
| TBPRD    | TBPRD     | 1200 (4B0h)    | Period = 1201 TBCLK counts |
| TBPHS    | TBPHS     | 0              | Clear Phase Register to 0  |
| TBCTL    | CTRMODE   | TB_UP          |                            |
|          | PHSEN     | TB_ENABLE      | Slave module               |
|          | PRDLDD    | TB_SHADOW      |                            |
|          | SYNCOSEL  | TB_SYNC_IN     | Sync flow-through          |
| CMPA     | CMPA      | 600 (258h)     | Set 50% duty for EPWM2A    |
| CMPCTL   | SHDWAMODE | CC_SHADOW      |                            |
|          | SHDWBMODE | CC_SHADOW      |                            |
|          | LOADAMODE | CC_CTR_ZERO    | Load on CTR = 0            |
|          | LOADBMODE | CC_CTR_ZERO    | Load on CTR = 0            |
| AQCTLA   | ZRO       | AQ_SET         | Set actions for EPWM2A     |
|          | CAU       | AQ_CLEAR       |                            |
| DBCTL    | MODE      | DB_FULL_ENABLE | Enable Dead-band module    |
|          | POLSEL    | DB_ACTV_HIC    | Active Hi complementary    |
| DBFED    | DBFED     | 30             | FED = 30 TBCLKs            |
|          | DBRED     | 40             | RED = 40 TBCLKs            |

**Example 15-8. Code Snippet for Configuration in Figure 15-68**

```
// Run Time (Note: Example execution of one run-time instance)
//=====
EPwm2Regs.TBPHS = 1200-300; // Set Phase reg to 300/1200 * 360 = 90 deg
EPwm1Regs.DBFED = FED1_NewValue; // Update ZVS transition interval
EPwm1Regs.DBRED = RED1_NewValue; // Update ZVS transition interval
EPwm2Regs.DBFED = FED2_NewValue; // Update ZVS transition interval
EPwm2Regs.DBRED = RED2_NewValue; // Update ZVS transition interval
```

## 15.2.4 Registers

**Table 15-58. ePWM Module Control and Status Registers Grouped by Submodule**

| Acronym   | Offset <sup>(1)</sup> | Size<br>(x16) | Shadow | Register Description  |
|---|-----------------------|---------------|--------|---|
| <b>Time-Base Submodule Registers (Section 15.2.4.1)</b>                   |                       |               |        |   |
| TBCTL   | 0h                    | 1             | No     | Time-Base Control Register                                    |
| TBSTS   | 2h                    | 1             | No     | Time-Base Status Register                                     |
| TBPHSHR   | 4h                    | 1             | No     | Extension for HRPWM Phase Register <sup>(2)</sup>             |
| TBPHS   | 6h                    | 1             | No     | Time-Base Phase Register                                      |
| TBCNT   | 8h                    | 1             | No     | Time-Base Counter Register                                    |
| TBPRD   | Ah                    | 1             | Yes    | Time-Base Period Register                                     |
| <b>Counter-Compare Submodule Registers (Section 15.2.4.2)</b>             |                       |               |        |   |
| CMPCTL  | Eh                    | 1             | No     | Counter-Compare Control Register                              |
| CMPAHR  | 10h                   | 1             | No     | Extension for HRPWM Counter-Compare A Register <sup>(2)</sup> |
| CMPA  | 12h                   | 1             | Yes    | Counter-Compare A Register                                    |
| CMPB  | 14h                   | 1             | Yes    | Counter-Compare B Register                                    |
| <b>Action-Qualifier Submodule Registers (Section 15.2.4.3)</b>            |                       |               |        |   |
| AQCTLA  | 16h                   | 1             | No     | Action-Qualifier Control Register for Output A (EPWMxA)       |
| AQCTLB  | 18h                   | 1             | No     | Action-Qualifier Control Register for Output B (EPWMxB)       |
| AQSFR   | 1Ah                   | 1             | No     | Action-Qualifier Software Force Register                      |
| AQCSFR  | 1Ch                   | 1             | Yes    | Action-Qualifier Continuous S/W Force Register Set            |
| <b>Dead-Band Generator Submodule Registers (Section 15.2.4.4)</b>         |                       |               |        |   |
| DBCTL   | 1Eh                   | 1             | No     | Dead-Band Generator Control Register                          |
| DBRED   | 20h                   | 1             | No     | Dead-Band Generator Rising Edge Delay Count Register          |
| DBFED   | 22h                   | 1             | No     | Dead-Band Generator Falling Edge Delay Count Register         |
| <b>Trip-Zone Submodule Registers (Section 15.2.4.5)</b>                   |                       |               |        |   |
| TZSEL   | 24h                   | 1             | No     | Trip-Zone Select Register                                     |
| TZCTL   | 28h                   | 1             | No     | Trip-Zone Control Register                                    |
| TZEINT  | 2Ah                   | 1             | No     | Trip-Zone Enable Interrupt Register                           |
| TZFLG   | 2Ch                   | 1             | No     | Trip-Zone Flag Register                                       |
| TZCLR   | 2Eh                   | 1             | No     | Trip-Zone Clear Register                                      |
| TZFRC   | 30h                   | 1             | No     | Trip-Zone Force Register                                      |
| <b>Event-Trigger Submodule Registers (Section 15.2.4.6)</b>               |                       |               |        |   |
| ETSEL   | 32h                   | 1             | No     | Event-Trigger Selection Register                              |
| ETPS  | 34h                   | 1             | No     | Event-Trigger Pre-Scale Register                              |
| ETFLG   | 36h                   | 1             | No     | Event-Trigger Flag Register                                   |
| ETCLR   | 38h                   | 1             | No     | Event-Trigger Clear Register                                  |
| ETFRC   | 3Ah                   | 1             | No     | Event-Trigger Force Register                                  |
| <b>PWM-Chopper Submodule Registers (Section 15.2.4.7)</b>                 |                       |               |        |   |
| PCCTL   | 3Ch                   | 1             | No     | PWM-Chopper Control Register                                  |
| <b>High-Resolution PWM (HRPWM) Submodule Registers (Section 15.2.4.8)</b> |                       |               |        |   |
| HRCTL   | 40h                   | 1             | No     | HRPWM Control Register <sup>(2)</sup>                         |

<sup>(1)</sup> Locations not shown are reserved.

<sup>(2)</sup> These registers are only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, these locations are reserved. See [Section 15.1.2](#) to determine which instances include the HRPWM.

### 15.2.4.1 Time-Base Submodule Registers

**Table 15-59** lists the memory-mapped registers for the time-base submodule. All other register offset addresses not listed in **Table 15-59** should be considered as reserved locations and the register contents should not be modified.

**Table 15-59. Time-Base Submodule Registers**

| Offset | Acronym | Register Description                                    | Section            |
|--------|---------|---|--------------------|
| 0h     | TBCTL   | Time-Base Control Register                              | Section 15.2.4.1.1 |
| 2h     | TBSTS   | Time-Base Status Register                               | Section 15.2.4.1.2 |
| 4h     | TBPHSR  | Time-Base Phase High-Resolution Register <sup>(1)</sup> | Section 15.2.4.8.1 |
| 6h     | TBPHS   | Time-Base Phase Register                                | Section 15.2.4.1.3 |
| 8h     | TBCNT   | Time-Base Counter Register                              | Section 15.2.4.1.4 |
| Ah     | TBPRD   | Time-Base Period Register                               | Section 15.2.4.1.5 |

<sup>(1)</sup> This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved.

#### 15.2.4.1.1 Time-Base Control Register (TBCTL)

The time-base control register (TBCTL) is shown in **Figure 15-70** and described in **Table 15-60**.

**Figure 15-70. Time-Base Control Register (TBCTL)**

| 15         | 14      | 13      | 12     | 10    | 9         | 8      |
|------------|---------|---------|--------|-------|-----------|--------|
| FREE, SOFT | PHSDIR  |         | CLKDIV |       | HSPCLKDIV |        |
| R/W-0      | R/W-0   |         | R/W-0  |       | R/W-0     |        |
| 7          | 6       | 5       | 4      | 3     | 2         | 1      |
| HSPCLKDIV  | SWFSYNC | SYNCSEL | PRDL   | PHSEN | CTRMODE   |        |
| R/W-1      | R/W-0   | R/W-0   | R/W-0  | R/W-0 | R/W-0     | R/W-3h |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-60. Time-Base Control Register (TBCTL) Field Descriptions**

| Bit   | Field      | Value | Description   |
|-------|------------|-------|---|
| 15-14 | FREE, SOFT | 0-3h  | Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:<br><br>0 Stop after the next time-base counter increment or decrement<br>1h Stop when counter completes a whole cycle: <ul style="list-style-type: none"><li>• Up-count mode: stop when the time-base counter = period (TBCNT = TBPRD)</li><li>• Down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h)</li><li>• Up-down-count mode: stop when the time-base counter = 0000 (TBCNT = 0000h)</li></ul><br>2h-3h Free run          |
| 13    | PHSDIR     |       | Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCNT) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event..<br><br>In the up-count and down-count modes this bit is ignored.<br>0 Count down after the synchronization event.<br>1 Count up after the synchronization event. |

**Table 15-60. Time-Base Control Register (TBCTL) Field Descriptions (continued)**

| Bit   | Field     | Value | Description  |
|-------|-----------|-------|--|
| 12:10 | CLKDIV    | 0-7h  | Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value.<br>$TBCLK = SYSCLKOUT/(HSPCLKDIV \times CLKDIV)$   |
|       |           | 0     | /1 (default on reset)  |
|       |           | 1h    | /2   |
|       |           | 2h    | /4   |
|       |           | 3h    | /8   |
|       |           | 4h    | /16  |
|       |           | 5h    | /32  |
|       |           | 6h    | /64  |
|       |           | 7h    | /128   |
| 9-7   | HSPCLKDIV | 0-7h  | High-Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value.<br>$TBCLK = SYSCLKOUT/(HSPCLKDIV \times CLKDIV)$<br>This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.  |
|       |           | 0     | /1   |
|       |           | 1h    | /2 (default on reset)  |
|       |           | 2h    | /4   |
|       |           | 3h    | /6   |
|       |           | 4h    | /8   |
|       |           | 5h    | /10  |
|       |           | 6h    | /12  |
|       |           | 7h    | /14  |
| 6     | SWFSYNC   | 0     | Software Forced Synchronization Pulse<br>Writing a 0 has no effect and reads always return a 0.  |
|       |           | 1     | Writing a 1 forces a one-time synchronization pulse to be generated.<br>This event is ORed with the EPWMxSYNCI input of the ePWM module.<br>SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCSEL = 00.  |
| 5-4   | SYNCSEL   | 0-3h  | Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.<br>EPWMxSYNC:<br>CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)<br>CTR = CMPB : Time-base counter equal to counter-compare B (TBCNT = CMPB)<br>Disable EPWMxSYNCO signal |
| 3     | PRDLD     | 0     | Active Period Register Load From Shadow Register Select<br>The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCNT, is equal to zero.<br>A write or read to the TBPRD register accesses the shadow register.                    |
|       |           | 1     | Load the TBPRD register immediately without using a shadow register.<br>A write or read to the TBPRD register directly accesses the active register.   |
| 2     | PHSEN     | 0     | Counter Register Load From Phase Register Enable<br>Do not load the time-base counter (TBCNT) from the time-base phase register (TBPHS)  |
|       |           | 1     | Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit.   |

**Table 15-60. Time-Base Control Register (TBCTL) Field Descriptions (continued)**

| Bit | Field   | Value | Description  |
|-----|---------|-------|--|
| 1-0 | CTRMODE | 0-3h  | <p>Counter Mode. The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change.</p> <p>These bits set the time-base counter mode of operation as follows:</p> <ul style="list-style-type: none"> <li>0 Up-count mode</li> <li>1h Down-count mode</li> <li>2h Up-down-count mode</li> <li>3h Stop-freeze counter operation (default on reset)</li> </ul> |

**15.2.4.1.2 Time-Base Status Register (TBSTS)**

The time-base status register (TBSTS) is shown in [Figure 15-71](#) and described in [Table 15-61](#).

**Figure 15-71. Time-Base Status Register (TBSTS)**

|    |          |   |         |         |     |
|----|----------|---|---------|---------|-----|
| 15 | Reserved | 3 | 2       | 1       | 0   |
|    | R-0      |   | R/W1C-0 | R/W1C-0 | R-1 |

LEGEND: R/W = Read/Write; R/W1C = Read/Write 1 to clear; -n = value after reset

**Table 15-61. Time-Base Status Register (TBSTS) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-3 | Reserved | 0     | Reserved   |
| 2    | CTRMAX   | 0     | Time-Base Counter Max Latched Status Bit   |
|      |          | 1     | Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect.  |
|      |          | 1     | Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event.   |
| 1    | SYNCI    | 0     | Input Synchronization Latched Status Bit   |
|      |          | 1     | Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred. Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit will clear the latched event. |
| 0    | CTRDIR   | 0     | Time-Base Counter Direction Status Bit. At reset, the counter is frozen; therefore, this bit has no meaning. To make this bit meaningful, you must first set the appropriate mode via TBCTL[CTRMODE].  |
|      |          | 1     | Time-Base Counter is currently counting down.  |
|      |          | 1     | Time-Base Counter is currently counting up.  |

**15.2.4.1.3 Time-Base Phase Register (TBPHS)**

The time-base phase register (TBPHS) is shown in [Figure 15-72](#) and described in [Table 15-62](#).

**Figure 15-72. Time-Base Phase Register (TBPHS)**

|    |       |   |
|----|-------|---|
| 15 | TBPHS | 0 |
|    | R/W-0 |   |

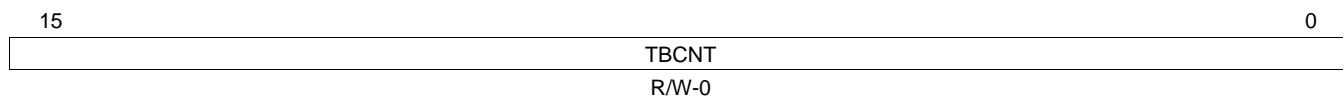
LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-62. Time-Base Phase Register (TBPHS) Field Descriptions**

| Bits | Name  | Value   | Description  |
|------|-------|---------|--|
| 15-0 | TBPHS | 0xFFFFh | <p>These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal.</p> <ul style="list-style-type: none"> <li>If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase.</li> <li>If TBCTL[PHSEN] = 1, then the time-base counter (TBCNT) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCl) or by a software forced synchronization.</li> </ul> |

**15.2.4.1.4 Time-Base Counter Register (TBCNT)**

The time-base counter register (TBCNT) is shown in [Figure 15-73](#) and described in [Table 15-63](#).

**Figure 15-73. Time-Base Counter Register (TBCNT)**


LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-63. Time-Base Counter Register (TBCNT) Field Descriptions**

| Bits | Name  | Value   | Description  |
|------|-------|---------|--|
| 15-0 | TBCNT | 0xFFFFh | <p>Reading these bits gives the current time-base counter value.</p> <p>Writing to these bits sets the current time-base counter value. The update happens as soon as the write occurs; the write is NOT synchronized to the time-base clock (TBCLK) and the register is not shadowed.</p> |

#### 15.2.4.1.5 Time-Base Period Register (TBPRD)

The time-base period register (TBPRD) is shown in [Figure 15-74](#) and described in [Table 15-64](#).

**Figure 15-74. Time-Base Period Register (TBPRD)**

|       |   |
|-------|---|
| 15    | 0 |
| TBPRD |   |
| R/W-0 |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-64. Time-Base Period Register (TBPRD) Field Descriptions**

| Bits | Name  | Value   | Description   |
|------|-------|---------|---|
| 15-0 | TBPRD | 0xFFFFh | <p>These bits determine the period of the time-base counter. This sets the PWM frequency.</p> <p>Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero.</li> <li>If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>The active and shadow registers share the same memory map address.</li> </ul> |

#### 15.2.4.2 Counter-Compare Submodule Registers

[Table 15-65](#) lists the memory-mapped registers for the counter-compare submodule. All other register offset addresses not listed in [Table 15-65](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-65. Counter-Compare Submodule Registers**

| Offset | Acronym | Register Description                                      | Section                            |
|--------|---------|---|------------------------------------|
| Eh     | CMPCTL  | Counter-Compare Control Register                          | <a href="#">Section 15.2.4.2.1</a> |
| 10h    | CMPAHR  | Counter-Compare A High-Resolution Register <sup>(1)</sup> | <a href="#">Section 15.2.4.8.2</a> |
| 12h    | CMPA    | Counter-Compare A Register                                | <a href="#">Section 15.2.4.2.2</a> |
| 14h    | CMPB    | Counter-Compare B Register                                | <a href="#">Section 15.2.4.2.3</a> |

<sup>(1)</sup> This register is only available on ePWM instances that include the high-resolution PWM (HRPWM) extension; otherwise, this location is reserved.

#### 15.2.4.2.1 Counter-Compare Control Register (CMPCTL)

The counter-compare control register (CMPCTL) is shown in [Figure 15-75](#) and described in [Table 15-66](#).

**Figure 15-75. Counter-Compare Control Register (CMPCTL)**

| 15       | Reserved  |   |          |           |   |           | 10  | 9         | 8   |
|----------|-----------|---|----------|-----------|---|-----------|-----|-----------|-----|
|          |           |   |          |           |   |           | R-0 | R-0       | R-0 |
| 7        | 6         | 5 | 4        | 3         | 2 | 1         |     |           | 0   |
| Reserved | SHDWBMODE |   | Reserved | SHDWAMODE |   | LOADBMODE |     | LOADAMODE |     |
| R-0      | R/W-0     |   | R-0      | R/W-0     |   | R/W-0     |     | R/W-0     |     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-66. Counter-Compare Control Register (CMPCTL) Field Descriptions**

| Bits  | Name      | Value | Description  |
|-------|-----------|-------|--|
| 15-10 | Reserved  | 0     | Reserved   |
| 9     | SHDWBFULL | 0     | Counter-compare B (CMPB) Shadow Register Full Status Flag. This bit self clears once a load-strobe occurs.   |
|       |           | 1     | CMPB shadow FIFO not full yet  |
|       |           | 1     | Indicates the CMPB shadow FIFO is full; a CPU write will overwrite current shadow value.   |
| 8     | SHDWAFULL | 0     | Counter-compare A (CMPA) Shadow Register Full Status Flag. The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag. This bit self clears once a load-strobe occurs. |
|       |           | 1     | CMPA shadow FIFO not full yet  |
|       |           | 1     | Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value.   |
| 7     | Reserved  | 0     | Reserved   |
| 6     | SHDWBMODE | 0     | Counter-compare B (CMPB) Register Operating Mode   |
|       |           | 1     | Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.   |
|       |           | 1     | Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action.   |
| 5     | Reserved  |       | Reserved   |
| 4     | SHDWAMODE | 0     | Counter-compare A (CMPA) Register Operating Mode   |
|       |           | 1     | Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.   |
|       |           | 1     | Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action  |
| 3-2   | LOADBMODE | 0-3h  | Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).  |
|       |           | 0     | Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)   |
|       |           | 1h    | Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)   |
|       |           | 2h    | Load on either CTR = 0 or CTR = PRD  |
|       |           | 3h    | Freeze (no loads possible)   |
| 1-0   | LOADAMODE | 0-3h  | Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).  |
|       |           | 0     | Load on CTR = 0: Time-base counter equal to zero (TBCNT = 0000h)   |
|       |           | 1h    | Load on CTR = PRD: Time-base counter equal to period (TBCNT = TBPRD)   |
|       |           | 2h    | Load on either CTR = 0 or CTR = PRD  |
|       |           | 3h    | Freeze (no loads possible)   |

### 15.2.4.2.2 Counter-Compare A Register (CMPA)

The counter-compare A register (CMPA) is shown in [Figure 15-76](#) and described in [Table 15-67](#).

**Figure 15-76. Counter-Compare A Register (CMPA)**

|    |       |   |
|----|-------|---|
| 15 | CMPA  | 0 |
|    | R/W-0 |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-67. Counter-Compare A Register (CMPA) Field Descriptions**

| Bits | Name | Value   | Description  |
|------|------|---------|--|
| 15-0 | CMPA | 0xFFFFh | <p>The value in the active CMPA register is continuously compared to the time-base counter (TBCNT). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>• Do nothing; the event is ignored.</li> <li>• Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>• Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>• Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>• If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register.</li> <li>• Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full.</li> <li>• If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>• In either mode, the active and shadow registers share the same memory map address.</li> </ul> |

### 15.2.4.2.3 Counter-Compare B Register (CMPB)

The counter-compare B register (CMPB) is shown in [Figure 15-77](#) and described in [Table 15-68](#).

**Figure 15-77. Counter-Compare B Register (CMPB)**

|       |   |
|-------|---|
| 15    | 0 |
| CMPB  |   |
| R/W-0 |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-68. Counter-Compare B Register (CMPB) Field Descriptions**

| Bits | Name | Value   | Description  |
|------|------|---------|--|
| 15-0 | CMPB | 0xFFFFh | <p>The value in the active CMPB register is continuously compared to the time-base counter (TBCNT). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare B" event. This event is sent to the action-qualifier where it is qualified and converted into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include:</p> <ul style="list-style-type: none"> <li>• Do nothing. event is ignored.</li> <li>• Clear: Pull the EPWMxA and/or EPWMxB signal low</li> <li>• Set: Pull the EPWMxA and/or EPWMxB signal high</li> <li>• Toggle the EPWMxA and/or EPWMxB signal</li> </ul> <p>Shadowing of this register is enabled and disabled by the CMPCTL[SHDWBMODE] bit. By default this register is shadowed.</p> <ul style="list-style-type: none"> <li>• If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register:</li> <li>• Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full.</li> <li>• If CMPCTL[SHDWBMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware.</li> <li>• In either mode, the active and shadow registers share the same memory map address.</li> </ul> |

### 15.2.4.3 Action-Qualifier Submodule Registers

[Table 15-69](#) lists the memory-mapped registers for the action-qualifier submodule. All other register offset addresses not listed in [Table 15-69](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-69. Action-Qualifier Submodule Registers**

| Offset | Acronym | Register Description                                | Section                            |
|--------|---------|---|------------------------------------|
| 16h    | AQCTLA  | Action-Qualifier Output A Control Register          | <a href="#">Section 15.2.4.3.1</a> |
| 18h    | AQCTLB  | Action-Qualifier Output B Control Register          | <a href="#">Section 15.2.4.3.2</a> |
| 1Ah    | AQSFRC  | Action-Qualifier Software Force Register            | <a href="#">Section 15.2.4.3.3</a> |
| 1Ch    | AQCSFRC | Action-Qualifier Continuous Software Force Register | <a href="#">Section 15.2.4.3.4</a> |

#### 15.2.4.3.1 Action-Qualifier Output A Control Register (AQCTLA)

The action-qualifier output A control register (AQCTLA) is shown in [Figure 15-78](#) and described in [Table 15-70](#).

**Figure 15-78. Action-Qualifier Output A Control Register (AQCTLA)**

| 15       | 12 | 11    | 10    | 9     | 8     | 7     | 6     | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|-------|-------|-------|-------|-------|-------|---|---|---|---|---|---|
| Reserved |    | CBD   | CBU   | CAD   | CAU   | PRD   | ZRO   |   |   |   |   |   |   |
| R-0      |    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-70. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions**

| Bits  | Name     | Value | Description   |
|-------|----------|-------|---|
| 15-12 | Reserved | 0     | Reserved  |
| 11-10 | CBD      | 0-3h  | Action when the time-base counter equals the active CMPB register and the counter is decrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |
| 9-8   | CBU      | 0-3h  | Action when the counter equals the active CMPB register and the counter is incrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |
| 7-6   | CAD      | 0-3h  | Action when the counter equals the active CMPA register and the counter is decrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |
| 5-4   | CAU      | 0-3h  | Action when the counter equals the active CMPA register and the counter is incrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |
| 3-2   | PRD      | 0-3h  | Action when the counter equals the period.<br>Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down. |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |
| 1-0   | ZRO      | 0-3h  | Action when counter equals zero.<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.                  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxA output low.   |
|       |          | 2h    | Set: force EPWMxA output high.  |
|       |          | 3h    | Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.  |

#### 15.2.4.3.2 Action-Qualifier Output B Control Register (AQCTLB)

The action-qualifier output B control register (AQCTLB) is shown in [Figure 15-79](#) and described in [Table 15-71](#).

**Figure 15-79. Action-Qualifier Output B Control Register (AQCTLB)**

| 15       | 12 | 11    | 10    | 9     | 8     | 7     | 6     | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|-------|-------|-------|-------|-------|-------|---|---|---|---|---|---|
| Reserved |    | CBD   | CBU   | CAD   | CAU   | PRD   | ZRO   |   |   |   |   |   |   |
| R-0      |    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-71. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions**

| Bits  | Name     | Value | Description   |
|-------|----------|-------|---|
| 15-12 | Reserved | 0     | Reserved  |
| 11-10 | CBD      | 0-3h  | Action when the counter equals the active CMPB register and the counter is decrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |
| 9-8   | CBU      | 0-3h  | Action when the counter equals the active CMPB register and the counter is incrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |
| 7-6   | CAD      | 0-3h  | Action when the counter equals the active CMPA register and the counter is decrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |
| 5-4   | CAU      | 0-3h  | Action when the counter equals the active CMPA register and the counter is incrementing.  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |
| 3-2   | PRD      | 0-3h  | Action when the counter equals the period.<br>Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down. |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |
| 1-0   | ZRO      | 0-3h  | Action when counter equals zero.<br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.                  |
|       |          | 0     | Do nothing (action disabled)  |
|       |          | 1h    | Clear: force EPWMxB output low.   |
|       |          | 2h    | Set: force EPWMxB output high.  |
|       |          | 3h    | Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.  |

### 15.2.4.3.3 Action-Qualifier Software Force Register (AQSFRC)

The action-qualifier software force register (AQSFRC) is shown in Figure 15-80 and described in Table 15-72.

**Figure 15-80. Action-Qualifier Software Force Register (AQSFRC)**

| 15       | 8      | 7     | 6     | 5     | 4      | 3     | 2      | 1 | 0 |
|----------|--------|-------|-------|-------|--------|-------|--------|---|---|
| Reserved | RLDCSF |       |       | OTSFB | ACTSFB | OTSFA | ACTSFA |   |   |
| R-0      | R/W-0  | R/W-0 | R/W-0 | R/W-0 | R/W-0  | R/W-0 |        |   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-72. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved   |
| 7-6  | RLDCSF   | 0-3h  | AQCSFRC Active Register Reload From Shadow Options   |
|      |          | 0     | Load on event counter equals zero  |
|      |          | 1h    | Load on event counter equals period  |
|      |          | 2h    | Load on event counter equals zero or counter equals period   |
|      |          | 3h    | Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register).   |
| 5    | OTSFB    | 0     | One-Time Software Forced Event on Output B<br>Writing a 0 (zero) has no effect. Always reads back a 0<br>This bit is auto cleared once a write to this register is complete, that is, a forced event is initiated.)<br>This is a one-shot forced event. It can be overridden by another subsequent event on output B.<br>Initiates a single s/w forced event |
|      |          | 1     |  |
| 4-3  | ACTSFB   | 0-3h  | Action when One-Time Software Force B Is invoked   |
|      |          | 0     | Does nothing (action disabled)   |
|      |          | 1h    | Clear (low)  |
|      |          | 2h    | Set (high)   |
|      |          | 3h    | Toggle (Low → High, High → Low)<br><b>Note:</b> This action is not qualified by counter direction (CNT_dir)  |
| 2    | OTSFA    | 0     | One-Time Software Forced Event on Output A<br>Writing a 0 (zero) has no effect. Always reads back a 0.<br>This bit is auto cleared once a write to this register is complete (that is, a forced event is initiated).<br>Initiates a single software forced event   |
|      |          | 1     |  |
| 1-0  | ACTSFA   | 0-3h  | Action When One-Time Software Force A Is Invoked   |
|      |          | 0     | Does nothing (action disabled)   |
|      |          | 1h    | Clear (low)  |
|      |          | 2h    | Set (high)   |
|      |          | 3h    | Toggle (Low → High, High → Low)<br><b>Note:</b> This action is not qualified by counter direction (CNT_dir)  |

#### 15.2.4.3.4 Action-Qualifier Continuous Software Force Register (AQCSFRC)

The action-qualifier continuous software force register (AQCSFRC) is shown in [Figure 15-81](#) and described in [Table 15-73](#).

**Figure 15-81. Action-Qualifier Continuous Software Force Register (AQCSFRC)**

| 15       | 4 | 3 | 2     | 1 | 0     |
|----------|---|---|-------|---|-------|
| Reserved |   |   | CSFB  |   | CSFA  |
| R-0      |   |   | R/W-0 |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-73. Action-Qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions**

| Bits | Name     | Value | Description  |
|------|----------|-------|--|
| 15-4 | Reserved | 0     | Reserved   |
| 3-2  | CSFB     | 0-3h  | Continuous Software Force on Output B<br>In immediate mode, a continuous force takes effect on the next TBCLK edge.<br>In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register. To configure shadow mode, use AQSFR[RLDCSF].<br>0 Forcing disabled, that is, has no effect<br>1h Forces a continuous low on output B<br>2h Forces a continuous high on output B<br>3h Software forcing is disabled and has no effect |
| 1-0  | CSFA     | 0-3h  | Continuous Software Force on Output A<br>In immediate mode, a continuous force takes effect on the next TBCLK edge.<br>In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.<br>0 Forcing disabled, that is, has no effect<br>1h Forces a continuous low on output A<br>2h Forces a continuous high on output A<br>3h Software forcing is disabled and has no effect  |

#### 15.2.4.4 Dead-Band Generator Submodule Registers

[Table 15-74](#) lists the memory-mapped registers for the dead-band generator submodule. All other register offset addresses not listed in [Table 15-74](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-74. Dead-Band Generator Submodule Registers**

| Offset | Acronym | Register Description                            | Section                            |
|--------|---------|---|------------------------------------|
| 1Eh    | DBCTL   | Dead-Band Generator Control Register            | <a href="#">Section 15.2.4.4.1</a> |
| 20h    | DBRED   | Dead-Band Generator Rising Edge Delay Register  | <a href="#">Section 15.2.4.4.2</a> |
| 22h    | DBFED   | Dead-Band Generator Falling Edge Delay Register | <a href="#">Section 15.2.4.4.3</a> |

#### 15.2.4.4.1 Dead-Band Generator Control Register (DBCTL)

The dead-band generator control register (DBCTL) is shown in [Figure 15-82](#) and described in [Table 15-75](#).

**Figure 15-82. Dead-Band Generator Control Register (DBCTL)**

| 15       | 6 | 5 | 4 | 3       | 2      | 1        | 0 |
|----------|---|---|---|---------|--------|----------|---|
| Reserved |   |   |   | IN_MODE | POLSEL | OUT_MODE |   |
| R-0      |   |   |   | R/W-0   | R/W-0  | R/W-0    |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-75. Dead-Band Generator Control Register (DBCTL) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-6 | Reserved | 0     | Reserved  |
| 5-4  | IN_MODE  | 0-3h  | <p>Dead Band Input Mode Control. Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in <a href="#">Figure 15-34</a>. This allows you to select the input source to the falling-edge and rising-edge delay.</p> <p>To produce classical dead-band waveforms, the default is EPWMxA In is the source for both falling and rising-edge delays.</p> <ul style="list-style-type: none"> <li>0 EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.</li> <li>1h EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</li> <li>2h EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.</li> <li>EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.</li> <li>3h EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.</li> <li>EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.</li> </ul>   |
| 3-2  | POLSEL   | 0-3h  | <p>Polarity Select Control. Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in <a href="#">Figure 15-34</a>. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule.</p> <p>The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter.</p> <p>These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.</p> <ul style="list-style-type: none"> <li>0 Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).</li> <li>1h Active low complementary (ALC) mode. EPWMxA is inverted.</li> <li>2h Active high complementary (AHC). EPWMxB is inverted.</li> <li>3h Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.</li> </ul>  |
| 1-0  | OUT_MODE | 0-3h  | <p>Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in <a href="#">Figure 15-34</a>. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.</p> <ul style="list-style-type: none"> <li>0 Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.</li> <li>1h Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule.</li> <li>The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].</li> <li>2h Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule.</li> <li>The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE].</li> <li>3h Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].</li> </ul> |

#### 15.2.4.4.2 Dead-Band Generator Rising Edge Delay Register (DBRED)

The dead-band generator rising edge delay register (DBRED) is shown in [Figure 15-83](#) and described in [Table 15-76](#).

**Figure 15-83. Dead-Band Generator Rising Edge Delay Register (DBRED)**

| 15       | 10 | 9 | 0     |
|----------|----|---|-------|
| Reserved |    |   | DEL   |
| R-0      |    |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-76. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions**

| Bits  | Name     | Value  | Description                              |
|-------|----------|--------|--|
| 15-10 | Reserved | 0      | Reserved                                 |
| 9-0   | DEL      | 0-3FFh | Rising Edge Delay Count. 10-bit counter. |

#### 15.2.4.4.3 Dead-Band Generator Falling Edge Delay Register (DBFED)

The dead-band generator falling edge delay register (DBFED) is shown in [Figure 15-84](#) and described in [Table 15-77](#).

**Figure 15-84. Dead-Band Generator Falling Edge Delay Register (DBFED)**

| 15       | 10 | 9 | 0     |
|----------|----|---|-------|
| Reserved |    |   | DEL   |
| R-0      |    |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-77. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions**

| Bits  | Name     | Value  | Description                              |
|-------|----------|--------|--|
| 15-10 | Reserved | 0      | Reserved                                 |
| 9-0   | DEL      | 0-3FFh | Falling Edge Delay Count. 10-bit counter |

### 15.2.4.5 Trip-Zone Submodule Registers

[Table 15-78](#) lists the memory-mapped registers for the trip-zone submodule. All other register offset addresses not listed in [Table 15-78](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-78. Trip-Zone Submodule Registers**

| Offset | Acronym | Register Description                | Section                            |
|--------|---------|-------------------------------------|------------------------------------|
| 24h    | TZSEL   | Trip-Zone Select Register           | <a href="#">Section 15.2.4.5.1</a> |
| 28h    | TZCTL   | Trip-Zone Control Register          | <a href="#">Section 15.2.4.5.2</a> |
| 2Ah    | TZEINT  | Trip-Zone Enable Interrupt Register | <a href="#">Section 15.2.4.5.3</a> |
| 2Ch    | TZFLG   | Trip-Zone Flag Register             | <a href="#">Section 15.2.4.5.4</a> |
| 2Eh    | TZCLR   | Trip-Zone Clear Register            | <a href="#">Section 15.2.4.5.5</a> |
| 30h    | TZFRC   | Trip-Zone Force Register            | <a href="#">Section 15.2.4.5.6</a> |

#### 15.2.4.5.1 Trip-Zone Select Register (TZSEL)

The trip-zone select register (TZSEL) is shown in [Figure 15-85](#) and described in [Table 15-79](#).

**Figure 15-85. Trip-Zone Select Register (TZSEL)**

| 15  | 9     | 8 | 7  | 1    | 0 |
|---|-------|---|--|------|---|
| Reserved/OSHT <sub>n</sub> <sup>(1)</sup> | OSHT1 |   | Reserved/CBC <sub>n</sub> <sup>(1)</sup> | CBC1 |   |

LEGEND: R/W = Read/Write; -n = value after reset

<sup>(1)</sup> Number of register bits depends on how many trip-zone pins are available in the device. See your device-specific data manual.

**Table 15-79. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions**

| Bits | Name              | Value | Description  |
|------|-------------------|-------|--|
| 15-8 | OSHT <sub>n</sub> | 0     | Trip-zone <sub>n</sub> (TZ <sub>n</sub> ) select. One-Shot (OSHT) trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register ( <a href="#">Section 15.2.4.5.2</a> ) is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until you clear the condition via the TZCLR register ( <a href="#">Section 15.2.4.5.5</a> ). |
|      |                   | 1     | Disable TZ <sub>n</sub> as a one-shot trip source for this ePWM module.<br>Enable TZ <sub>n</sub> as a one-shot trip source for this ePWM module.  |
| 7-0  | CBC <sub>n</sub>  | 0     | Trip-zone <sub>n</sub> (TZ <sub>n</sub> ) select. Cycle-by-Cycle (CBC) trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register ( <a href="#">Section 15.2.4.5.2</a> ) is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.                             |
|      |                   | 1     | Disable TZ <sub>n</sub> as a CBC trip source for this ePWM module.<br>Enable TZ <sub>n</sub> as a CBC trip source for this ePWM module.  |

#### 15.2.4.5.2 Trip-Zone Control Register (TZCTL)

The trip-zone control register (TZCTL) is shown in [Figure 15-86](#) and described in [Table 15-80](#).

**Figure 15-86. Trip-Zone Control Register (TZCTL)**

|    |          |   |     |   |     |   |
|----|----------|---|-----|---|-----|---|
| 15 | Reserved | 4 | 3   | 2 | 1   | 0 |
|    | R-0      |   | TZB |   | TZA |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-80. Trip-Zone Control Register (TZCTL) Field Descriptions**

| Bits | Name     | Value | Description  |
|------|----------|-------|--|
| 15–4 | Reserved | 0     | Reserved   |
| 3–2  | TZB      | 0-3h  | When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register ( <a href="#">Section 15.2.4.5.1</a> ).<br>0 High impedance (EPWMxB = High-impedance state)<br>1h Force EPWMxB to a high state<br>2h Force EPWMxB to a low state<br>3h Do nothing, no action is taken on EPWMxB. |
| 1–0  | TZA      | 0-3h  | When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register ( <a href="#">Section 15.2.4.5.1</a> ).<br>0 High impedance (EPWMxA = High-impedance state)<br>1h Force EPWMxA to a high state<br>2h Force EPWMxA to a low state<br>3h Do nothing, no action is taken on EPWMxA. |

#### 15.2.4.5.3 Trip-Zone Enable Interrupt Register (TZEINT)

The trip-zone enable interrupt register (TZEINT) is shown in [Figure 15-87](#) and described in [Table 15-81](#).

**Figure 15-87. Trip-Zone Enable Interrupt Register (TZEINT)**

|    |          |     |     |      |   |
|----|----------|-----|-----|------|---|
| 15 | Reserved | 3   | 2   | 1    | 0 |
|    | R-0      | OST | CBC | Rsvd |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-81. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions**

| Bits | Name     | Value | Description  |
|------|----------|-------|--|
| 15–3 | Reserved | 0     | Reserved   |
| 2    | OST      | 0     | Trip-zone One-Shot Interrupt Enable<br>Disable one-shot interrupt generation                 |
|      |          | 1     | Enable Interrupt generation; a one-shot trip event will cause a EPWMxTZINT interrupt.        |
| 1    | CBC      | 0     | Trip-zone Cycle-by-Cycle Interrupt Enable<br>Disable cycle-by-cycle interrupt generation.    |
|      |          | 1     | Enable interrupt generation; a cycle-by-cycle trip event will cause an EPWMxTZINT interrupt. |
| 0    | Reserved | 0     | Reserved   |

#### 15.2.4.5.4 Trip-Zone Flag Register (TZFLG)

The trip-zone flag register (TZFLG) is shown in [Figure 15-88](#) and described in [Table 15-82](#).

**Figure 15-88. Trip-Zone Flag Register (TZFLG)**

|    |          |   |     |     |     |
|----|----------|---|-----|-----|-----|
| 15 | Reserved | 3 | 2   | 1   | 0   |
|    | R-0      |   | R-0 | R-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-82. Trip-Zone Flag Register (TZFLG) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-3 | Reserved | 0     | Reserved  |
| 2    | OST      | 0     | Latched Status Flag for A One-Shot Trip Event.<br>No one-shot trip event has occurred.  |
|      |          | 1     | Indicates a trip event has occurred on a pin selected as a one-shot trip source.<br>This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 15.2.4.5.5</a> ).  |
| 1    | CBC      | 0     | Latched Status Flag for Cycle-By-Cycle Trip Event<br>No cycle-by-cycle trip event has occurred.   |
|      |          | 1     | Indicates a trip event has occurred on a pin selected as a cycle-by-cycle trip source. The TZFLG[CBC] bit will remain set until it is manually cleared by the user. If the cycle-by-cycle trip event is still present when the CBC bit is cleared, then CBC will be immediately set again. The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCNT = 0000h) if the trip condition is no longer present. The condition on the pins is only cleared when the TBCNT = 0000h no matter where in the cycle the CBC flag is cleared.<br>This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 15.2.4.5.5</a> ). |
| 0    | INT      | 0     | Latched Trip Interrupt Status Flag<br>Indicates no interrupt has been generated.  |
|      |          | 1     | Indicates an EPWMxTZINT interrupt was generated because of a trip condition.<br>No further EPWMxTZINT interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.<br>This bit is cleared by writing the appropriate value to the TZCLR register ( <a href="#">Section 15.2.4.5.5</a> ).  |

#### 15.2.4.5.5 Trip-Zone Clear Register (TZCLR)

The trip-zone clear register (TZCLR) is shown in [Figure 15-89](#) and described in [Table 15-83](#).

**Figure 15-89. Trip-Zone Clear Register (TZCLR)**

| 15  | Reserved | 3 | 2     | 1     | 0     |
|-----|----------|---|-------|-------|-------|
|     |          |   | OST   | CBC   | INT   |
| R-0 |          |   | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-83. Trip-Zone Clear Register (TZCLR) Field Descriptions**

| Bits   | Name     | Value | Description  |
|--|----------|-------|--|
| 15-3   | Reserved | 0     | Reserved   |
| 2  | OST      | 0     | Clear Flag for One-Shot Trip (OST) Latch   |
|  |          | 1     | Has no effect. Always reads back a 0.<br>Clears this Trip (set) condition.                                 |
| 1  | CBC      | 0     | Clear Flag for Cycle-By-Cycle (CBC) Trip Latch   |
|  |          | 1     | Has no effect. Always reads back a 0.<br>Clears this Trip (set) condition.                                 |
| 0  | INT      | 0     | Global Interrupt Clear Flag  |
|  |          | 1     | Has no effect. Always reads back a 0.<br>Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]). |
| <b>NOTE:</b> No further EPWMxTZINT interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated.<br>Clearing all flag bits will prevent further interrupts. |          |       |  |

#### 15.2.4.5.6 Trip-Zone Force Register (TZFRC)

The trip-zone force register (TZFRC) is shown in [Figure 15-90](#) and described in [Table 15-84](#).

**Figure 15-90. Trip-Zone Force Register (TZFRC)**

| 15  | Reserved | 3 | 2     | 1     | 0    |
|-----|----------|---|-------|-------|------|
|     |          |   | OST   | CBC   | Rsvd |
| R-0 |          |   | R/W-0 | R/W-0 | R-0  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-84. Trip-Zone Force Register (TZFRC) Field Descriptions**

| Bits | Name     | Value | Description  |
|------|----------|-------|--|
| 15-3 | Reserved | 0     | Reserved   |
| 2    | OST      | 0     | Force a One-Shot Trip Event via Software   |
|      |          | 1     | Writing of 0 is ignored. Always reads back a 0.<br>Forces a one-shot trip event and sets the TZFLG[OST] bit.       |
| 1    | CBC      | 0     | Force a Cycle-by-Cycle Trip Event via Software   |
|      |          | 1     | Writing of 0 is ignored. Always reads back a 0.<br>Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit. |
| 0    | Reserved | 0     | Reserved   |

#### 15.2.4.6 Event-Trigger Submodule Registers

[Table 15-85](#) lists the memory-mapped registers for the event-trigger submodule. See your device-specific data manual for the memory address of these registers. All other register offset addresses not listed in [Table 15-85](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-85. Event-Trigger Submodule Registers**

| Offset | Acronym | Register Description             | Section                            |
|--------|---------|----------------------------------|------------------------------------|
| 32h    | ETSEL   | Event-Trigger Selection Register | <a href="#">Section 15.2.4.6.1</a> |
| 34h    | ETPS    | Event-Trigger Prescale Register  | <a href="#">Section 15.2.4.6.2</a> |
| 36h    | ETFLG   | Event-Trigger Flag Register      | <a href="#">Section 15.2.4.6.3</a> |
| 38h    | ETCLR   | Event-Trigger Clear Register     | <a href="#">Section 15.2.4.6.4</a> |
| 3Ah    | ETFRC   | Event-Trigger Force Register     | <a href="#">Section 15.2.4.6.5</a> |

##### 15.2.4.6.1 Event-Trigger Selection Register (ETSEL)

The event-trigger selection register (ETSEL) is shown in [Figure 15-91](#) and described in [Table 15-86](#).

**Figure 15-91. Event-Trigger Selection Register (ETSEL)**

| 15       | 4   | 3     | 2      | 0           |
|----------|-----|-------|--------|-------------|
| Reserved | R-0 | INTEN | INTSEL | R/W-0 R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-86. Event-Trigger Selection Register (ETSEL) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-4 | Reserved | 0     | Reserved  |
| 3    | INTEN    | 0     | Enable ePWM Interrupt (EPWMx_INT) Generation                                  |
|      |          | 1     | Disable EPWMx_INT generation  |
| 2-0  | INTSEL   | 0-7h  | ePWM Interrupt (EPWMx_INT) Selection Options                                  |
|      |          | 0     | Reserved  |
|      |          | 1h    | Enable event time-base counter equal to zero. (TBCNT = 0000h)                 |
|      |          | 2h    | Enable event time-base counter equal to period (TBCNT = TBPRD)                |
|      |          | 3h    | Reserved  |
|      |          | 4h    | Enable event time-base counter equal to CMPA when the timer is incrementing.  |
|      |          | 5h    | Enable event time-base counter equal to CMPA when the timer is decrementing.  |
|      |          | 6h    | Enable event: time-base counter equal to CMPB when the timer is incrementing. |
|      |          | 7h    | Enable event: time-base counter equal to CMPB when the timer is decrementing. |

#### 15.2.4.6.2 Event-Trigger Prescale Register (ETPS)

The event-trigger prescale register (ETPS) is shown in [Figure 15-92](#) and described in [Table 15-87](#).

**Figure 15-92. Event-Trigger Prescale Register (ETPS)**

|    |          |     |   |        |     |       |
|----|----------|-----|---|--------|-----|-------|
| 15 | Reserved | 4   | 3 | 2      | 1   | 0     |
|    |          | R-0 |   | INTCNT | R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-87. Event-Trigger Prescale Register (ETPS) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-4 | Reserved | 0     | Reserved  |
| 3-2  | INTCNT   | 0-3h  | ePWM Interrupt Event (EPWMx_INT) Counter Register. These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].  |
|      |          | 0     | No events have occurred.  |
|      |          | 1h    | 1 event has occurred.   |
|      |          | 2h    | 2 events have occurred.   |
|      |          | 3h    | 3 events have occurred.   |
| 1-0  | INTPRD   | 0-3h  | ePWM Interrupt (EPWMx_INT) Period Select. These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared.<br><br>Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear.<br><br>Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented. |
|      |          | 0     | Disable the interrupt event counter. No interrupt will be generated and ETFLG[INT] is ignored.  |
|      |          | 1h    | Generate an interrupt on the first event INTCNT = 01 (first event)  |
|      |          | 2h    | Generate interrupt on ETPS[INTCNT] = 1,0 (second event)   |
|      |          | 3h    | Generate interrupt on ETPS[INTCNT] = 1,1 (third event)  |

#### 15.2.4.6.3 Event-Trigger Flag Register (ETFLG)

The event-trigger flag register (ETFLG) is shown in [Figure 15-93](#) and described in [Table 15-88](#).

**Figure 15-93. Event-Trigger Flag Register (ETFLG)**

|    |          |     |   |   |     |
|----|----------|-----|---|---|-----|
| 15 | Reserved | R-0 | 1 | 0 | INT |
|----|----------|-----|---|---|-----|

LEGEND: R = Read only; -n = value after reset

**Table 15-88. Event-Trigger Flag Register (ETFLG) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-1 | Reserved | 0     | Reserved  |
| 0    | INT      | 0     | Latched ePWM Interrupt (EPWMx_INT) Status Flag<br>Indicates no event occurred   |
|      |          | 1     | Indicates that an ePWMx interrupt (EPWMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Refer to <a href="#">Figure 15-47</a> . |

#### 15.2.4.6.4 Event-Trigger Clear Register (ETCLR)

The event-trigger clear register (ETCLR) is shown in [Figure 15-94](#) and described in [Table 15-89](#).

**Figure 15-94. Event-Trigger Clear Register (ETCLR)**

|    |          |     |   |   |     |
|----|----------|-----|---|---|-----|
| 15 | Reserved | R-0 | 1 | 0 | INT |
|----|----------|-----|---|---|-----|

LEGEND: R = Read only; -n = value after reset

**Table 15-89. Event-Trigger Clear Register (ETCLR) Field Descriptions**

| Bits | Name     | Value | Description  |
|------|----------|-------|--|
| 15-1 | Reserved | 0     | Reserved   |
| 0    | INT      | 0     | ePWM Interrupt (EPWMx_INT) Flag Clear Bit<br>Writing a 0 has no effect. Always reads back a 0. |
|      |          | 1     | Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated.           |

#### 15.2.4.6.5 Event-Trigger Force Register (ETFRC)

The event-trigger force register (ETFRC) is shown in [Figure 15-95](#) and described in [Table 15-90](#).

**Figure 15-95. Event-Trigger Force Register (ETFRC)**

|    |          |     |     |     |
|----|----------|-----|-----|-----|
| 15 | Reserved | R-0 | 1   | 0   |
|    |          |     | INT | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 15-90. Event-Trigger Force Register (ETFRC) Field Descriptions**

| Bits | Name     | Value | Description   |
|------|----------|-------|---|
| 15-1 | Reserved | 0     | Reserved  |
| 0    | INT      | 0     | INT Force Bit. The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless. |
|      |          | 1     | Writing 0 to this bit will be ignored. Always reads back a 0.   |
|      |          | 1     | Generates an interrupt on $\overline{EPWMxINT}$ and set the INT flag bit. This bit is used for test purposes.                               |

#### 15.2.4.7 PWM-Chopper Submodule Register

The PWM-chopper control register (PCCTL) is shown in [Figure 15-96](#) and described in [Table 15-91](#).

**Figure 15-96. PWM-Chopper Control Register (PCCTL)**

| 15       | 11 | 10      | 8 | 7       | 5 | 4       | 1 | 0     |
|----------|----|---------|---|---------|---|---------|---|-------|
| Reserved |    | CHPDUTY |   | CHPFREQ |   | OSHTWTH |   | CHPEN |
| R-0      |    | R/W-0   |   | R/W-0   |   | R/W-0   |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-91. PWM-Chopper Control Register (PCCTL) Bit Descriptions**

| Bits  | Name     | Value | Description                                   |
|-------|----------|-------|---|
| 15-11 | Reserved | 0     | Reserved                                      |
| 10-8  | CHPDUTY  | 0-7h  | Chopping Clock Duty Cycle                     |
|       |          | 0     | Duty = 1/8 (12.5%)                            |
|       |          | 1h    | Duty = 2/8 (25.0%)                            |
|       |          | 2h    | Duty = 3/8 (37.5%)                            |
|       |          | 3h    | Duty = 4/8 (50.0%)                            |
|       |          | 4h    | Duty = 5/8 (62.5%)                            |
|       |          | 5h    | Duty = 6/8 (75.0%)                            |
|       |          | 6h    | Duty = 7/8 (87.5%)                            |
|       |          | 7h    | Reserved                                      |
| 7-5   | CHPFREQ  | 0-7h  | Chopping Clock Frequency                      |
|       |          | 0     | Divide by 1 (no prescale)                     |
|       |          | 1h    | Divide by 2                                   |
|       |          | 2h    | Divide by 3                                   |
|       |          | 3h-7h | Divide by 4 to divide by 8                    |
| 4-1   | OSHTWTH  | 0-Fh  | One-Shot Pulse Width                          |
|       |          | 0     | 1 × SYSCLKOUT/8 wide                          |
|       |          | 1h    | 2 × SYSCLKOUT/8 wide                          |
|       |          | 2h    | 3 × SYSCLKOUT/8 wide                          |
|       |          | 3h-Fh | 4 × SYSCLKOUT/8 wide to 16 × SYSCLKOUT/8 wide |
| 0     | CHPEN    | 0     | PWM-chopping Enable                           |
|       |          | 1     | Disable (bypass) PWM chopping function        |
|       |          | 1     | Enable chopping function                      |

#### 15.2.4.8 High-Resolution PWM Submodule Registers

[Table 15-92](#) lists the memory-mapped registers for the high-resolution PWM submodule. All other register offset addresses not listed in [Table 15-92](#) should be considered as reserved locations and the register contents should not be modified.

**Table 15-92. High-Resolution PWM Submodule Registers**

| Offset | Acronym | Register Description                       | Section                            |
|--------|---------|--|------------------------------------|
| 4h     | TBPHSHR | Time-Base Phase High-Resolution Register   | <a href="#">Section 15.2.4.8.1</a> |
| 10h    | CMPAHR  | Counter-Compare A High-Resolution Register | <a href="#">Section 15.2.4.8.2</a> |
| 40h    | HRCTL   | HRPWM Control Register                     | <a href="#">Section 15.2.4.8.3</a> |

#### 15.2.4.8.1 Time-Base Phase High-Resolution Register (TBPHSHR)

The time-base phase high-resolution register (TBPHSHR) is shown in [Figure 15-97](#) and described in [Table 15-93](#).

**Figure 15-97. Time-Base Phase High-Resolution Register (TBPHSHR)**

| 15     | 8 | 7 | 0        |
|--------|---|---|----------|
| TBPHSH |   |   | Reserved |
| R/W-0  |   |   | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-93. Time-Base Phase High-Resolution Register (TBPHSHR) Field Descriptions**

| Bit  | Field    | Value | Description                          |
|------|----------|-------|--------------------------------------|
| 15-8 | TBPHSH   | 0-FFh | Time-base phase high-resolution bits |
| 7-0  | Reserved | 0     | Reserved                             |

#### 15.2.4.8.2 Counter-Compare A High-Resolution Register (CMPAHR)

The counter-compare A high-resolution register (CMPAHR) is shown in [Figure 15-98](#) and described in [Table 15-94](#).

**Figure 15-98. Counter-Compare A High-Resolution Register (CMPAHR)**

| 15     | 8 | 7 | 0        |
|--------|---|---|----------|
| CMPAHR |   |   | Reserved |
| R/W-0  |   |   | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-94. Counter-Compare A High-Resolution Register (CMPAHR) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | CMPAHR   | 1-FFh | Compare A High-Resolution register bits for MEP step control. A minimum value of 1h is needed to enable HRPWM capabilities. Valid MEP range of operation 1-255h. |
| 7-0  | Reserved | 0     | Reserved   |

### 15.2.4.8.3 HRPWM Control Register (HRCTL)

The HRPWM control register (HRCTL) is shown in [Figure 15-99](#) and described in [Table 15-95](#).

**Figure 15-99. HRPWM Control Register (HRCTL)**

| 15       | 4        | 3          | 2       | 1 | 0 |
|----------|----------|------------|---------|---|---|
| Reserved | PULSESEL | DELBUSSSEL | DELMODE |   |   |
| R-0      | R/W-0    | R/W-0      | R/W-0   |   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-95. HRPWM Control Register (HRCTL) Field Descriptions**

| Bit  | Field      | Value                       | Description   |
|------|------------|-----------------------------|---|
| 15-4 | Reserved   | 0                           | Reserved  |
| 3    | PULSESEL   | 0<br>1                      | Pulse select bits. Selects which pulse to use for timing events in the HRPWM module:<br>0 Select CNT_zero pulse<br>1 Select PRD_eq pulse<br>Note: The user needs to select the pulse to match the selection in the EPWM module. If TBPHSHR bus is selected, then CNT_zero pulse should be used. If COMPAHR bus is selected, then it should match the bit setting of the CMPCTL[LOADMODE] bits in the EPWM module as follows:<br>0 CNT_zero pulse<br>1h PRD_eq pulse<br>2h CNT_zero or PRD_eq (should not use with HRPWM)<br>3h No loads (should not use with HRPWM) |
| 2    | DELBUSSSEL | 0<br>1                      | Delay Bus Select Bit: Selects which bus is used to select the delay for the PWM pulse:<br>0 Select CMPAHR(8) bus from compare module of EPWM (default on reset).<br>1 Select TBPHSHR(8) bus from time base module.  |
| 1-0  | DELMODE    | 0-3h<br>0<br>1h<br>2h<br>3h | Delay Mode Bits: Selects which edge of the PWM pulse the delay is inserted:<br>0 No delay inserted (default on reset)<br>1h Delay inserted rising edge<br>2h Delay inserted falling edge<br>3h Delay inserted on both edges<br>Note: When DELMODE = 0,0, the HRCALM[CALMODE] bits are ignored and the delay line is in by-pass mode. Additionally, DLYIN is connected to CALIN and a continuous low value is fed to the delay line to minimize activity in the module.  |

## 15.3 Enhanced Capture (eCAP) Module

### 15.3.1 *Introduction*

#### 15.3.1.1 Purpose of the Peripheral

Uses for eCAP include:

- Sample rate measurements of audio inputs
- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

#### 15.3.1.2 Features

The eCAP module includes the following features:

- 32-bit time base counter
- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single shot capture of up to four event time-stamps
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- All above resources dedicated to a single input pin
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

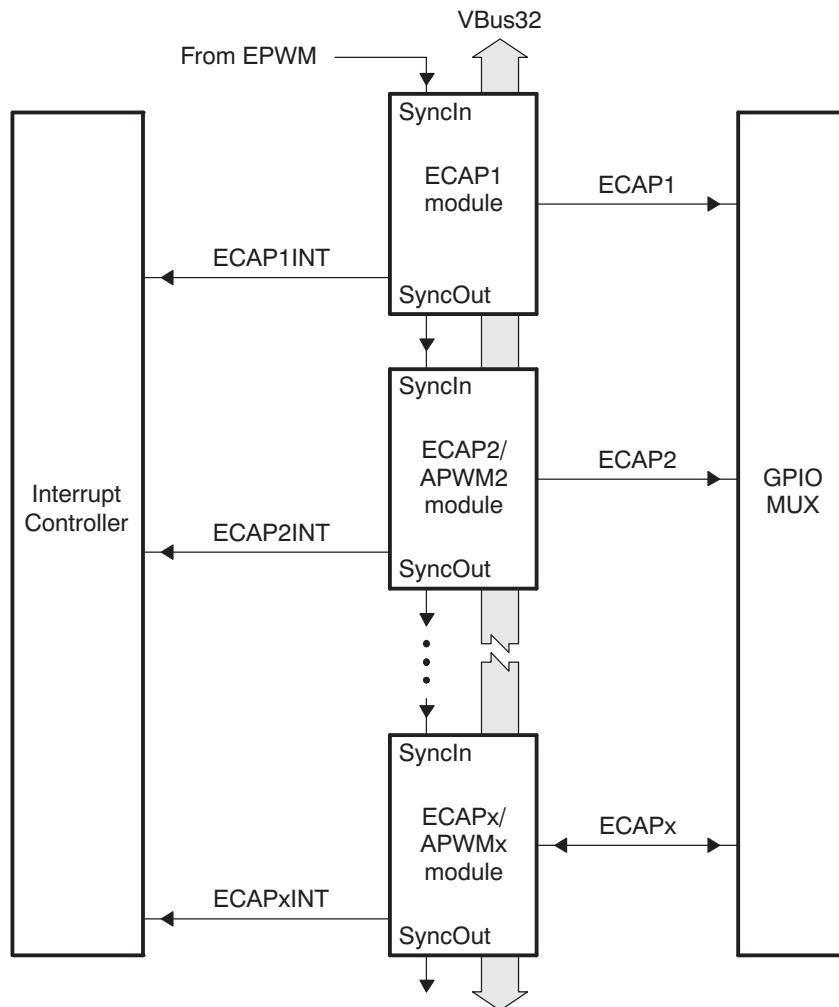
### 15.3.2 Functional Description

The eCAP module represents one complete capture channel that can be instantiated multiple times depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

- Dedicated input capture pin
- 32-bit time base counter
- 4 × 32-bit time-stamp capture registers (CAP1-CAP4)
- 4-stage sequencer (Modulo4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Independent edge polarity (rising/falling edge) selection for all 4 events
- Input capture signal prescaling (from 2-62)
- One-shot compare register (2 bits) to freeze captures after 1 to 4 time-stamp events
- Control for continuous time-stamp captures using a 4-deep circular buffer (CAP1-CAP4) scheme
- Interrupt capabilities on any of the 4 capture events

Multiple identical eCAP modules can be contained in a system as shown in [Figure 15-100](#). The number of modules is device-dependent and is based on target application needs. In this chapter, the letter x within a signal or module name is used to indicate a generic eCAP instance on a device.

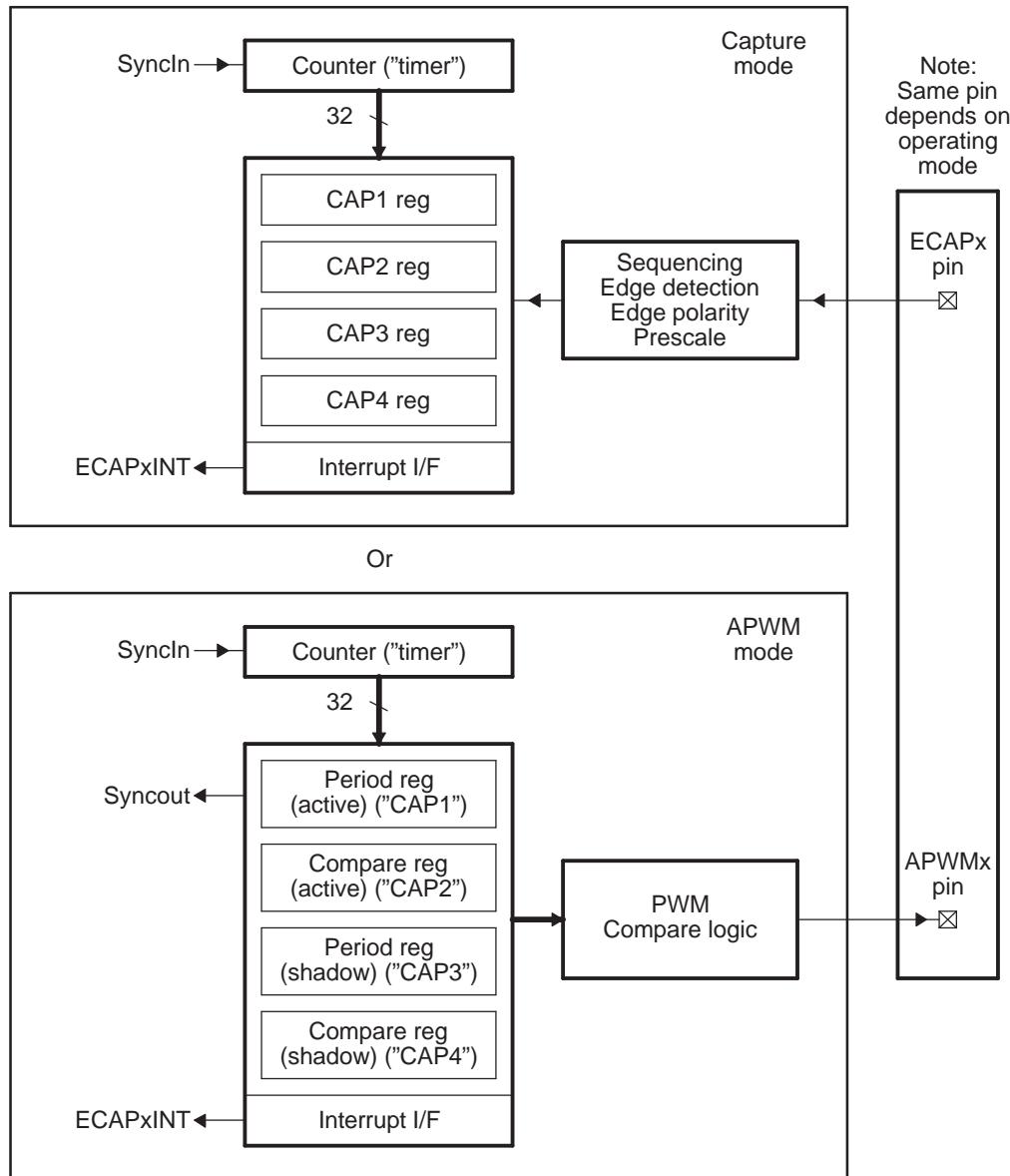
**Figure 15-100. Multiple eCAP Modules**



### 15.3.2.1 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32 bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. [Figure 15-101](#) is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

**Figure 15-101. Capture and APWM Modes of Operation**

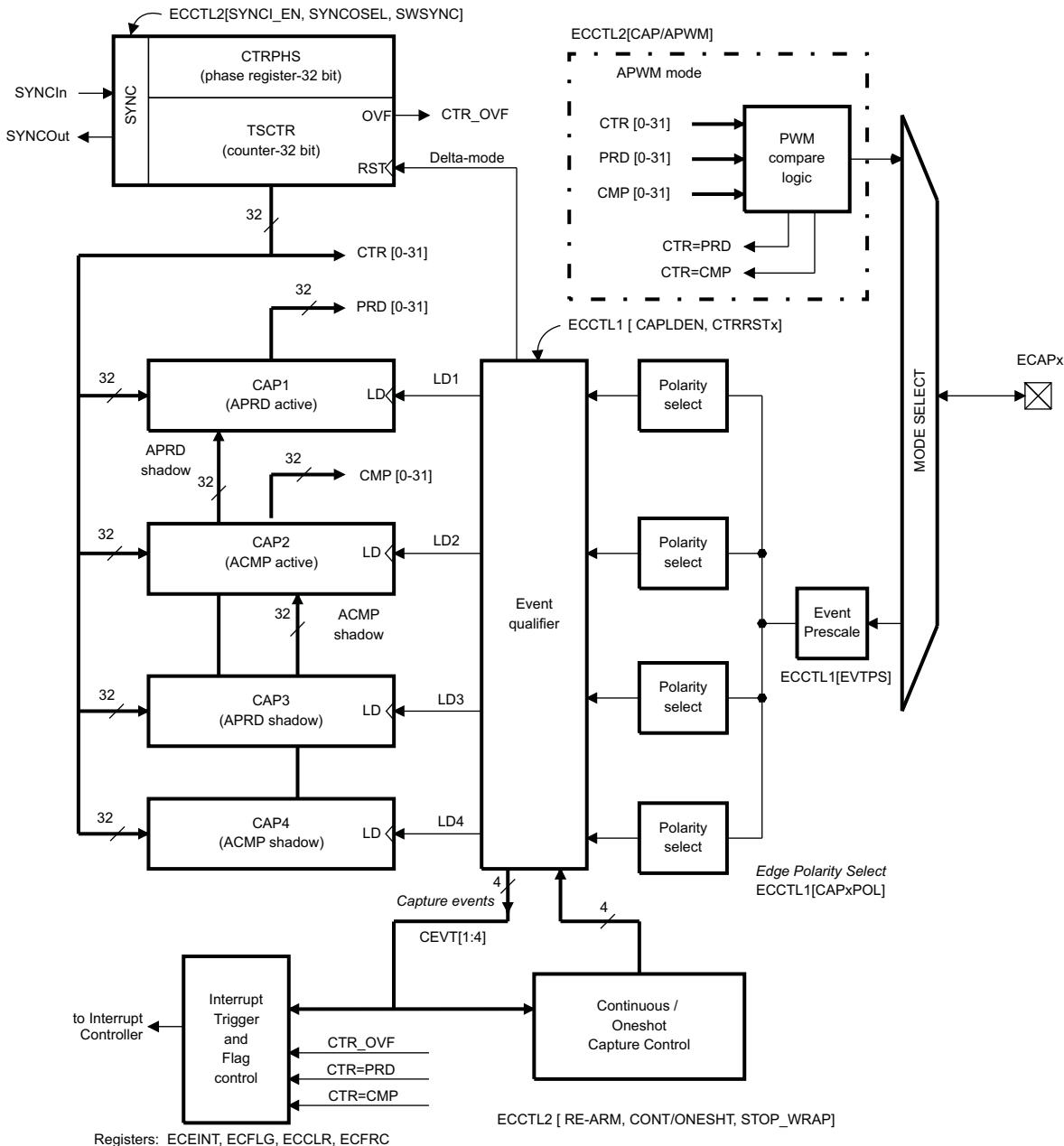


- (1) A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.
- (2) In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

### 15.3.2.2 Capture Mode Description

Figure 15-102 shows the various components that implement the capture function.

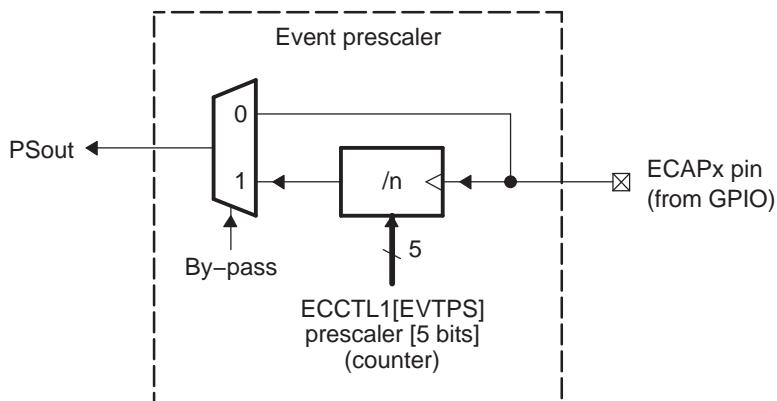
**Figure 15-102. Capture Function Diagram**



### 15.3.2.2.1 Event Prescaler

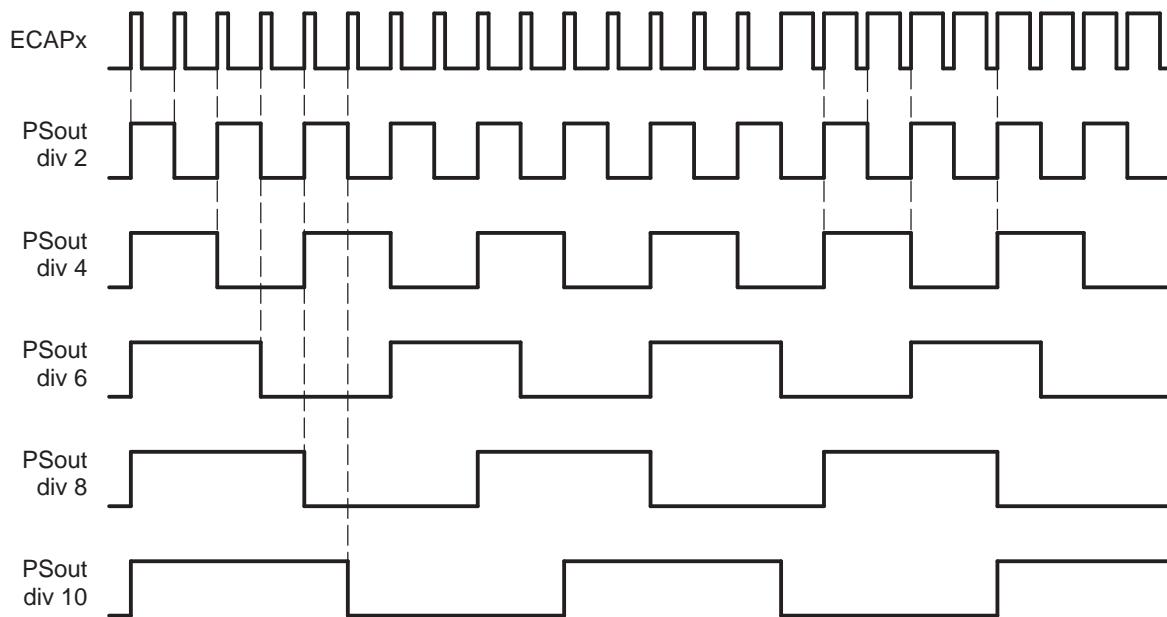
An input capture signal (pulse train) can be prescaled by  $N = 2-62$  (in multiples of 2) or can bypass the prescaler. This is useful when very high frequency signals are used as inputs. [Figure 15-103](#) shows a functional diagram and [Figure 15-104](#) shows the operation of the prescale function.

**Figure 15-103. Event Prescale Control**



- (1) When a prescale value of 1 is chosen (ECCTL1[13:9] = 0000) the input capture signal by-passes the prescale logic completely.

**Figure 15-104. Prescale Function Waveforms**



### 15.3.2.2.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection multiplexers are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAP $n$  register by the Mod4 counter. The CAP $n$  register is loaded on the falling edge.

### 15.3.2.2.3 Continuous/One-Shot Control

- The Mod4 (2 bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

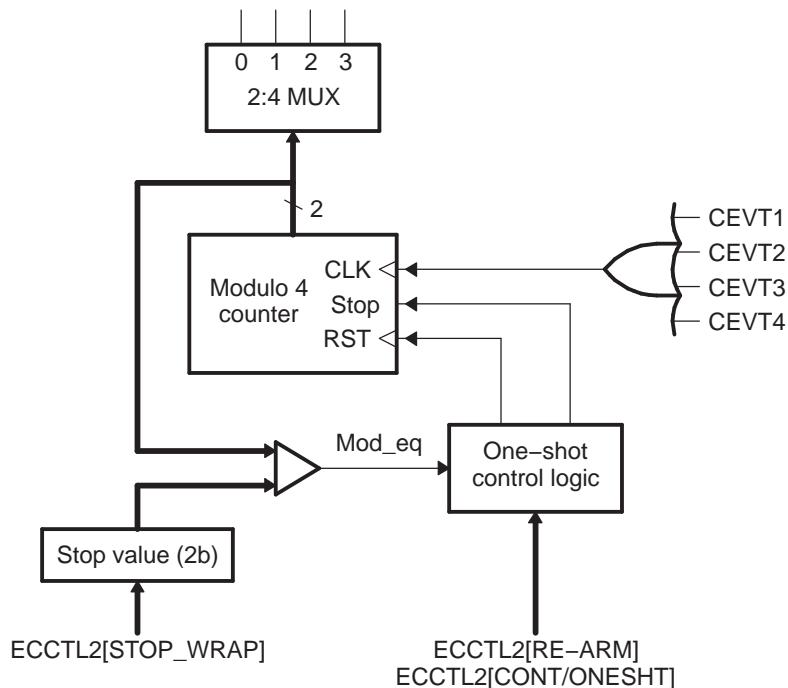
The continuous/one-shot block (Figure 15-105) controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (time-stamps).

Re-arming prepares the eCAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of CAP1-4 registers again, providing the CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0), the one-shot action is ignored, and capture values continue to be written to CAP1-4 in a circular buffer sequence.

**Figure 15-105. Continuous/One-shot Block Diagram**



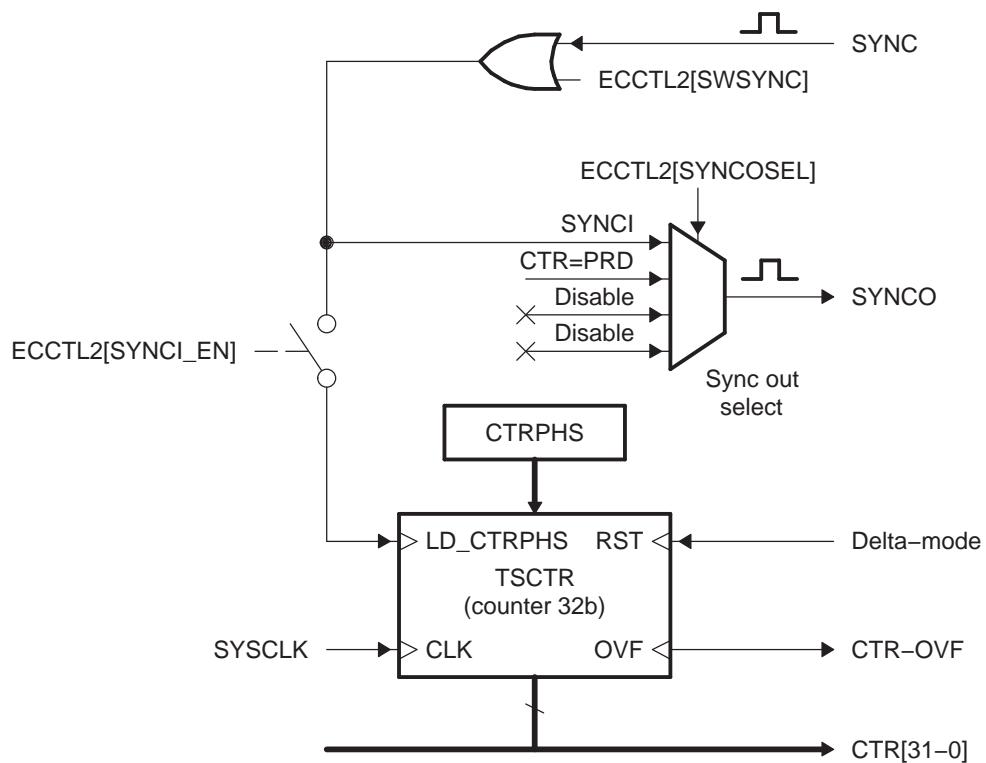
#### 15.3.2.2.4 32-Bit Counter and Phase Control

This counter (Figure 15-106) provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.

**Figure 15-106. Counter and Synchronization Block Diagram**



### 15.3.2.2.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via control bit CAPLDEN. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

### 15.3.2.2.6 Interrupt Control

An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP). See [Figure 15-107](#).

A counter overflow event (FFFF FFFFh->0000 0000h) is also provided as an interrupt source (CTROVF).

The capture events are edge and sequencer qualified (that is, ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the eCAPn module) going to the interrupt controller.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR = PRD, CTR = CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the interrupt controller only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

### 15.3.2.2.7 Shadow Load and Lockout Control

In capture mode, this logic inhibits (locks out) any shadow loading of CAP1 or CAP2 from APRD and ACMP registers, respectively.

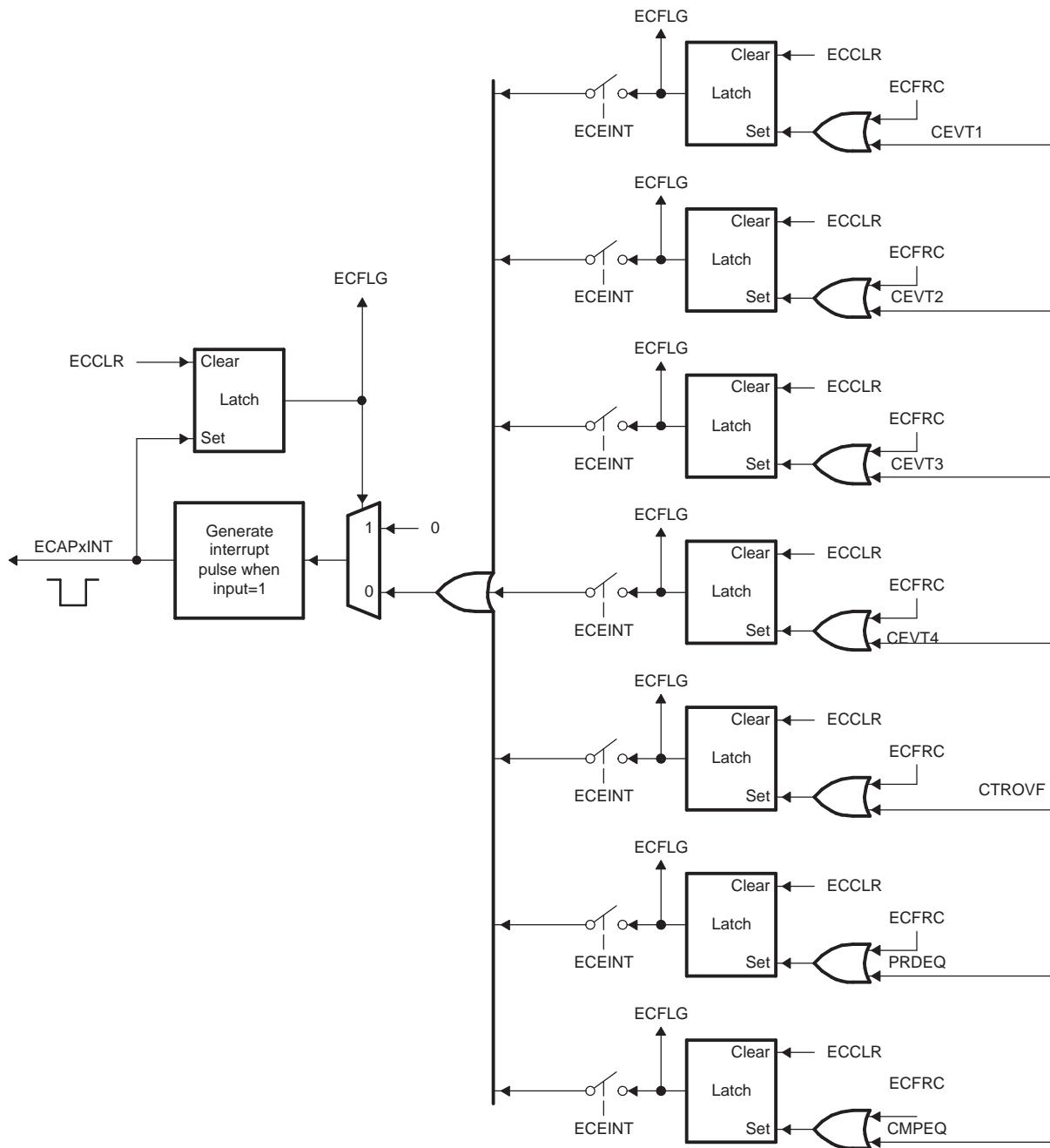
In APWM mode, shadow loading is active and two choices are permitted:

- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, CTR[31:0] = PRD[31:0]

---

**NOTE:** The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR = PRD, CTR = CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

---

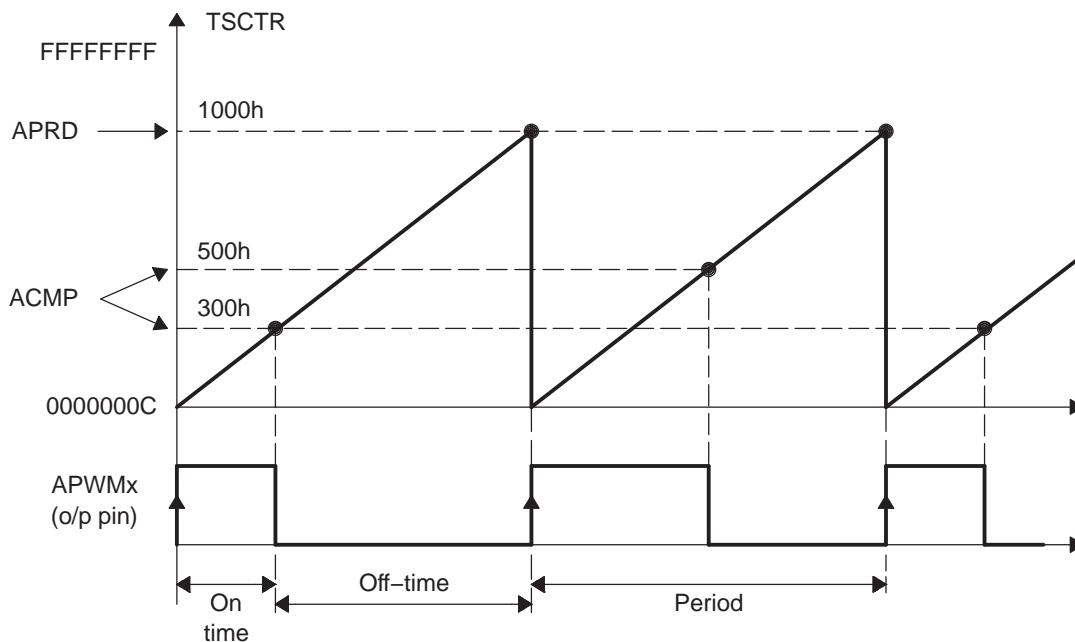
**Figure 15-107. Interrupts in eCAP Module**


### 15.3.2.2.8 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers either immediately upon a write, or on a CTR = PRD trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, you only need to use the shadow registers.

**Figure 15-108. PWM Waveform Details Of APWM Mode Operation**



The behavior of APWM active-high mode (APWMPOL == 0) is:

CMP = 0x00000000, output low for duration of period (0% duty)

CMP = 0x00000001, output high 1 cycle

CMP = 0x00000002, output high 2 cycles

CMP = PERIOD, output high except for 1 cycle (<100% duty)

CMP = PERIOD+1, output high for complete period (100% duty)

CMP > PERIOD+1, output high for complete period

The behavior of APWM active-low mode (APWMPOL == 1) is:

CMP = 0x00000000, output high for duration of period (0% duty)

CMP = 0x00000001, output low 1 cycle

CMP = 0x00000002, output low 2 cycles

CMP = PERIOD, output low except for 1 cycle (<100% duty)

CMP = PERIOD+1, output low for complete period (100% duty)

CMP > PERIOD+1, output low for complete period

### 15.3.3 Use Cases

The following sections will provide Applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, below are useful #defines which will help in the understanding of the examples.

```

// ECCTL1 ( ECAP Control Reg 1)
//=====
// CAPxPOL bits
#define EC_RISING          0x0
#define EC_FALLING         0x1

// CTRRSTx bits
#define EC_ABS_MODE        0x0
#define EC_DELTA_MODE      0x1

// PRESCALE bits
#define EC_BYPASS          0x0
#define EC_DIV1             0x0
#define EC_DIV2             0x1
#define EC_DIV4             0x2
#define EC_DIV6             0x3
#define EC_DIV8             0x4
#define EC_DIV10            0x5

// ECCTL2 ( ECAP Control Reg 2)
//=====
// CONT/ONESHOT bit
#define EC_CONTINUOUS       0x0
#define EC_ONESHOT          0x1

// STOPVALUE bit
#define EC_EVENT1           0x0
#define EC_EVENT2           0x1
#define EC_EVENT3           0x2
#define EC_EVENT4           0x3

// RE-ARM bit
#define EC_ARM              0x1

// TSCTRSTOP bit
#define EC_FREEZE           0x0
#define EC_RUN               0x1

// SYNCSEL bit
#define EC_SYNCIN            0x0
#define EC_CTR_PRD           0x1
#define EC_SYNC_DIS          0x2

// CAP/APWM mode bit
#define EC_CAP_MODE          0x0
#define EC_APWM_MODE         0x1

// APWMMPOL bit
#define EC_ACTV_HI           0x0
#define EC_ACTV_LO           0x1

// Generic
#define EC_DISABLE           0x0
#define EC_ENABLE            0x1
#define EC_FORCE             0x1

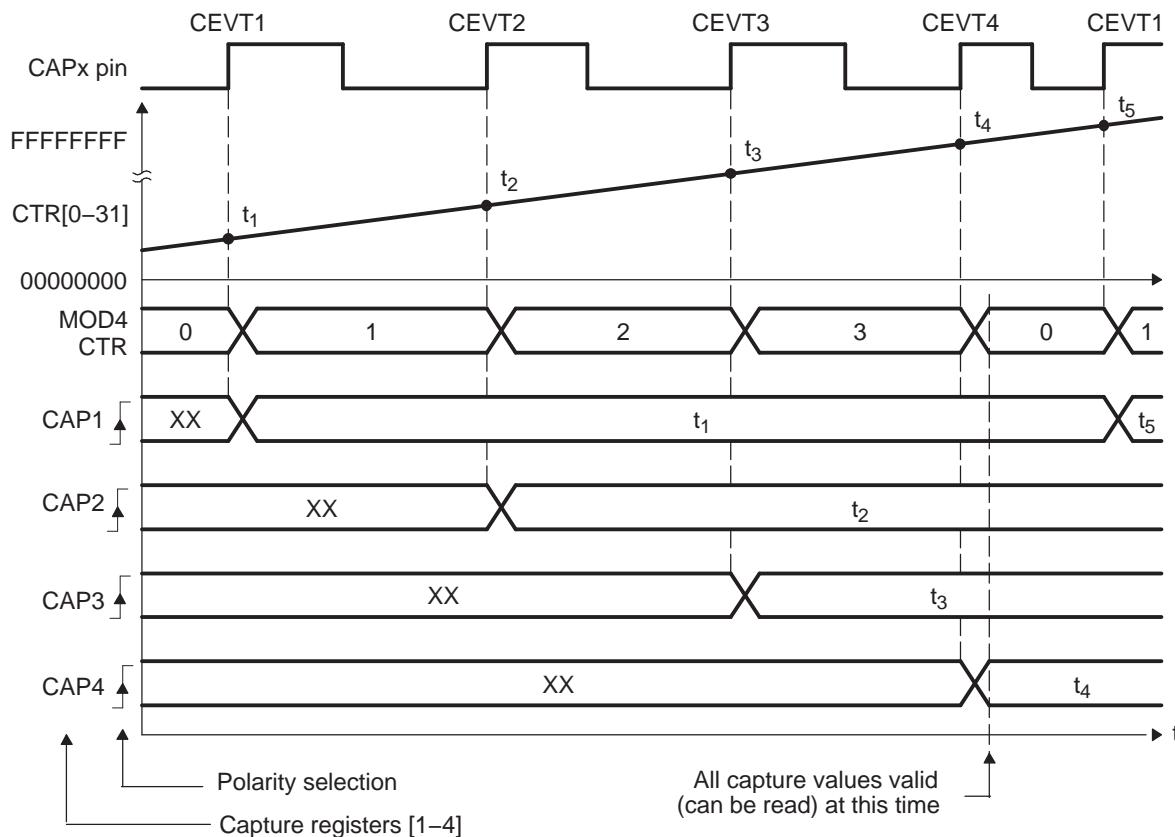
```

### 15.3.3.1 Absolute Time-Stamp Operation Rising Edge Trigger Example

Figure 15-109 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFF FFFFh (maximum value), it wraps around to 0000 0000h (not shown in Figure 15-109), if this occurs, the CTROVF (counter overflow) flag is set, and an interrupt (if enabled) occurs, CTROVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. Captured time-stamps are valid at the point indicated by the diagram, after the 4th event, hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPn registers.

**Figure 15-109. Capture Sequence for Absolute Time-Stamp, Rising Edge Detect**



**Table 15-96. ECAP Initialization for CAP Mode Absolute Time, Rising Edge Trigger**

| Register | Bit        | Value         |
|----------|------------|---------------|
| ECCTL1   | CAP1POL    | EC_RISING     |
| ECCTL1   | CAP2POL    | EC_RISING     |
| ECCTL1   | CAP3POL    | EC_RISING     |
| ECCTL1   | CAP4POL    | EC_RISING     |
| ECCTL1   | CTRRST1    | EC_ABS_MODE   |
| ECCTL1   | CTRRST2    | EC_ABS_MODE   |
| ECCTL1   | CTRRST3    | EC_ABS_MODE   |
| ECCTL1   | CTRRST4    | EC_ABS_MODE   |
| ECCTL1   | CAPLDEN    | EC_ENABLE     |
| ECCTL1   | PRESCALE   | EC_DIV1       |
| ECCTL2   | CAP_APWM   | EC_CAP_MODE   |
| ECCTL2   | CONT_ONESH | EC_CONTINUOUS |
| ECCTL2   | SYNCO_SEL  | EC_SYNCO_DIS  |
| ECCTL2   | SYNCI_EN   | EC_DISABLE    |
| ECCTL2   | TSCTRSTOP  | EC_RUN        |

**Example 15-9. Code Snippet for CAP Mode Absolute Time, Rising Edge Trigger**

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

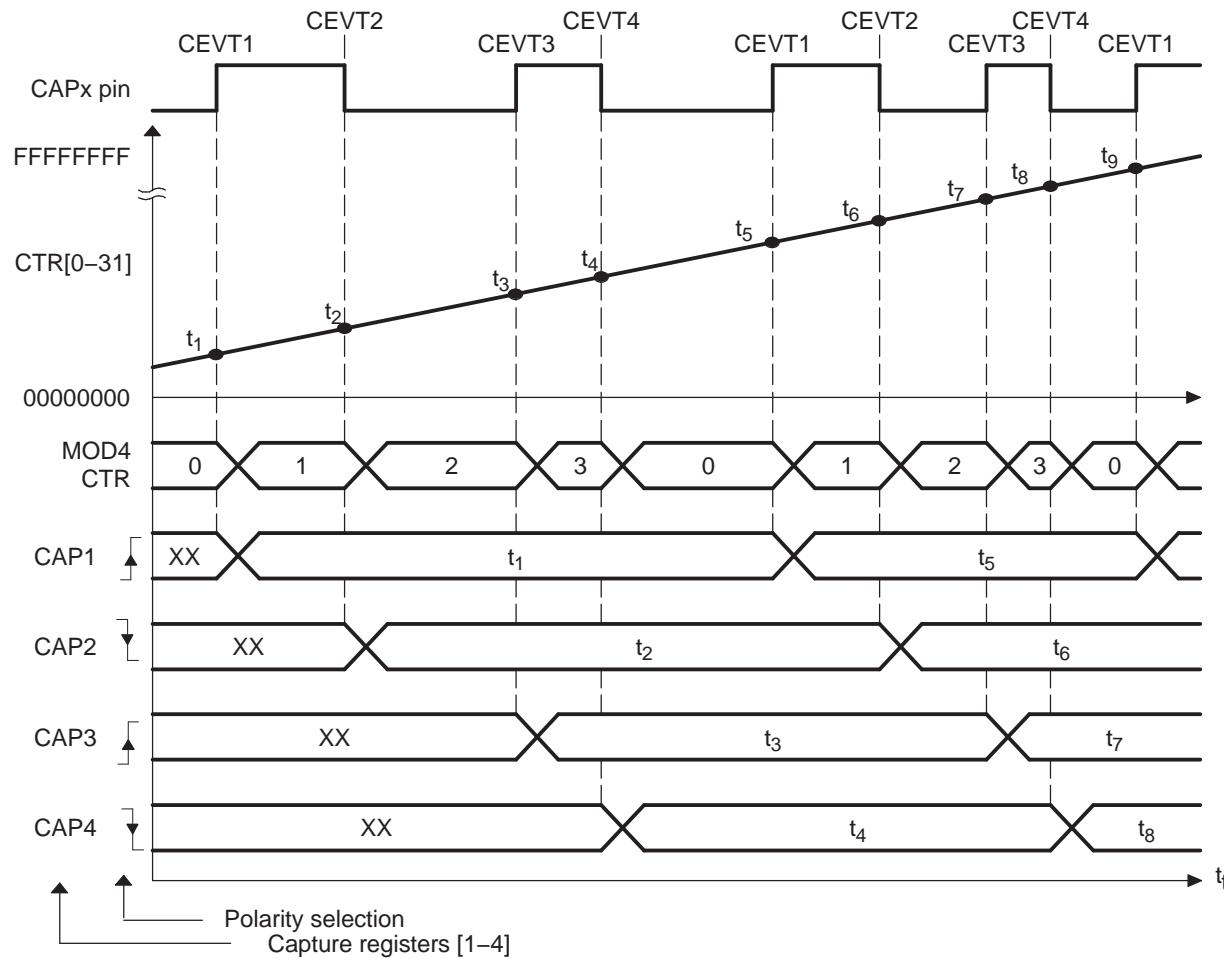
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;           // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;           // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;           // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;           // Fetch Time-Stamp captured at t4

Period1 = TSt2-TSt1;             // Calculate 1st period
Period2 = TSt3-TSt2;             // Calculate 2nd period
Period3 = TSt4-TSt3;             // Calculate 3rd period
```

### 15.3.3.2 Absolute Time-Stamp Operation Rising and Falling Edge Trigger Example

In Figure 15-110 the eCAP operating mode is almost the same as in the previous section except capture events are qualified as either rising or falling edge, this now gives both period and duty cycle information: Period1 =  $t_3 - t_1$ , Period2 =  $t_5 - t_3$ , ...etc. Duty Cycle1 (on-time %) =  $(t_2 - t_1) / \text{Period1} \times 100\%$ , etc. Duty Cycle1 (off-time %) =  $(t_3 - t_2) / \text{Period1} \times 100\%$ , etc.

**Figure 15-110. Capture Sequence for Absolute Time-Stamp, Rising and Falling Edge Detect**



**Table 15-97. ECAP Initialization for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

| Register | Bit        | Value         |
|----------|------------|---------------|
| ECCTL1   | CAP1POL    | EC_RISING     |
| ECCTL1   | CAP2POL    | EC_FALLING    |
| ECCTL1   | CAP3POL    | EC_RISING     |
| ECCTL1   | CAP4POL    | EC_FALLING    |
| ECCTL1   | CTRRST1    | EC_ABS_MODE   |
| ECCTL1   | CTRRST2    | EC_ABS_MODE   |
| ECCTL1   | CTRRST3    | EC_ABS_MODE   |
| ECCTL1   | CTRRST4    | EC_ABS_MODE   |
| ECCTL1   | CAPLDEN    | EC_ENABLE     |
| ECCTL1   | PRESCALE   | EC_DIV1       |
| ECCTL2   | CAP_APWM   | EC_CAP_MODE   |
| ECCTL2   | CONT_ONESH | EC_CONTINUOUS |
| ECCTL2   | SYNCO_SEL  | EC_SYNCO_DIS  |
| ECCTL2   | SYNCI_EN   | EC_DISABLE    |
| ECCTL2   | TSCTRSTOP  | EC_RUN        |

**Example 15-10. Code Snippet for CAP Mode Absolute Time, Rising and Falling Edge Trigger**

```
// Code snippet for CAP mode Absolute Time, Rising & Falling edge triggers

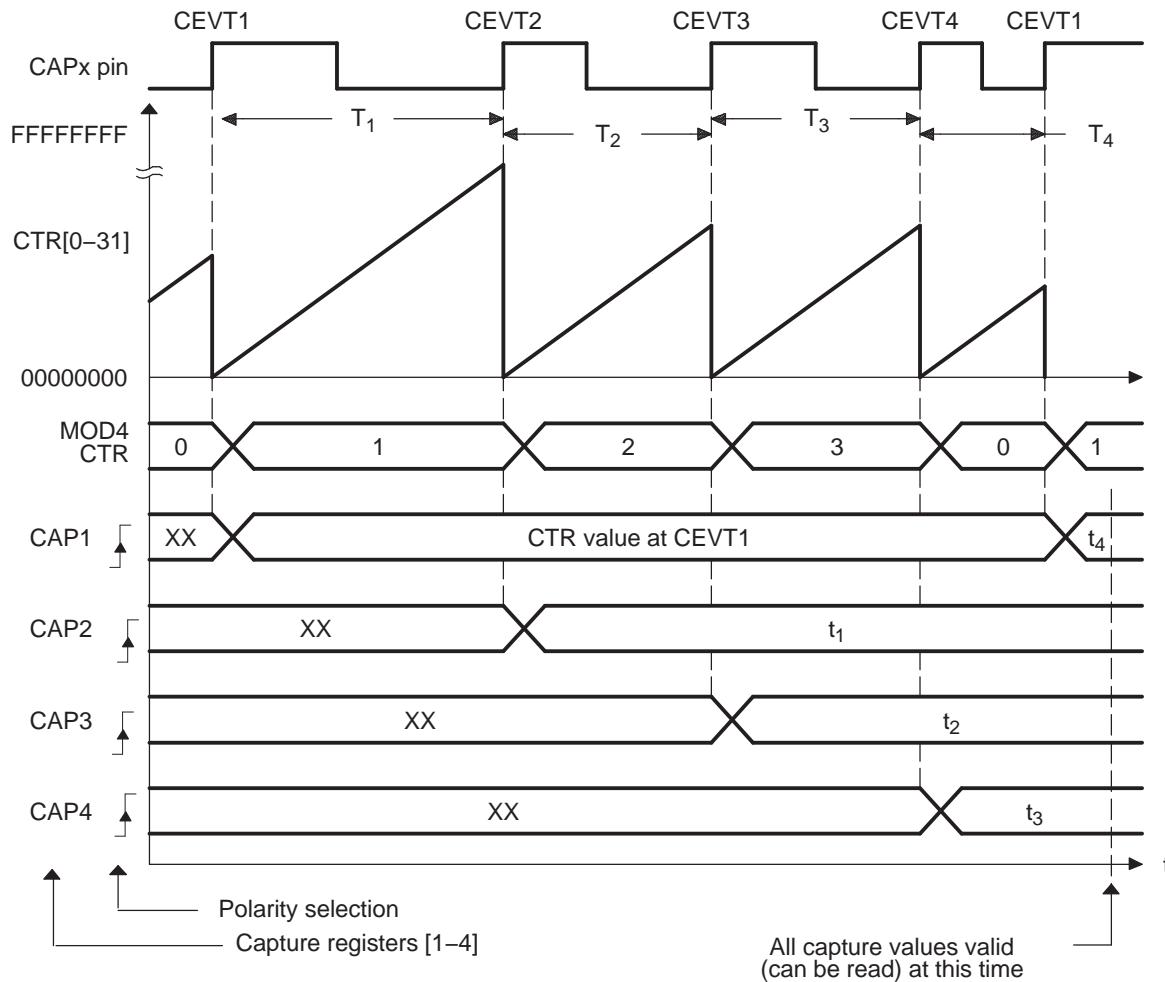
// Run Time ( e.g. CEVT4 triggered ISR call)
//=====
TSt1 = ECAPxRegs.CAP1;           // Fetch Time-Stamp captured at t1
TSt2 = ECAPxRegs.CAP2;           // Fetch Time-Stamp captured at t2
TSt3 = ECAPxRegs.CAP3;           // Fetch Time-Stamp captured at t3
TSt4 = ECAPxRegs.CAP4;           // Fetch Time-Stamp captured at t4

Period1 = TSt3-TSt1;             // Calculate 1st period
DutyOnTime1 = TSt2-TSt1;         // Calculate On time
DutyOffTime1 = TSt3-TSt2;        // Calculate Off time
```

### 15.3.3.3 Time Difference (Delta) Operation Rising Edge Trigger Example

Figure 15-111 shows how the eCAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (time-stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFF FFFFh (maximum value), before the next event, it wraps around to 0000 0000h and continues, a CNTOVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. The advantage of Delta-time Mode is that the CAPn contents directly give timing data without the need for CPU calculations: Period1 =  $T_1$ , Period2 =  $T_2$ ,...etc. As shown in Figure 15-111, the CEVT1 event is a good trigger point to read the timing data,  $T_1$ ,  $T_2$ ,  $T_3$ ,  $T_4$  are all valid here.

**Figure 15-111. Capture Sequence for Delta Mode Time-Stamp, Rising Edge Detect**



**Table 15-98. ECAP Initialization for CAP Mode Delta Time, Rising Edge Trigger**

| Register | Bit        | Value         |
|----------|------------|---------------|
| ECCTL1   | CAP1POL    | EC_RISING     |
| ECCTL1   | CAP2POL    | EC_RISING     |
| ECCTL1   | CAP3POL    | EC_RISING     |
| ECCTL1   | CAP4POL    | EC_RISING     |
| ECCTL1   | CTRRST1    | EC_DELTA_MODE |
| ECCTL1   | CTRRST2    | EC_DELTA_MODE |
| ECCTL1   | CTRRST3    | EC_DELTA_MODE |
| ECCTL1   | CTRRST4    | EC_DELTA_MODE |
| ECCTL1   | CAPLDEN    | EC_ENABLE     |
| ECCTL1   | PRESCALE   | EC_DIV1       |
| ECCTL2   | CAP_APWM   | EC_CAP_MODE   |
| ECCTL2   | CONT_ONESH | EC_CONTINUOUS |
| ECCTL2   | SYNCO_SEL  | EC_SYNCO_DIS  |
| ECCTL2   | SYNCI_EN   | EC_DISABLE    |
| ECCTL2   | TSCTRSTOP  | EC_RUN        |

**Example 15-11. Code Snippet for CAP Mode Delta Time, Rising Edge Trigger**

```
// Code snippet for CAP mode Delta Time, Rising edge trigger

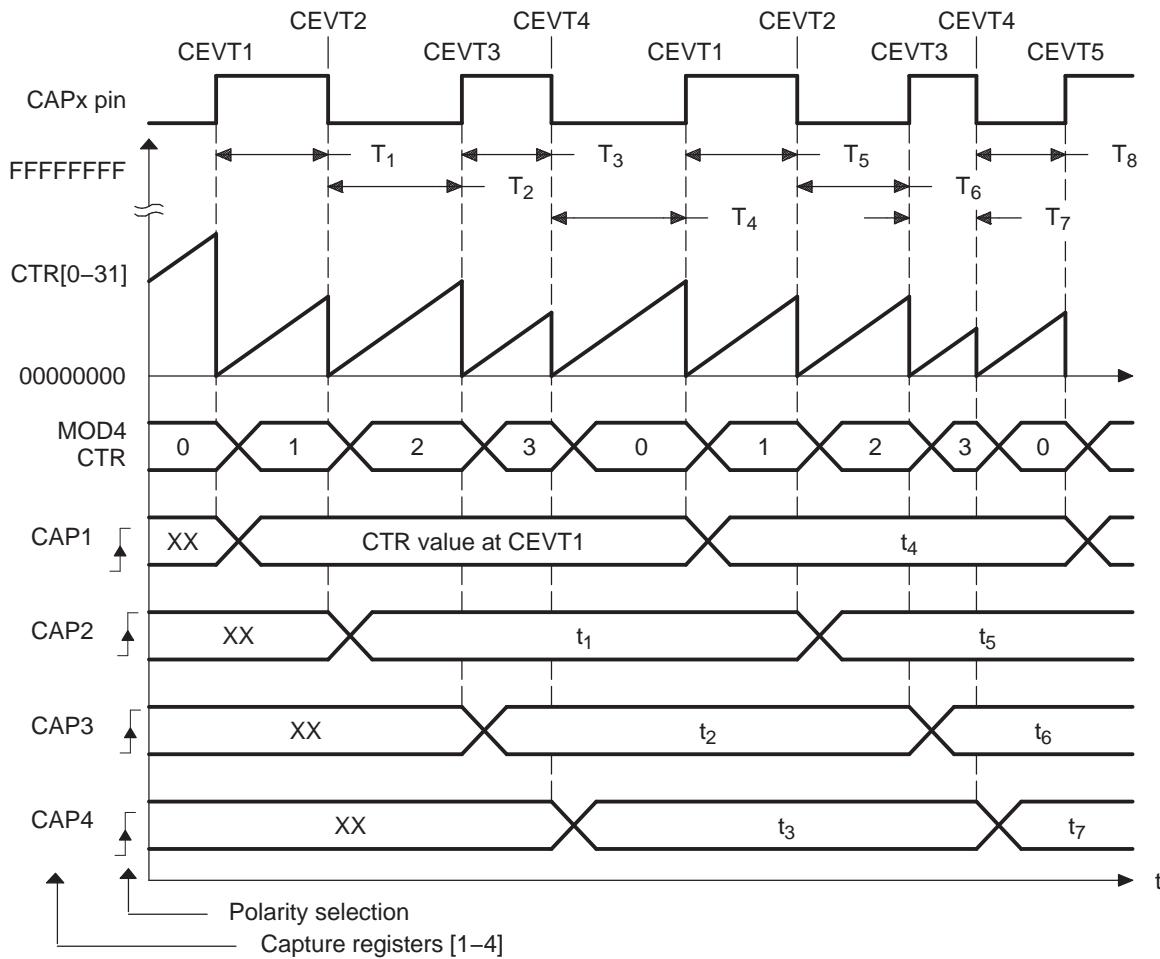
// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Period value.
Period4 = ECAPxRegs.CAP1;      // Fetch Time-Stamp captured at T1
Period1 = ECAPxRegs.CAP2;      // Fetch Time-Stamp captured at T2
Period2 = ECAPxRegs.CAP3;      // Fetch Time-Stamp captured at T3
Period3 = ECAPxRegs.CAP4;      // Fetch Time-Stamp captured at T4
```

#### 15.3.3.4 Time Difference (Delta) Operation Rising and Falling Edge Trigger Example

In Figure 15-112 the eCAP operating mode is almost the same as in previous section except Capture events are qualified as either Rising or Falling edge, this now gives both Period and Duty cycle information: Period1 =  $T_1 + T_2$ , Period2 =  $T_3 + T_4$ , ...etc Duty Cycle1 (on-time %) =  $T_1 / \text{Period1} \times 100\%$ , etc Duty Cycle1 (off-time %) =  $T_2 / \text{Period1} \times 100\%$ , etc

During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, that is, during run-time, only the shadow registers must be used.

**Figure 15-112. Capture Sequence for Delta Mode Time-Stamp, Rising and Falling Edge Detect**



**Table 15-99. ECAP Initialization for CAP Mode Delta Time, Rising and Falling Edge Triggers**

| Register | Bit        | Value         |
|----------|------------|---------------|
| ECCTL1   | CAP1POL    | EC_RISING     |
| ECCTL1   | CAP2POL    | EC_FALLING    |
| ECCTL1   | CAP3POL    | EC_RISING     |
| ECCTL1   | CAP4POL    | EC_FALLING    |
| ECCTL1   | CTRRST1    | EC_DELTA_MODE |
| ECCTL1   | CTRRST2    | EC_DELTA_MODE |
| ECCTL1   | CTRRST3    | EC_DELTA_MODE |
| ECCTL1   | CTRRST4    | EC_DELTA_MODE |
| ECCTL1   | CAPLDEN    | EC_ENABLE     |
| ECCTL1   | PRESCALE   | EC_DIV1       |
| ECCTL2   | CAP_APWM   | EC_CAP_MODE   |
| ECCTL2   | CONT_ONESH | EC_CONTINUOUS |
| ECCTL2   | SYNCO_SEL  | EC_SYNCO_DIS  |
| ECCTL2   | SYNCI_EN   | EC_DISABLE    |
| ECCTL2   | TSCTRSTOP  | EC_RUN        |

**Example 15-12. Code Snippet for CAP Mode Delta Time, Rising and Falling Edge Triggers**

```
// Code snippet for CAP mode Delta Time, Rising and Falling edge triggers

// Run Time ( e.g. CEVT1 triggered ISR call)
//=====
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 = ECAPxRegs.CAP2;      // Fetch Time-Stamp captured at T2
DutyOffTime1 = ECAPxRegs.CAP3;     // Fetch Time-Stamp captured at T3
DutyOnTime2 = ECAPxRegs.CAP4;      // Fetch Time-Stamp captured at T4
DutyOffTime2 = ECAPxRegs.CAP1;     // Fetch Time-Stamp captured at T1

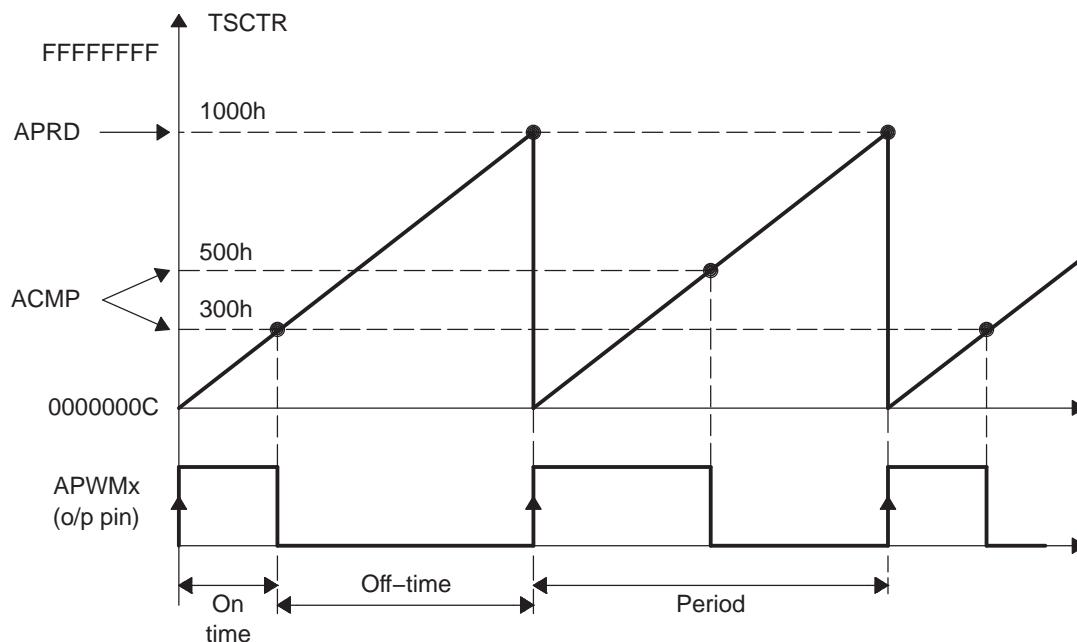
Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```

### 15.3.3.5 Application of the APWM Mode

#### 15.3.3.5.1 Simple PWM Generation (Independent Channel/s) Example

In this example, the eCAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWM<sub>n</sub>. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time.

**Figure 15-113. PWM Waveform Details of APWM Mode Operation**



**Table 15-100. ECAP Initialization for APWM Mode**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 0x1000       |
| CTRPHS   | CTRPHS    | 0x0          |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCI_EN  | EC_DISABLE   |
| ECCTL2   | SYNCO_SEL | EC_SYNCO_DIS |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Example 15-13. Code Snippet for APWM Mode**

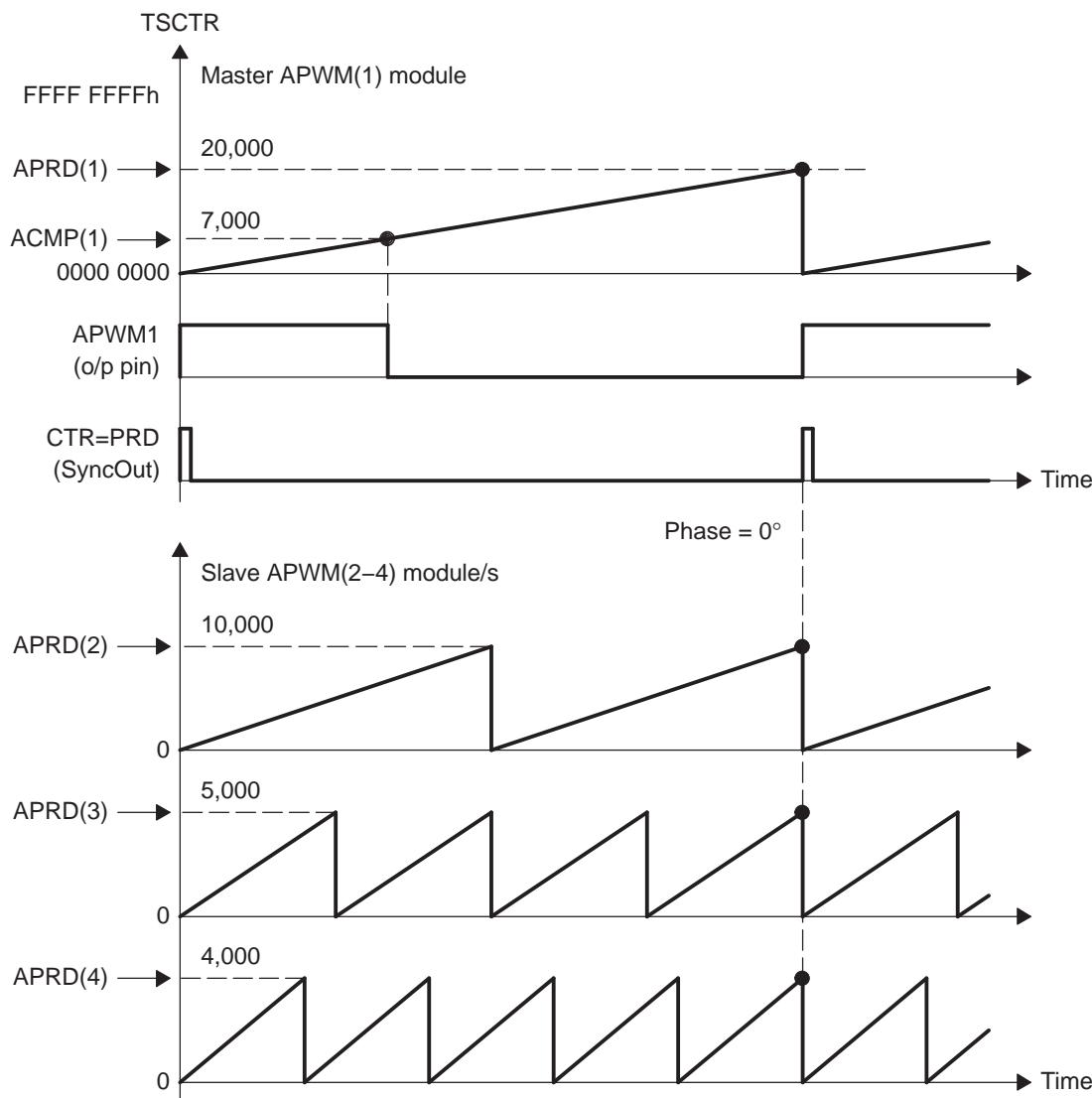
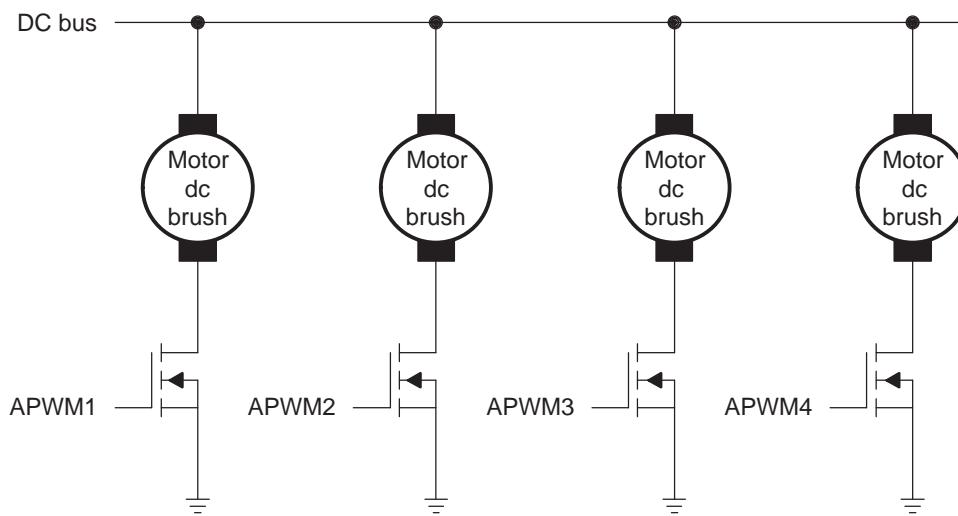
```
// Code snippet for APWM mode Example 1

// Run Time (Instant 1, e.g. ISR call)
//=====
ECAPxRegs.CAP2 = 0x300;      // Set Duty cycle i.e. compare value

// Run Time (Instant 2, e.g. another ISR call)
//=====
ECAPxRegs.CAP2 = 0x500;      // Set Duty cycle i.e. compare value
```

**15.3.3.5.2 Multichannel PWM Generation with Synchronization Example**

Figure 15-114 takes advantage of the synchronization feature between eCAP modules. Here 4 independent PWM channels are required with different frequencies, but at integer multiples of each other to avoid "beat" frequencies. Hence one eCAP module is configured as the Master and the remaining 3 are Slaves all receiving their synch pulse (CTR = PRD) from the master. Note the Master is chosen to have the lower frequency ( $F_1 = 1/20,000$ ) requirement. Here Slave2 Freq =  $2 \times F_1$ , Slave3 Freq =  $4 \times F_1$  and Slave4 Freq =  $5 \times F_1$ . Note here values are in decimal notation. Also, only the APWM1 output waveform is shown.

**Figure 15-114. Multichannel PWM Example Using 4 eCAP Modules**


**Table 15-101. ECAP1 Initialization for Multichannel PWM Generation with Synchronization**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 20000        |
| CTRPHS   | CTRPHS    | 0            |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCI_EN  | EC_DISABLE   |
| ECCTL2   | SYNCO_SEL | EC_CTR_PRD   |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Table 15-102. ECAP2 Initialization for Multichannel PWM Generation with Synchronization**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 10000        |
| CTRPHS   | CTRPHS    | 0            |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCI_EN  | EC_ENABLE    |
| ECCTL2   | SYNCO_SEL | EC_SYNCI     |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Table 15-103. ECAP3 Initialization for Multichannel PWM Generation with Synchronization**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 5000         |
| CTRPHS   | CTRPHS    | 0            |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCI_EN  | EC_ENABLE    |
| ECCTL2   | SYNCO_SEL | EC_SYNCI     |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Table 15-104. ECAP4 Initialization for Multichannel PWM Generation with Synchronization**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 4000         |
| CTRPHS   | CTRPHS    | 0            |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCI_EN  | EC_ENABLE    |
| ECCTL2   | SYNCO_SEL | EC_SYNCO_DIS |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

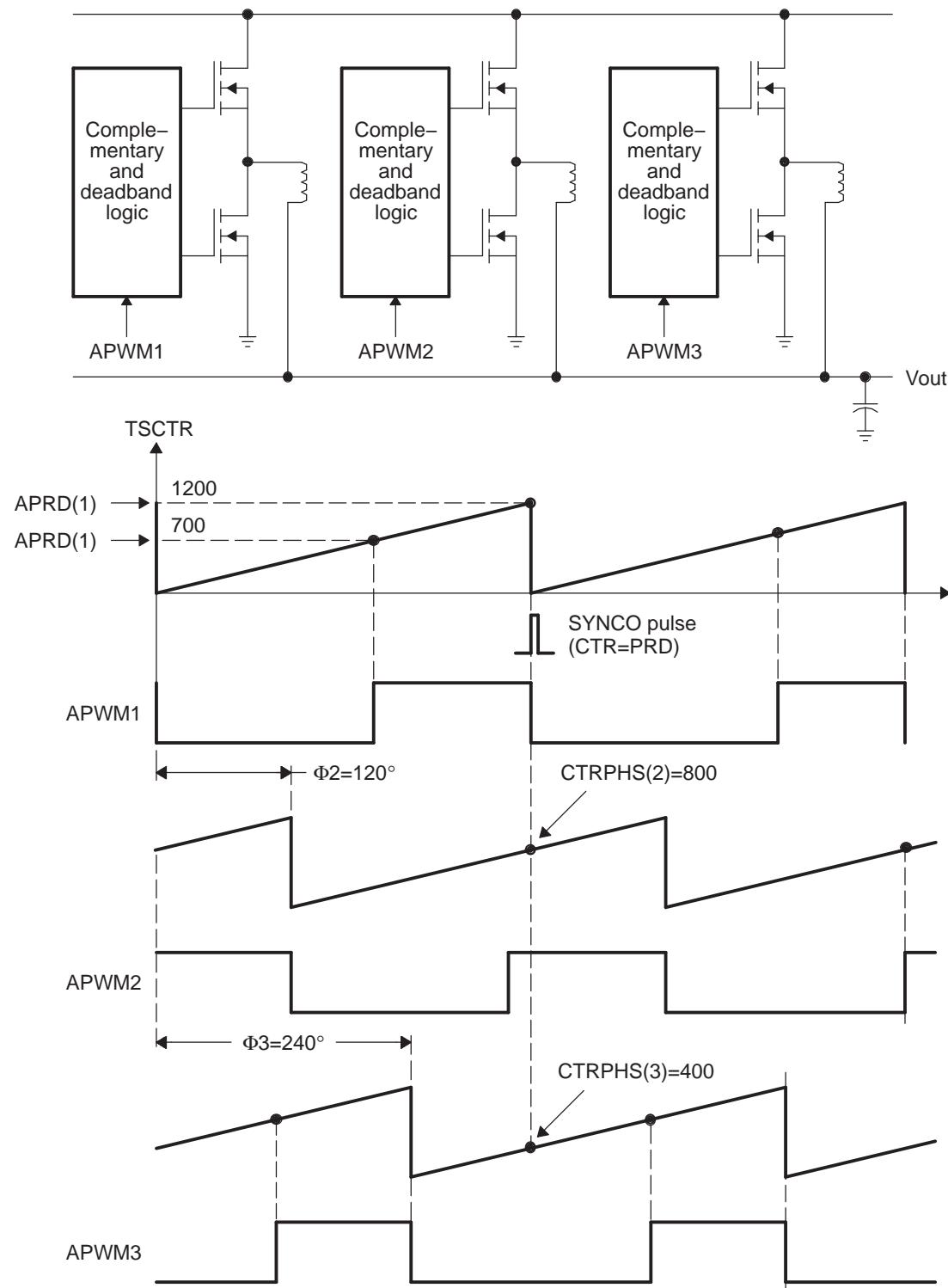
***Example 15-14. Code Snippet for Multichannel PWM Generation with Synchronization***

```
// Code snippet for APWM mode Example 2

// Run Time (Note: Example execution of one run-time instant)
//=====
ECAP1Regs.CAP2 = 7000;      // Set Duty cycle i.e., compare value = 7000
ECAP2Regs.CAP2 = 2000;      // Set Duty cycle i.e., compare value = 2000
ECAP3Regs.CAP2 = 550;       // Set Duty cycle i.e., compare value = 550
ECAP4Regs.CAP2 = 6500;      // Set Duty cycle i.e., compare value = 6500
```

***15.3.3.5.3 Multichannel PWM Generation with Phase Control Example***

In [Figure 15-115](#), the Phase control feature of the APWM mode is used to control a 3 phase Interleaved DC/DC converter topology. This topology requires each phase to be off-set by 120° from each other. Hence if “Leg” 1 (controlled by APWM1) is the reference Leg (or phase), that is, 0°, then Leg 2 need 120° off-set and Leg 3 needs 240° off-set. The waveforms in [Figure 15-115](#) show the timing relationship between each of the phases (Legs). Note eCAP1 module is the Master and issues a sync out pulse to the slaves (modules 2, 3) whenever TSCTR = Period value.

**Figure 15-115. Multiphase (channel) Interleaved PWM Example Using 3 eCAP Modules**


**Table 15-105. ECAP1 Initialization for Multichannel PWM Generation with Phase Control**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 1200         |
| CTRPHS   | CTRPHS    | 0            |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCl_EN  | EC_DISABLE   |
| ECCTL2   | SYNCO_SEL | EC_CTR_PRD   |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Table 15-106. ECAP2 Initialization for Multichannel PWM Generation with Phase Control**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 1200         |
| CTRPHS   | CTRPHS    | 800          |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCl_EN  | EC_ENABLE    |
| ECCTL2   | SYNCO_SEL | EC_SYNCl     |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Table 15-107. ECAP3 Initialization for Multichannel PWM Generation with Phase Control**

| Register | Bit       | Value        |
|----------|-----------|--------------|
| CAP1     | CAP1      | 1200         |
| CTRPHS   | CTRPHS    | 400          |
| ECCTL2   | CAP_APWM  | EC_APWM_MODE |
| ECCTL2   | APWMPOL   | EC_ACTV_HI   |
| ECCTL2   | SYNCl_EN  | EC_ENABLE    |
| ECCTL2   | SYNCO_SEL | EC_SYNCO_DIS |
| ECCTL2   | TSCTRSTOP | EC_RUN       |

**Example 15-15. Code Snippet for Multichannel PWM Generation with Phase Control**

```
// Code snippet for APWM mode Example 3

// Run Time (Note: Example execution of one run-time instant)
//=====
// All phases are set to the same duty cycle
ECAP1Regs.CAP2 = 700;      // Set Duty cycle i.e. compare value = 700
ECAP2Regs.CAP2 = 700;      // Set Duty cycle i.e. compare value = 700
ECAP3Regs.CAP2 = 700;      // Set Duty cycle i.e. compare value = 700
```

### 15.3.4 Registers

**Table 15-108** shows the eCAP module control and status register set. All 32-bit registers are aligned on even address boundaries and are organized in little-endian mode. The 16 least-significant bits of a 32-bit register are located on lowest address (even address).

---

**NOTE:** In APWM mode, writing to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

---

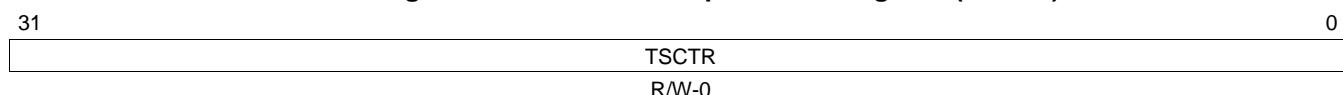
**Table 15-108. Control and Status Register Set**

| Offset | Acronym | Description                         | Size (x16) | Section                           |
|--------|---------|-------------------------------------|------------|-----------------------------------|
| 0h     | TSCTR   | Time-Stamp Counter Register         | 2          | <a href="#">Section 15.3.4.1</a>  |
| 4h     | CTRPHS  | Counter Phase Offset Value Register | 2          | <a href="#">Section 15.3.4.2</a>  |
| 8h     | CAP1    | Capture 1 Register                  | 2          | <a href="#">Section 15.3.4.3</a>  |
| Ch     | CAP2    | Capture 2 Register                  | 2          | <a href="#">Section 15.3.4.4</a>  |
| 10h    | CAP3    | Capture 3 Register                  | 2          | <a href="#">Section 15.3.4.5</a>  |
| 14h    | CAP4    | Capture 4 Register                  | 2          | <a href="#">Section 15.3.4.6</a>  |
| 28h    | ECCTL1  | Capture Control Register 1          | 1          | <a href="#">Section 15.3.4.7</a>  |
| 2Ah    | ECCTL2  | Capture Control Register 2          | 1          | <a href="#">Section 15.3.4.8</a>  |
| 2Ch    | ECEINT  | Capture Interrupt Enable Register   | 1          | <a href="#">Section 15.3.4.9</a>  |
| 2Eh    | ECFLG   | Capture Interrupt Flag Register     | 1          | <a href="#">Section 15.3.4.10</a> |
| 30h    | ECCLR   | Capture Interrupt Clear Register    | 1          | <a href="#">Section 15.3.4.11</a> |
| 32h    | ECFRC   | Capture Interrupt Force Register    | 1          | <a href="#">Section 15.3.4.12</a> |
| 5Ch    | REVID   | Revision ID Register                | 2          | <a href="#">Section 15.3.4.13</a> |

#### 15.3.4.1 Time-Stamp Counter Register (TSCTR)

The time-stamp counter register (TSCTR) is shown in [Figure 15-116](#) and described in [Table 15-109](#).

**Figure 15-116. Time-Stamp Counter Register (TSCTR)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-109. Time-Stamp Counter Register (TSCTR) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | TSCTR | 0-FFFF FFFFh | Active 32-bit counter register that is used as the capture time-base |

#### 15.3.4.2 Counter Phase Control Register (CTRPHS)

The counter phase control register (CTRPHS) is shown in [Figure 15-117](#) and described in [Table 15-110](#).

**Figure 15-117. Counter Phase Control Register (CTRPHS)**

|    |        |       |
|----|--------|-------|
| 31 | CTRPHS | 0     |
|    |        | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-110. Counter Phase Control Register (CTRPHS) Field Descriptions**

| Bit  | Field  | Value        | Description  |
|------|--------|--------------|--|
| 31-0 | CTRPHS | 0xFFFF FFFFh | Counter phase value register that can be programmed for phase lag/lead. This register shadows TSCTR and is loaded into TSCTR upon either a SYNC1 event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases. |

#### 15.3.4.3 Capture 1 Register (CAP1)

The capture 1 register (CAP1) is shown in [Figure 15-118](#) and described in [Table 15-111](#).

**Figure 15-118. Capture 1 Register (CAP1)**

|    |      |       |
|----|------|-------|
| 31 | CAP1 | 0     |
|    |      | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-111. Capture 1 Register (CAP1) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | CAP1  | 0xFFFF FFFFh | This register can be loaded (written) by: <ul style="list-style-type: none"><li>• Time-Stamp (i.e., counter value) during a capture event</li><li>• Software - may be useful for test purposes</li><li>• APRD active register when used in APWM mode</li></ul> |

#### 15.3.4.4 Capture 2 Register (CAP2)

The capture 2 register (CAP2) is shown in [Figure 15-119](#) and described in [Table 15-112](#).

**Figure 15-119. Capture 2 Register (CAP2)**

|    |      |       |
|----|------|-------|
| 31 | CAP2 | 0     |
|    |      | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-112. Capture 2 Register (CAP2) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | CAP2  | 0xFFFF FFFFh | This register can be loaded (written) by: <ul style="list-style-type: none"> <li>• Time-Stamp (i.e., counter value) during a capture event</li> <li>• Software - may be useful for test purposes</li> <li>• ACMP active register when used in APWM mode</li> </ul> |

#### 15.3.4.5 Capture 3 Register (CAP3)

The capture 3 register (CAP3) is shown in [Figure 15-120](#) and described in [Table 15-113](#).

**Figure 15-120. Capture 3 Register (CAP3)**

|    |      |       |
|----|------|-------|
| 31 | CAP3 | 0     |
|    |      | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-113. Capture 3 Register (CAP3) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | CAP3  | 0xFFFF FFFFh | In CMP mode, this is a time-stamp capture register. In APWM mode, this is the period shadow (APRD) register. You update the PWM period value through this register. In this mode, CAP3 shadows CAP1. |

#### 15.3.4.6 Capture 4 Register (CAP4)

The capture 4 register (CAP4) is shown in [Figure 15-121](#) and described in [Table 15-114](#).

**Figure 15-121. Capture 4 Register (CAP4)**

|    |      |       |
|----|------|-------|
| 31 | CAP4 | 0     |
|    |      | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-114. Capture 4 Register (CAP4) Field Descriptions**

| Bit  | Field | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | CAP4  | 0xFFFF FFFFh | In CMP mode, this is a time-stamp capture register. In APWM mode, this is the compare shadow (ACMP) register. You update the PWM compare value through this register. In this mode, CAP4 shadows CAP2. |

#### 15.3.4.7 ECAP Control Register 1 (ECCTL1)

The ECAP control register 1 (ECCTL1) is shown in [Figure 15-122](#) and described in [Table 15-115](#).

**Figure 15-122. ECAP Control Register 1 (ECCTL1)**

|           |         |         |          |         |         |         |         |
|-----------|---------|---------|----------|---------|---------|---------|---------|
| 15        | 14      | 13      |          |         |         | 9       | 8       |
| FREE/SOFT |         |         | PRESCALE |         |         |         | CAPLDEN |
|           |         | R/W-0   |          | R/W-0   |         |         | R/W-0   |
| 7         | 6       | 5       | 4        | 3       | 2       | 1       | 0       |
| CTRRST4   | CAP4POL | CTRRST3 | CAP3POL  | CTRRST2 | CAP2POL | CTRRST1 | CAP1POL |
| R/W-0     | R/W-0   | R/W-0   | R/W-0    | R/W-0   | R/W-0   | R/W-0   | R/W-0   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-115. ECAP Control Register 1 (ECCTL1) Field Descriptions**

| Bit   | Field     | Value  | Description  |
|-------|-----------|--|--|
| 15-14 | FREE/SOFT | 0-3h<br>0<br>1h<br>2h-3h                                     | Emulation Control<br>TSCTR counter stops immediately on emulation suspend<br>TSCTR counter runs until = 0<br>TSCTR counter is unaffected by emulation suspend (Run Free)                           |
| 13-9  | PRESCALE  | 0-1Fh<br>0<br>1<br>2h<br>3h<br>4h<br>5h<br>...<br>1Eh<br>1Fh | Event Filter prescale select<br>Divide by 1 (i.e., no prescale, by-pass the prescaler)<br>Divide by 2<br>Divide by 4<br>Divide by 6<br>Divide by 8<br>Divide by 10<br>Divide by 60<br>Divide by 62 |
| 8     | CAPLDEN   | 0<br>1   | Enable Loading of CAP1-4 registers on a capture event<br>Disable CAP1-4 register loads at capture event time.<br>Enable CAP1-4 register loads at capture event time.                               |

**Table 15-115. ECAP Control Register 1 (ECCTL1) Field Descriptions (continued)**

| Bit | Field   | Value | Description  |
|-----|---------|-------|--|
| 7   | CTRRST4 | 0     | Counter Reset on Capture Event 4<br><i>Do not</i> reset counter on Capture Event 4 (absolute time stamp operation) |
|     |         | 1     | Reset counter after Capture Event 4 time-stamp has been captured<br>(used in difference mode operation)            |
| 6   | CAP4POL | 0     | Capture Event 4 Polarity select<br>Capture Event 4 triggered on a rising edge (RE)                                 |
|     |         | 1     | Capture Event 4 triggered on a falling edge (FE)   |
| 5   | CTRRST3 | 0     | Counter Reset on Capture Event 3<br><i>Do not</i> reset counter on Capture Event 3 (absolute time stamp)           |
|     |         | 1     | Reset counter after Event 3 time-stamp has been captured<br>(used in difference mode operation)                    |
| 4   | CAP3POL | 0     | Capture Event 3 Polarity select<br>Capture Event 3 triggered on a rising edge (RE)                                 |
|     |         | 1     | Capture Event 3 triggered on a falling edge (FE)   |
| 3   | CTRRST2 | 0     | Counter Reset on Capture Event 2<br><i>Do not</i> reset counter on Capture Event 2 (absolute time stamp)           |
|     |         | 1     | Reset counter after Event 2 time-stamp has been captured<br>(used in difference mode operation)                    |
| 2   | CAP2POL | 0     | Capture Event 2 Polarity select<br>Capture Event 2 triggered on a rising edge (RE)                                 |
|     |         | 1     | Capture Event 2 triggered on a falling edge (FE)   |
| 1   | CTRRST1 | 0     | Counter Reset on Capture Event 1<br><i>Do not</i> reset counter on Capture Event 1 (absolute time stamp)           |
|     |         | 1     | Reset counter after Event 1 time-stamp has been captured (used in difference mode operation)                       |
| 0   | CAP1POL | 0     | Capture Event 1 Polarity select<br>Capture Event 1 triggered on a rising edge (RE)                                 |
|     |         | 1     | Capture Event 1 triggered on a falling edge (FE)   |

### 15.3.4.8 ECAP Control Register 2 (ECCTL2)

The ECAP control register 2 (ECCTL2) is shown in [Figure 15-123](#) and described in [Table 15-116](#).

**Figure 15-123. ECAP Control Register 2 (ECCTL2)**

| 15       |           |          |           |       | 11        | 10       | 9           | 8      |
|----------|-----------|----------|-----------|-------|-----------|----------|-------------|--------|
| Reserved |           |          |           |       |           | APWM POL | CAP/APWM    | SWSYNC |
|          | R-0       |          |           |       |           |          | R/W-0       | R/W-0  |
| 7        | 6         | 5        | 4         | 3     | 2         | 1        | 0           |        |
|          | SYNCO_SEL | SYNCI_EN | TSCTRSTOP | REARM | STOP_WRAP |          | CONT/ONESHT |        |
|          | R/W-0     | R/W-0    | R/W-0     | R/W-0 | R/W-1     |          | R/W-0       |        |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-116. ECAP Control Register 2 (ECCTL2) Field Descriptions**

| Bit   | Field     | Value  | Description   |
|-------|-----------|--------|---|
| 15-11 | Reserved  | 0      | Reserved  |
| 10    | APWM POL  | 0<br>1 | APWM output polarity select. This is applicable only in APWM operating mode<br>Output is active high (Compare value defines high time)<br>Output is active low (Compare value defines low time)   |
| 9     | CAP/APWM  | 0<br>1 | CAP/APWM operating mode select<br>ECAP module operates in capture mode. This mode forces the following configuration: <ul style="list-style-type: none"><li>Inhibits TSCTR resets via CTR = PRD event</li><li>Inhibits shadow loads on CAP1 and 2 registers</li><li>Permits user to enable CAP1-4 register load</li><li>ECAPn/APWMn pin operates as a capture input</li></ul> ECAP module operates in APWM mode. This mode forces the following configuration: <ul style="list-style-type: none"><li>Resets TSCTR on CTR = PRD event (period boundary)</li><li>Permits shadow loading on CAP1 and 2 registers</li><li>Disables loading of time-stamps into CAP1-4 registers</li><li>ECAPn/APWMn pin operates as a APWM output</li></ul> |
| 8     | SWSYNC    | 0<br>1 | Software-forced Counter (TSCTR) Synchronizing. This provides a convenient software method to synchronize some or all ECAP time bases. In APWM mode, the synchronizing can also be done via the CTR = PRD event.<br>Writing a zero has no effect. Reading always returns a zero<br>Writing a one forces a TSCTR shadow load of current ECAP module and any ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing a 1, this bit returns to a zero.<br>Note: Selection CTR = PRD is meaningful only in APWM mode; however, you can choose it in CAP mode if you find doing so useful.   |
| 7-6   | SYNCO_SEL | 0-3h   | Sync-Out Select<br>0 Select sync-in event to be the sync-out signal (pass through)<br>1h Select CTR = PRD event to be the sync-out signal<br>2h Disable sync out signal<br>3h Disable sync out signal   |
| 5     | SYNCI_EN  | 0<br>1 | Counter (TSCTR) Sync-In select mode<br>0 Disable sync-in option<br>1 Enable counter (TSCTR) to be loaded from CTRPHS register upon either a SYNCI signal or a S/W force event.  |
| 4     | TSCTRSTOP | 0<br>1 | Time Stamp (TSCTR) Counter Stop (freeze) Control<br>0 TSCTR stopped<br>1 TSCTR free-running   |

**Table 15-116. ECAP Control Register 2 (ECCTL2) Field Descriptions (continued)**

| Bit | Field       | Value | Description   |
|-----|-------------|-------|---|
| 3   | RE-ARM      |       | <p>One-Shot Re-Arming Control, that is, wait for stop trigger. Note: The re-arm function is valid in one shot or continuous mode.</p> <p>0 Has no effect (reading always returns a 0)</p> <p>1 Arms the one-shot sequence as follows:<br/>           1) Resets the Mod4 counter to zero<br/>           2) Unfreezes the Mod4 counter<br/>           3) Enables capture register loads</p>   |
| 2-1 | STOP_WRAP   | 0-3h  | <p>Stop value for one-shot mode. This is the number (between 1-4) of captures allowed to occur before the CAP(1-4) registers are frozen, that is, capture sequence is stopped.<br/>           Wrap value for continuous mode. This is the number (between 1-4) of the capture register in which the circular buffer wraps around and starts again.</p> <p>0 Stop after Capture Event 1 in one-shot mode.<br/>           Wrap after Capture Event 1 in continuous mode.</p> <p>1h Stop after Capture Event 2 in one-shot mode.<br/>           Wrap after Capture Event 2 in continuous mode.</p> <p>2h Stop after Capture Event 3 in one-shot mode.<br/>           Wrap after Capture Event 3 in continuous mode.</p> <p>3h Stop after Capture Event 4 in one-shot mode.<br/>           Wrap after Capture Event 4 in continuous mode.</p> <p>Notes: STOP_WRAP is compared to Mod4 counter and, when equal, 2 actions occur:<br/>           • Mod4 counter is stopped (frozen)<br/>           • Capture register loads are inhibited<br/>           In one-shot mode, further interrupt events are blocked until re-armed.</p> |
| 0   | CONT/ONESHT | 0     | <p>Continuous or one-shot mode control (applicable only in capture mode)</p> <p>0 Operate in continuous mode</p> <p>1 Operate in one-shot mode</p>  |

#### 15.3.4.9 ECAP Interrupt Enable Register (ECEINT)

The ECAP interrupt enable register (ECEINT) is shown in [Figure 15-124](#) and described in [Table 15-117](#).

The interrupt enable bits (CEVT $n$ ) block any of the selected events from generating an interrupt. Events will still be latched into the flag bit (ECFLG register) and can be forced/cleared via the ECFRC/ECCLR registers.

The proper procedure for configuring peripheral modes and interrupts is:

1. Disable global interrupts
2. Stop eCAP counter
3. Disable eCAP interrupts
4. Configure peripheral registers
5. Clear spurious eCAP interrupt flags
6. Enable eCAP interrupts
7. Start eCAP counter
8. Enable global interrupts

**Figure 15-124. ECAP Interrupt Enable Register (ECEINT)**

| 15      | Reserved |        |       |       |       |       |          |       | 8     |
|---------|----------|--------|-------|-------|-------|-------|----------|-------|-------|
| R-0     |          |        |       |       |       |       |          |       |       |
| 7       | 6        | 5      | 4     | 3     | 2     | 1     | 0        |       |       |
| CTR=CMP | CTR=PRD  | CTROVF | CEVT4 | CEVT3 | CEVT2 | CETV1 | Reserved | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-117. ECAP Interrupt Enable Register (ECEINT) Field Descriptions**

| Bit  | Field    | Value | Description                                     |
|------|----------|-------|---|
| 15-8 | Reserved | 0     | Reserved  |
| 7    | CTR=CMP  | 0     | Counter Equal Compare Interrupt Enable          |
|      |          | 1     | Disable Compare Equal as an Interrupt source    |
|      |          | 1     | Enable Compare Equal as an Interrupt source     |
| 6    | CTR=PRD  | 0     | Counter Equal Period Interrupt Enable           |
|      |          | 1     | Disable Period Equal as an Interrupt source     |
|      |          | 1     | Enable Period Equal as an Interrupt source      |
| 5    | CTROVF   | 0     | Counter Overflow Interrupt Enable               |
|      |          | 1     | Disable counter Overflow as an Interrupt source |
|      |          | 1     | Enable counter Overflow as an Interrupt source  |
| 4    | CEVT4    | 0     | Capture Event 4 Interrupt Enable                |
|      |          | 1     | Disable Capture Event 4 as an Interrupt source  |
|      |          | 1     | Enable Capture Event 4 as an Interrupt source   |
| 3    | CEVT3    | 0     | Capture Event 3 Interrupt Enable                |
|      |          | 1     | Disable Capture Event 3 as an Interrupt source  |
|      |          | 1     | Enable Capture Event 3 as an Interrupt source   |
| 2    | CEVT2    | 0     | Capture Event 2 Interrupt Enable                |
|      |          | 1     | Disable Capture Event 2 as an Interrupt source  |
|      |          | 1     | Enable Capture Event 2 as an Interrupt source   |
| 1    | CEVT1    | 0     | Capture Event 1 Interrupt Enable                |
|      |          | 1     | Disable Capture Event 1 as an Interrupt source  |
|      |          | 1     | Enable Capture Event 1 as an Interrupt source   |
| 0    | Reserved | 0     | Reserved  |

### 15.3.4.10 ECAP Interrupt Flag Register (ECFLG)

The ECAP interrupt flag register (ECFLG) is shown in [Figure 15-125](#) and described in [Table 15-118](#).

**Figure 15-125. ECAP Interrupt Flag Register (ECFLG)**

|         |          |        |       |       |       |       |     |  |   |
|---------|----------|--------|-------|-------|-------|-------|-----|--|---|
| 15      | Reserved |        |       |       |       |       |     |  | 8 |
|         | R-0      |        |       |       |       |       |     |  |   |
| 7       | 6        | 5      | 4     | 3     | 2     | 1     | 0   |  |   |
| CTR=CMP | CTR=PRD  | CTROVF | CEVT4 | CETV3 | CEVT2 | CETV1 | INT |  |   |
| R-0     | R-0      | R-0    | R-0   | R-0   | R-0   | R-0   | R-0 |  |   |

LEGEND: R = Read only; -n = value after reset

**Table 15-118. ECAP Interrupt Flag Register (ECFLG) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved   |
| 7    | CTR=CMP  | 0     | Compare Equal Compare Status Flag. This flag is only active in APWM mode.<br>Indicates no event occurred |
|      |          | 1     | Indicates the counter (TSCTR) reached the compare register value (ACMP)                                  |
| 6    | CTR=PRD  | 0     | Counter Equal Period Status Flag. This flag is only active in APWM mode.<br>Indicates no event occurred  |
|      |          | 1     | Indicates the counter (TSCTR) reached the period register value (APRD) and was reset.                    |
| 5    | CTROVF   | 0     | Counter Overflow Status Flag. This flag is active in CAP and APWM mode.<br>Indicates no event occurred.  |
|      |          | 1     | Indicates the counter (TSCTR) has made the transition from 0xFFFFFFFF to 0x00000000                      |
| 4    | CEVT4    | 0     | Capture Event 4 Status Flag This flag is only active in CAP mode.<br>Indicates no event occurred         |
|      |          | 1     | Indicates the fourth event occurred at ECAPn pin   |
| 3    | CEVT3    | 0     | Capture Event 3 Status Flag. This flag is active only in CAP mode.<br>Indicates no event occurred.       |
|      |          | 1     | Indicates the third event occurred at ECAPn pin.   |
| 2    | CEVT2    | 0     | Capture Event 2 Status Flag. This flag is only active in CAP mode.<br>Indicates no event occurred.       |
|      |          | 1     | Indicates the second event occurred at ECAPn pin.  |
| 1    | CEVT1    | 0     | Capture Event 1 Status Flag. This flag is only active in CAP mode.<br>Indicates no event occurred.       |
|      |          | 1     | Indicates the first event occurred at ECAPn pin.   |
| 0    | INT      | 0     | Global Interrupt Status Flag<br>Indicates no interrupt generated.  |
|      |          | 1     | Indicates that an interrupt was generated.   |

### 15.3.4.11 ECAP Interrupt Clear Register (ECCLR)

The ECAP interrupt clear register (ECCLR) is shown in [Figure 15-126](#) and described in [Table 15-119](#).

**Figure 15-126. ECAP Interrupt Clear Register (ECCLR)**

|         |          |        |       |       |       |       |       |  |   |
|---------|----------|--------|-------|-------|-------|-------|-------|--|---|
| 15      | Reserved |        |       |       |       |       |       |  | 8 |
|         | R-0      |        |       |       |       |       |       |  |   |
| 7       | 6        | 5      | 4     | 3     | 2     | 1     | 0     |  |   |
| CTR=CMP | CTR=PRD  | CTROVF | CEVT4 | CETV3 | CETV2 | CETV1 | INT   |  |   |
| R/W-0   | R/W-0    | R/W-0  | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-119. ECAP Interrupt Clear Register (ECCLR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 15-8 | Reserved | 0     | Reserved  |
| 7    | CTR=CMP  | 0     | Counter Equal Compare Status Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0  |
|      |          | 1     | Writing a 1 clears the CTR=CMP flag condition   |
| 6    | CTR=PRD  | 0     | Counter Equal Period Status Flag  |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0  |
|      |          | 1     | Writing a 1 clears the CTR=PRD flag condition   |
| 5    | CTROVF   | 0     | Counter Overflow Status Flag  |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0  |
|      |          | 1     | Writing a 1 clears the CTROVF flag condition  |
| 4    | CEVT4    | 0     | Capture Event 4 Status Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0.   |
|      |          | 1     | Writing a 1 clears the CEVT3 flag condition.  |
| 3    | CEVT3    | 0     | Capture Event 3 Status Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0.   |
|      |          | 1     | Writing a 1 clears the CEVT3 flag condition.  |
| 2    | CEVT2    | 0     | Capture Event 2 Status Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0.   |
|      |          | 1     | Writing a 1 clears the CEVT2 flag condition.  |
| 1    | CEVT1    | 0     | Capture Event 1 Status Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0.   |
|      |          | 1     | Writing a 1 clears the CEVT1 flag condition.  |
| 0    | INT      | 0     | Global Interrupt Clear Flag   |
|      |          | 1     | Writing a 0 has no effect. Always reads back a 0.   |
|      |          | 1     | Writing a 1 clears the INT flag and enable further interrupts to be generated if any of the event flags are set to 1. |

### 15.3.4.12 ECAP Interrupt Forcing Register (ECFRC)

The ECAP interrupt forcing register (ECFRC) is shown in [Figure 15-127](#) and described in [Table 15-120](#).

**Figure 15-127. ECAP Interrupt Forcing Register (ECFRC)**

| 15       | 14      | 13     | 12    | 11    | 10    | 9     | 8        |
|----------|---------|--------|-------|-------|-------|-------|----------|
| Reserved |         |        |       |       |       |       |          |
| R-0      |         |        |       |       |       |       |          |
| 7        | 6       | 5      | 4     | 3     | 2     | 1     | 0        |
| CTR=CMP  | CTR=PRD | CTROVF | CEVT4 | CETV3 | CETV2 | CETV1 | Reserved |
| R/W-0    | R/W-0   | R/W-0  | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0      |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-120. ECAP Interrupt Forcing Register (ECFRC) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved   |
| 7    | CTR=CMP  | 0     | Force Counter Equal Compare Interrupt  |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CTR=CMP flag bit.            |
| 6    | CTR=PRD  | 0     | Force Counter Equal Period Interrupt   |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CTR=PRD flag bit.            |
| 5    | CTROVF   | 0     | Force Counter Overflow   |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 to this bit sets the CTROVF flag bit. |
| 4    | CEVT4    | 0     | Force Capture Event 4  |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CEVT4 flag bit               |
| 3    | CEVT3    | 0     | Force Capture Event 3  |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CEVT3 flag bit               |
| 2    | CEVT2    | 0     | Force Capture Event 2  |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CEVT2 flag bit.              |
| 1    | CEVT1    | 0     | Force Capture Event 1  |
|      |          | 1     | No effect. Always reads back a 0.<br>Writing a 1 sets the CEVT1 flag bit.              |
| 0    | Reserved | 0     | Reserved   |

#### 15.3.4.13 Revision ID Register (REVID)

The revision ID register (REVID) is shown in [Figure 15-128](#) and described in [Table 15-121](#).

**Figure 15-128. Revision ID Register (REVID)**



LEGEND: R = Read only; -n = value after reset

**Table 15-121. Revision ID Register (REVID) Field Descriptions**

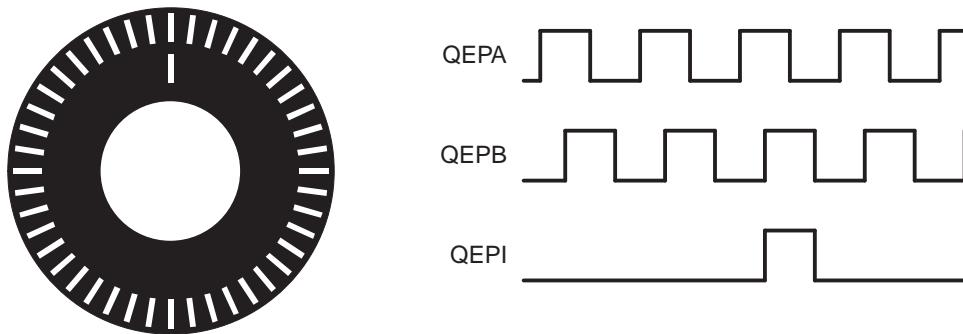
| Bit  | Field | Value      | Description  |
|------|-------|------------|--------------|
| 31-0 | REV   | 44D2 2100h | Revision ID. |

## 15.4 Enhanced Quadrature Encoder Pulse (eQEP) Module

### 15.4.1 Introduction

A single track of slots patterns the periphery of an incremental encoder disk, as shown in Figure 15-129. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference.

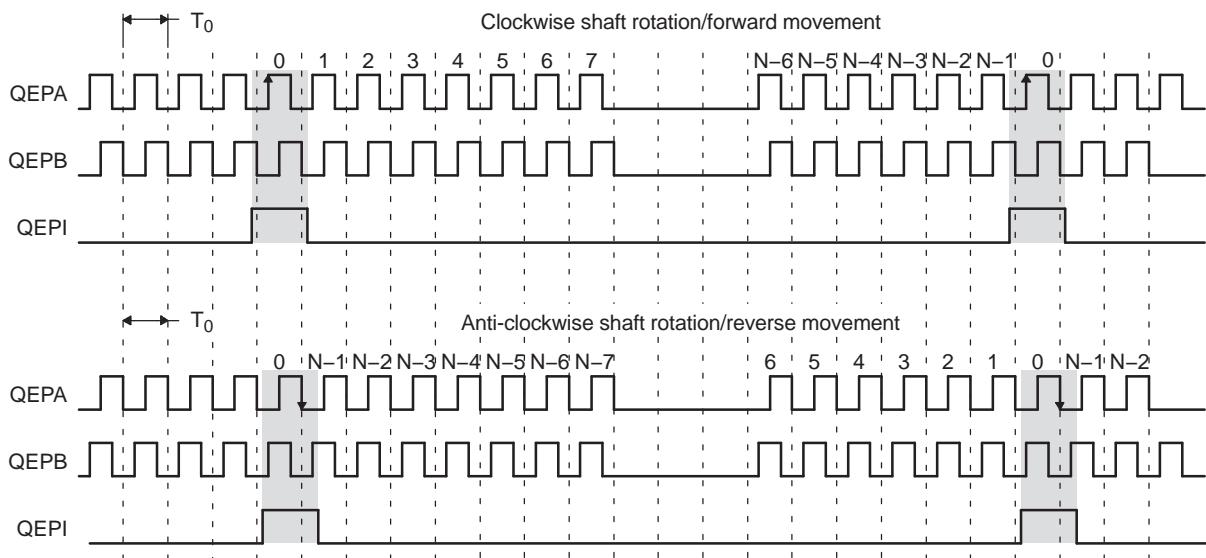
**Figure 15-129. Optical Encoder Disk**



To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90 degrees out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vice versa as shown in Figure 15-130.

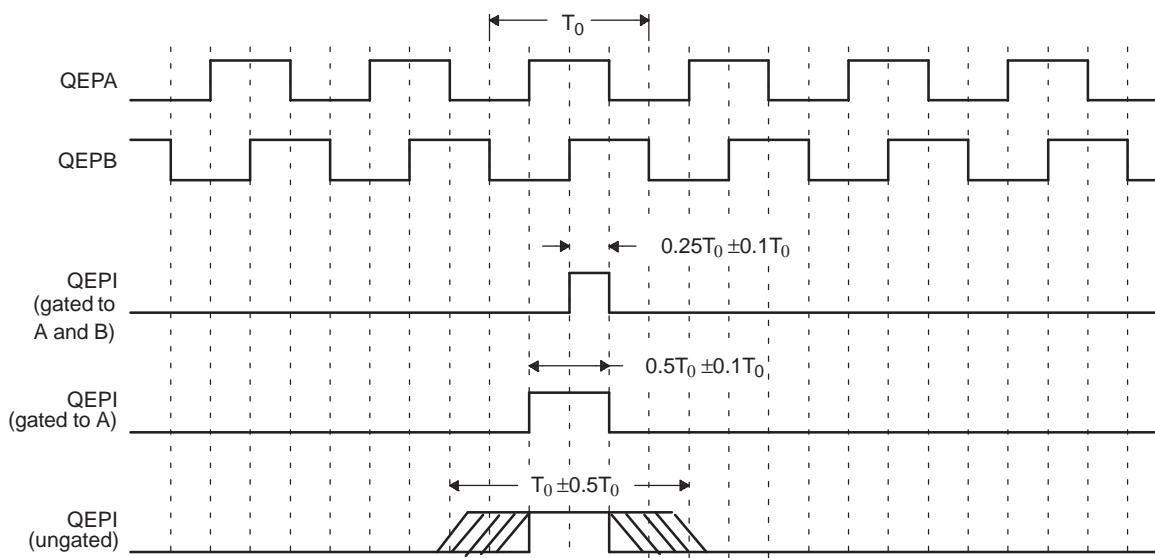
The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

**Figure 15-130. QEP Encoder Output Signal for Forward/Reverse Movement**



Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in [Figure 15-131](#). A nonstandard form of index pulse is ungated. In the ungated configuration, the index edges are not necessarily coincident with A and B signals. The gated index pulse is aligned to any of the four quadrature edges and width of the index pulse and can be equal to a quarter, half, or full period of the quadrature signal.

**Figure 15-131. Index Pulse Example**



Some typical applications of shaft encoders include robotics and even computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

**General Issues:** Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (1)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (2)$$

where

$v(k)$ : Velocity at time instant k

$x(k)$ : Position at time instant k

$x(k-1)$ : Position at time instant k - 1

T: Fixed unit time or inverse of velocity calculation rate

$\Delta X$ : Incremental position movement in unit time

$t(k)$ : Time instant "k"

$t(k-1)$ : Time instant "k - 1"

X: Fixed unit position

$\Delta T$ : Incremental time elapsed for unit position movement.

[Equation 1](#) is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity  $[x(k) - x(k-1)]$  is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant  $1/T$  (where T is the constant time between unit time events and is known in advance).

Estimation based on [Equation 1](#) has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, for example, 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, [Equation 2](#) provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. [Equation 2](#) can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does [Equation 1](#). A combination of relatively large motor speeds and high sensor resolution makes the time interval  $\Delta T$  small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use [Equation 2](#) at low speed and have the software switch over to [Equation 1](#) when the motor speed rises above some specified threshold.

### 15.4.2 Functional Description

This section provides the eQEP inputs and functional description.

---

**NOTE:** Multiple identical eQEP modules can be contained in a system. The number of modules is device-dependent and is based on target application needs. In this document, the letter x within a signal or module name is used to indicate a generic eQEP instance on a device.

---

#### 15.4.2.1 EQEP Inputs

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input.

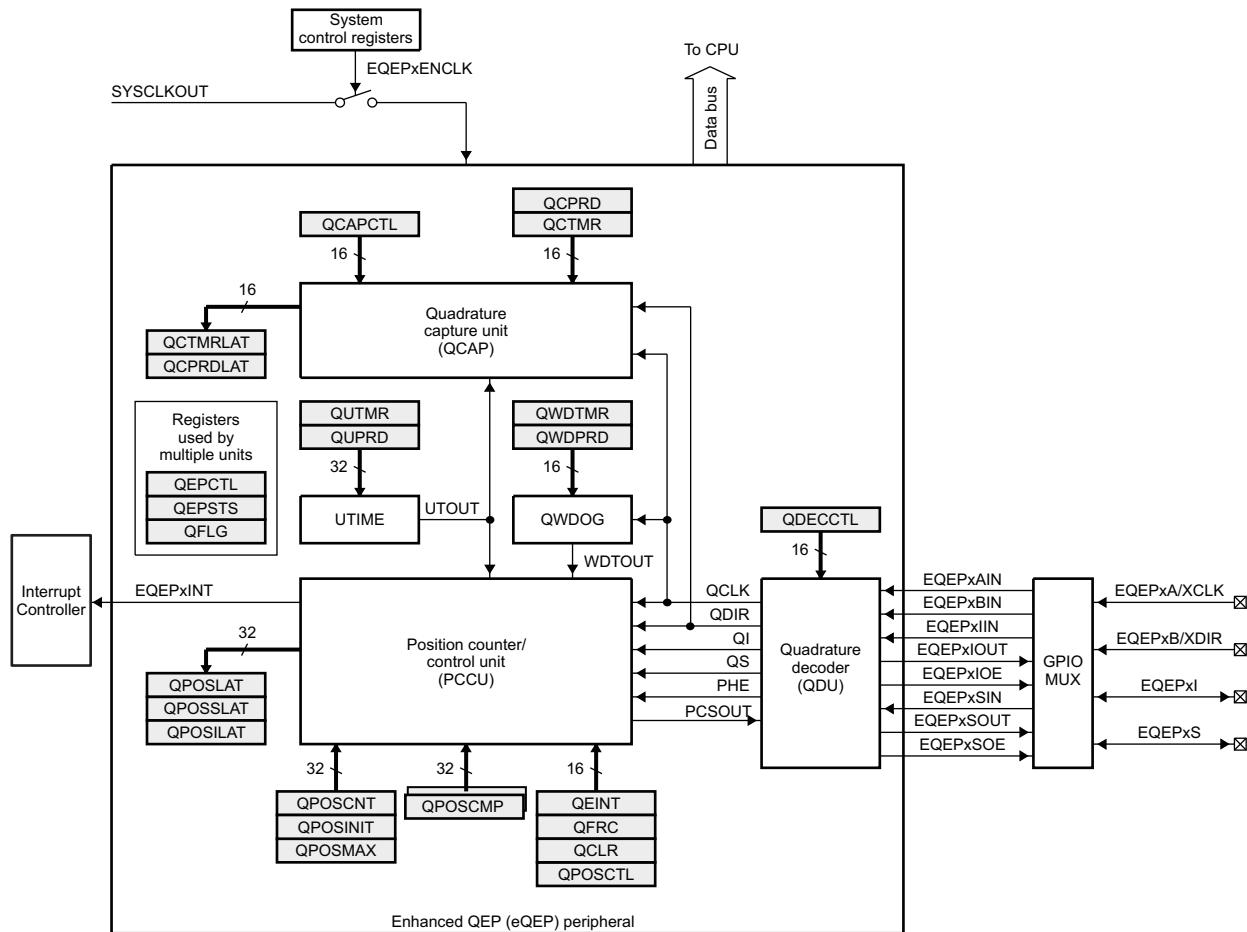
- QEPA/XCLK and QEPB/XDIR: These two pins can be used in quadrature-clock mode or direction-count mode.
  - Quadrature-clock Mode: The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.
  - Direction-count Mode: In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.
- QEPI: Index or Zero Marker: The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.
- QEPS: Strobe Input: This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

#### 15.4.2.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in [Figure 15-132](#)):

- Programmable input qualification for each pin (part of the GPIO MUX)
- Quadrature decoder unit (QDU)
- Position counter and control unit for position measurement (PCCU)
- Quadrature edge-capture unit for low-speed measurement (QCAP)
- Unit time base for speed/frequency measurement (UTIME)
- Watchdog timer for detecting stalls (QWDOG)

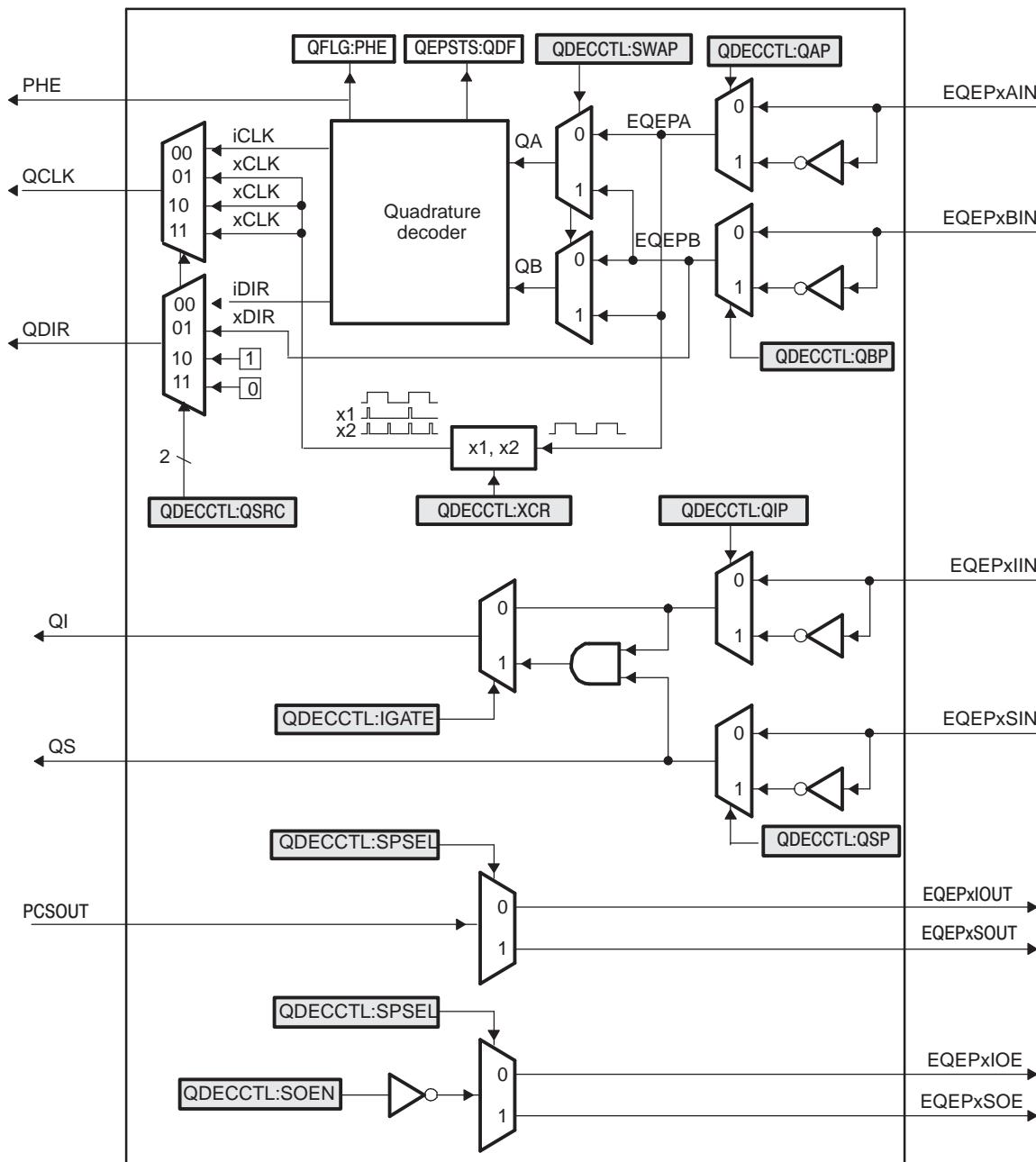
**Figure 15-132. Functional Block Diagram of the eQEP Peripheral**



### 15.4.2.3 Quadrature Decoder Unit (QDU)

Figure 15-133 shows a functional block diagram of the QDU.

**Figure 15-133. Functional Block Diagram of Decoder Unit**



### 15.4.2.3.1 Position Counter Input Modes

Clock and direction input to position counter is selected using the QSRC bit in the eQEP decoder control register (QDECCTL), based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode

#### 15.4.2.3.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

**Direction Decoding**— The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in the QDF bit in the eQEP status register (QEPSTS). [Table 15-122](#) and [Figure 15-134](#) show the direction decoding logic in truth table and state machine form. Both edges of the QEPA and QEPB signals are sensed to generate count pulses for the position counter. Therefore, the frequency of the clock generated by the eQEP logic is four times that of each input sequence. [Figure 15-135](#) shows the direction decoding and clock generation from the eQEP input signals.

**Phase Error Flag**— In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in [Figure 15-134](#) are invalid transitions that generate a phase error.

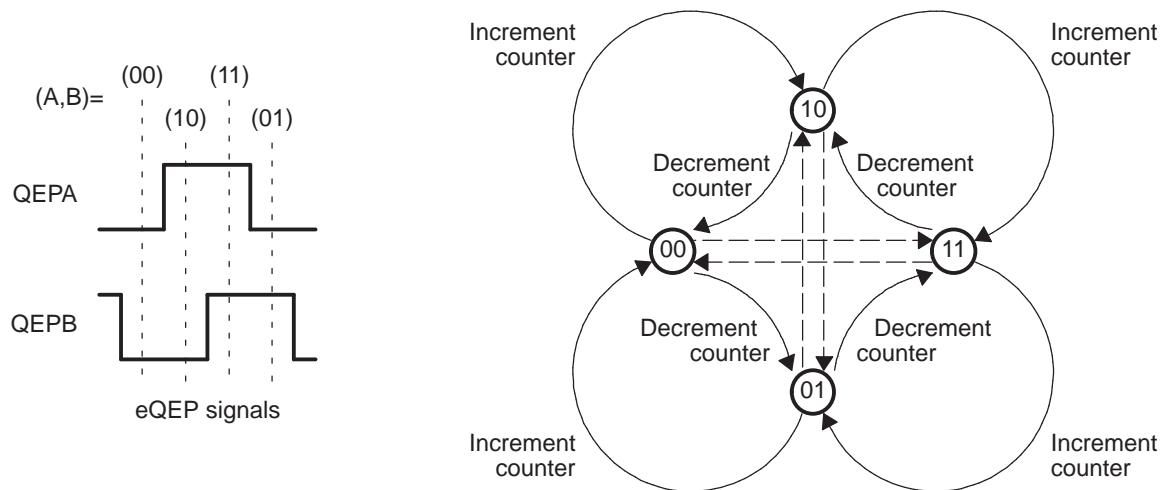
**Count Multiplication**— The eQEP position counter provides 4x times the resolution of an input clock by generating a quadrature-clock (QCLK) on the rising/falling edges of both eQEP input clocks (QEPA and QEPB) as shown in [Figure 15-135](#).

**Reverse Count**— In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the eQEP decoder control register (QDECCTL). This will swap the input to the quadrature decoder thereby reversing the counting direction.

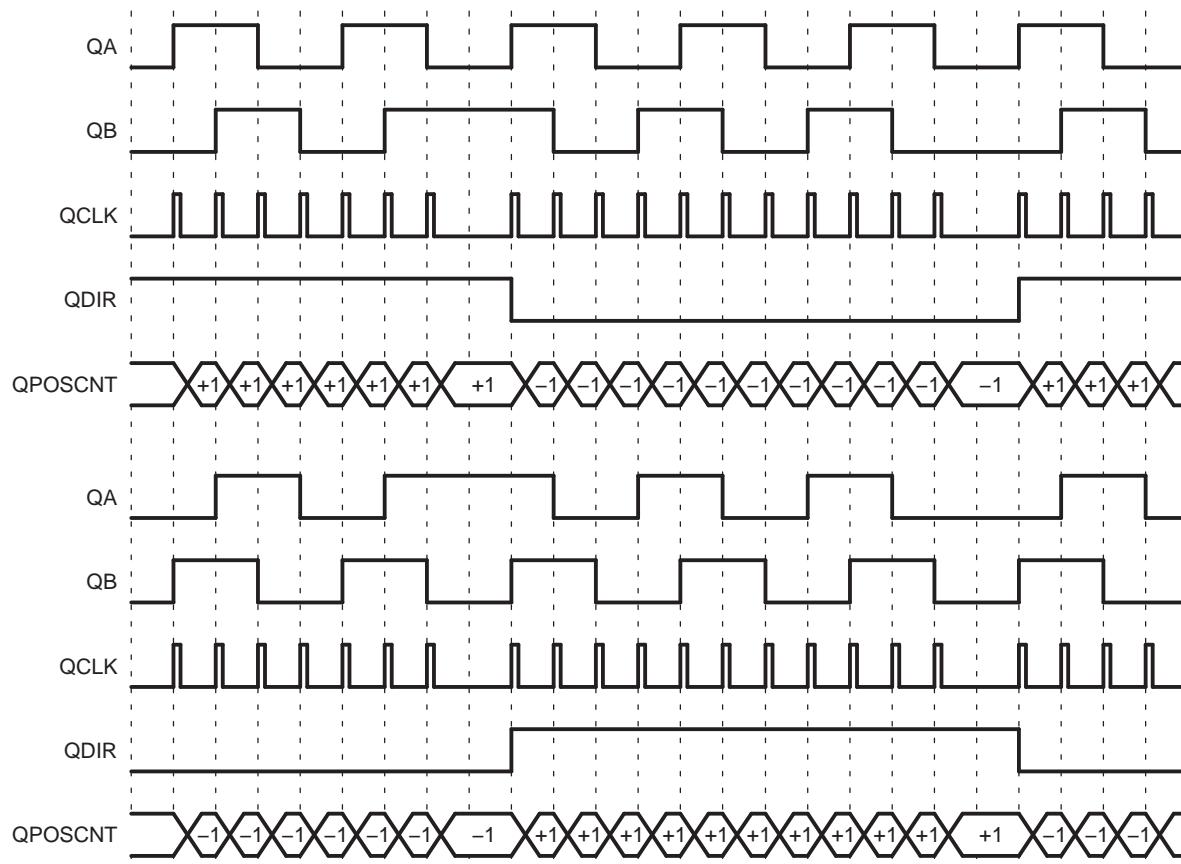
**Table 15-122. Quadrature Decoder Truth Table**

| Previous Edge | Present Edge | QDIR   | QPOSCNT                |
|---------------|--------------|--------|------------------------|
| QA↑           | QB↑          | UP     | Increment              |
|               | QB↓          | DOWN   | Decrement              |
|               | QA↓          | TOGGLE | Increment or Decrement |
| QA↓           | QB↓          | UP     | Increment              |
|               | QB↑          | DOWN   | Decrement              |
|               | QA↑          | TOGGLE | Increment or Decrement |
| QB↑           | QA↑          | DOWN   | Increment              |
|               | QA↓          | UP     | Decrement              |
|               | QB↓          | TOGGLE | Increment or Decrement |
| QB↓           | QA↓          | DOWN   | Increment              |
|               | QA↑          | UP     | Decrement              |
|               | QB↑          | TOGGLE | Increment or Decrement |

**Figure 15-134. Quadrature Decoder State Machine**



**Figure 15-135. Quadrature-clock and Direction Decoding**



#### **15.4.2.3.1.2 Direction-count Mode**

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high and decremented when the direction input is low.

#### **15.4.2.3.1.3 Up-Count Mode**

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the QEPA input. Setting of the XCR bit in the eQEP decoder control register (QDECCTL) enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2x factor.

#### **15.4.2.3.1.4 Down-Count Mode**

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the QEPA input. Setting of the XCR bit in the eQEP decoder control register (QDECCTL) enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2x factor.

#### **15.4.2.3.2 eQEP Input Polarity Selection**

Each eQEP input can be inverted using the in the eQEP decoder control register (QDECCTL[8:5]) control bits. As an example, setting of the QIP bit in QDECCTL inverts the index input.

#### **15.4.2.3.3 Position-Compare Sync Output**

The eQEP peripheral includes a position-compare unit that is used to generate the position-compare sync signal on compare match between the position counter register (QPOS\_CNT) and the position-compare register (QPOS\_CMP). This sync signal can be output using an index pin or strobe pin of the EQEP peripheral.

Setting the SOEN bit in the eQEP decoder control register (QDECCTL) enables the position-compare sync output and the SPSEL bit in QDECCTL selects either an eQEP index pin or an eQEP strobe pin.

#### **15.4.2.4 Position Counter and Control Unit (PCCU)**

The position counter and control unit provides two configuration registers (QEPCTL and QPOSCTL) for setting up position counter operational modes, position counter initialization/latch modes and position-compare logic for sync signal generation.

##### **15.4.2.4.1 Position Counter Operating Modes**

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in QFLG register.

#### 15.4.2.4.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM] = 00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

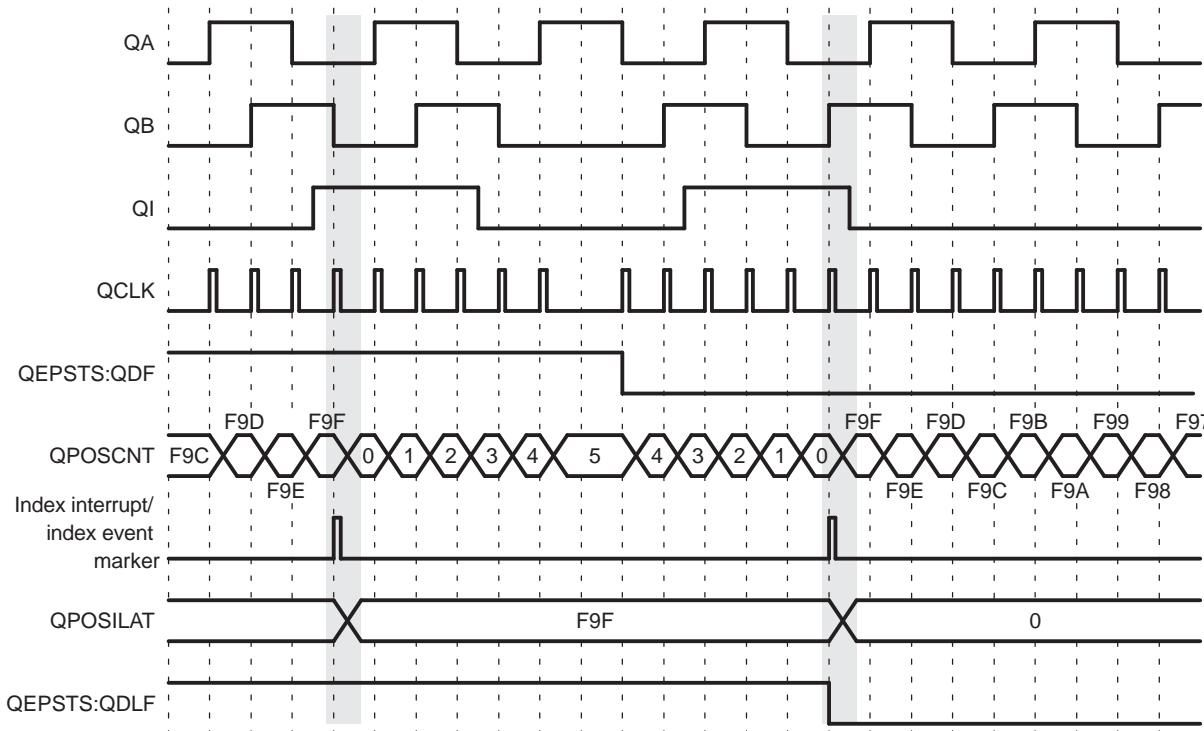
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in [Figure 15-136](#).

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] bits are ignored in this mode and position counter error flag/interrupt flag are generated only in index event reset mode.

**Figure 15-136. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or F9Fh)**

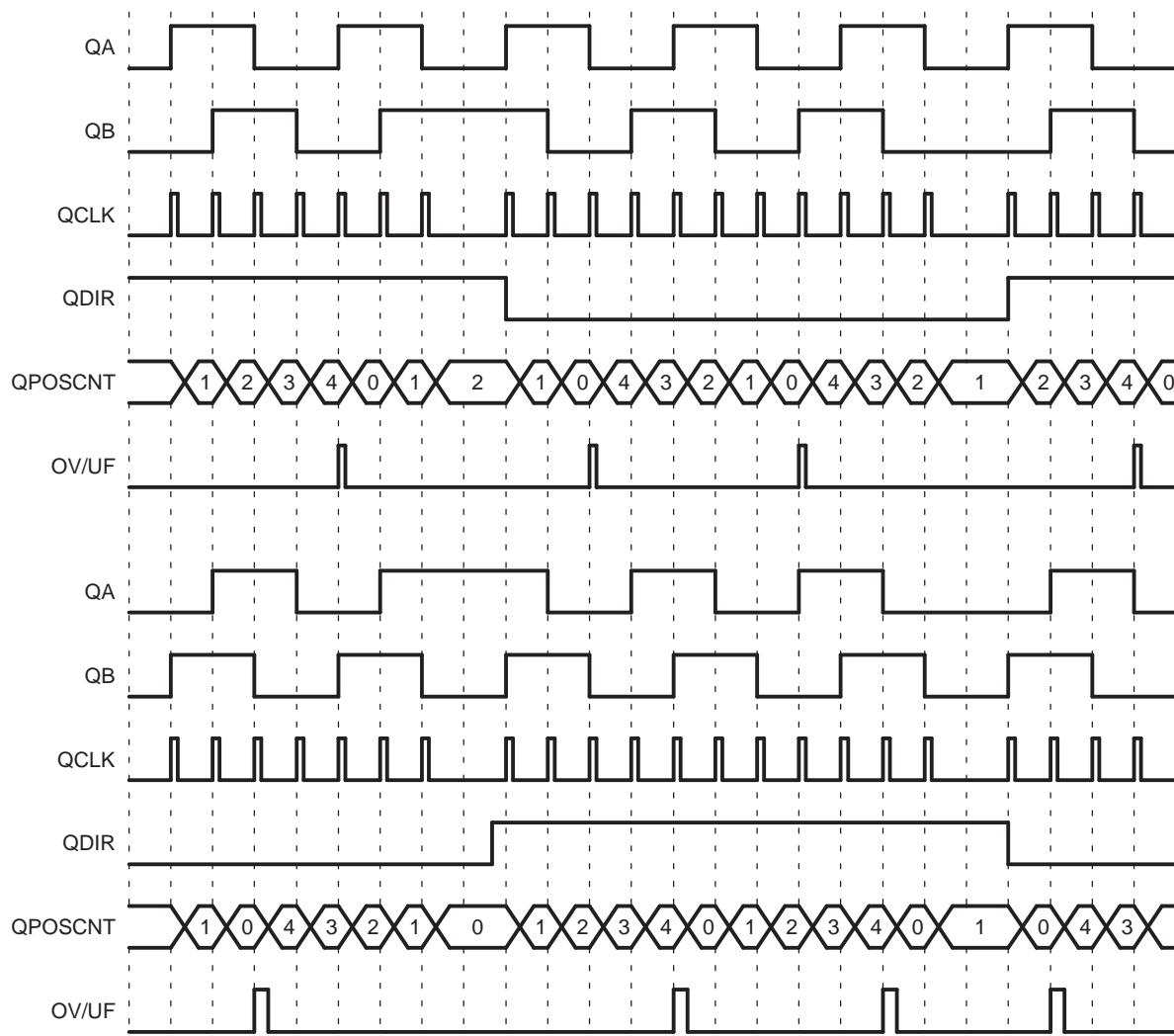


#### 15.4.2.4.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM]=01)

If the position counter is equal to QPOSMAX, then the position counter is reset to 0 on the next eQEP clock for forward movement and position counter overflow flag is set. If the position counter is equal to ZERO, then the position counter is reset to QPOSMAX on the next QEP clock for reverse movement and position counter underflow flag is set. Figure 15-137 shows the position counter reset operation in this mode.

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers; it also remembers the quadrature edge on the first index marker so that the same relative quadrature transition is used for the software index marker (QEPCTL[IEL]=11).

**Figure 15-137. Position Counter Underflow/Overflow (QPOSMAX = 4)**



#### **15.4.2.4.1.3 Position Counter Reset on the First Index Event ( $QEPCTL[PCRM] = 10$ )**

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in [Section 15.4.2.4.1.2](#).

First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for software index marker ( $QEPCTL[IEL]=11$ ).

#### **15.4.2.4.1.4 Position Counter Reset on Unit Time out Event ( $QEPCTL[PCRM] = 11$ )**

In this mode, the QPOSCNT value is latched to the QPOSLAT register and then the QPOSCNT is reset (to 0 or QPOSMAX, depending on the direction mode selected by QDECCTL[QSRC] bits on a unit time event). This is useful for frequency measurement.

#### **15.4.2.4.2 Position Counter Latch**

The eQEP index and strobe input can be configured to latch the position counter (QPOSCNT) into QPOSILAT and QPOSSLAT, respectively, on occurrence of a definite event on these pins.

##### **15.4.2.4.2.1 Index Event Latch**

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode ( $QEPCTL[PCRM] = 01$  and  $QEPCTL[PCRM] = 10$  modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge ( $QEPCTL[IEL] = 01$ )
- Latch on Falling edge ( $QEPCTL[IEL] = 10$ )
- Latch on Index Event Marker ( $QEPCTL[IEL] = 11$ )

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

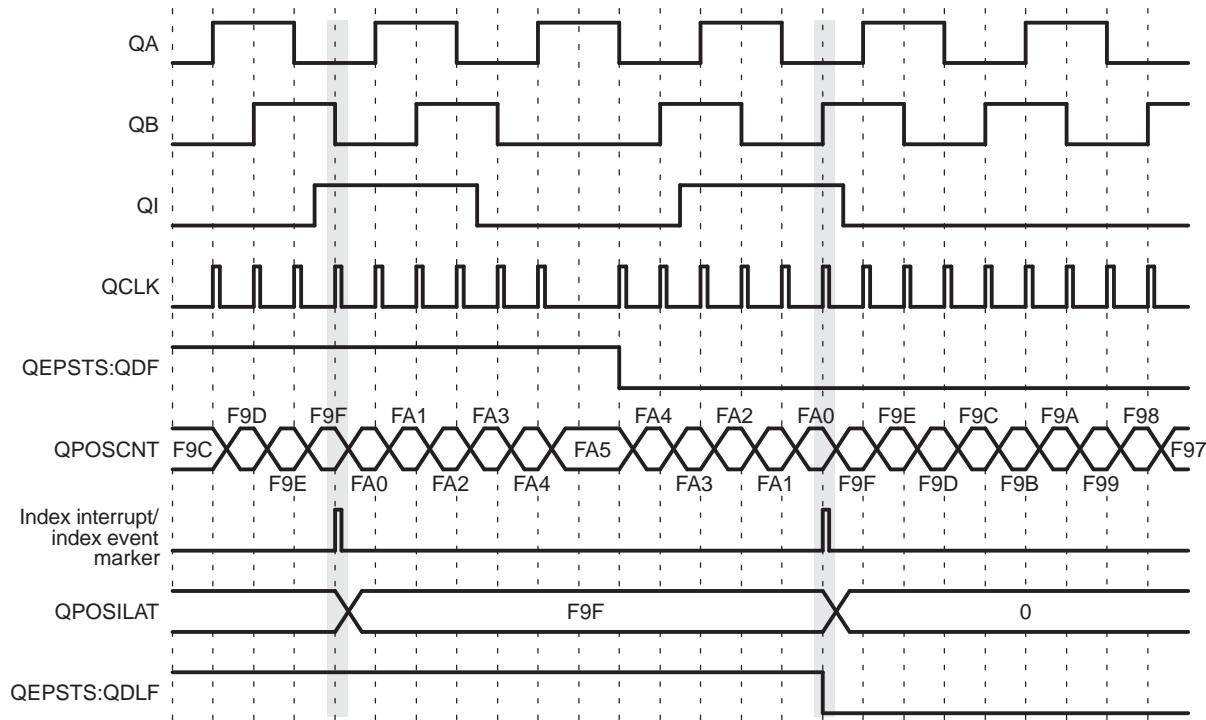
The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register. The index event latch configuration bits ( $QEPCTZ[IEL]$ ) are ignored when  $QEPCTL[PCRM] = 00$ .

**Latch on Rising Edge ( $QEPCTL[IEL] = 01$ )**— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

**Latch on Falling Edge ( $QEPCTL[IEL] = 10$ )**— The position counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

**Latch on Index Event Marker/Software Index Marker ( $QEPCTL[IEL] = 11$ )**— The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter ( $QEPCTL[IEL] = 11$ ).

[Figure 15-138](#) shows the position counter latch using an index event marker.

**Figure 15-138. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)**


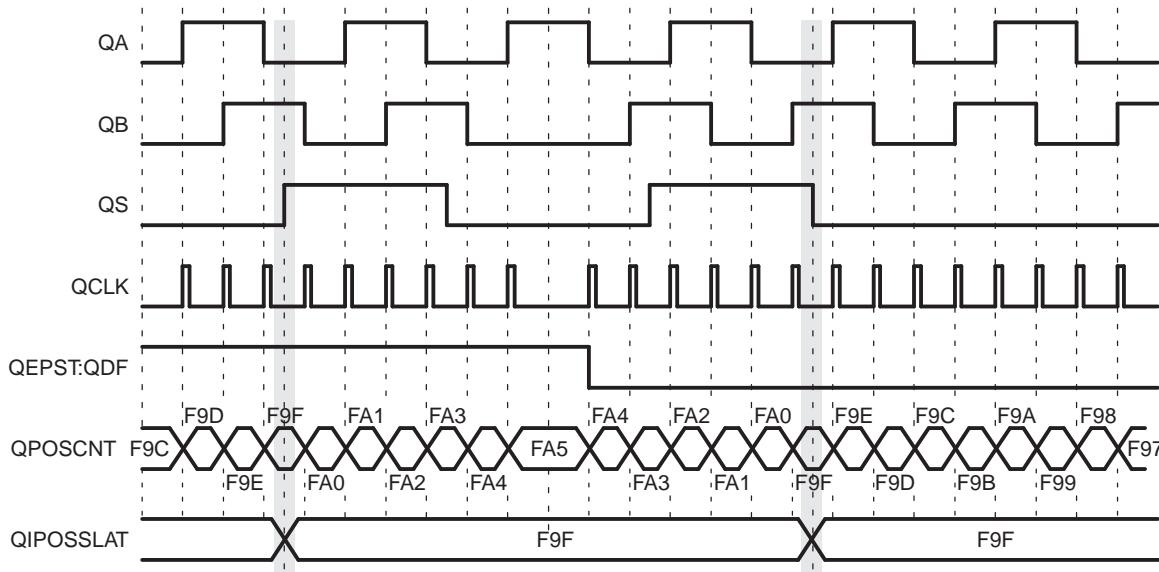
#### 15.4.2.4.2.2 Strobe Event Latch

The position-counter value is latched to the QPOSSLAT register on the rising edge of the strobe input by clearing the QEPCTL[SEL] bit.

If the QEPCTL[SEL] bit is set, then the position counter value is latched to the QPOSSLAT register on the rising edge of the strobe input for forward direction and on the falling edge of the strobe input for reverse direction as shown in [Figure 15-139](#).

The strobe event latch interrupt flag (QFLG[SEL]) is set when the position counter is latched to the QPOSSLAT register.

**Figure 15-139. Strobe Event Latch (QEPCTL[SEL] = 1)**



#### 15.4.2.4.3 Position Counter Initialization

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

**Index Event Initialization (IEI)**— The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input.

If the QEPCTL[IEI] bits are 10, then the position counter (QPOS\_CNT) is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

The index event initialization interrupt flag (QFLG[IEI]) is set when the position counter is initialized with a value in the QPOSINIT register.

**Strobe Event Initialization (SEI)**— If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

If the QEPCTL[SEL] bits are 11, then the position counter (QPOS\_CNT) is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

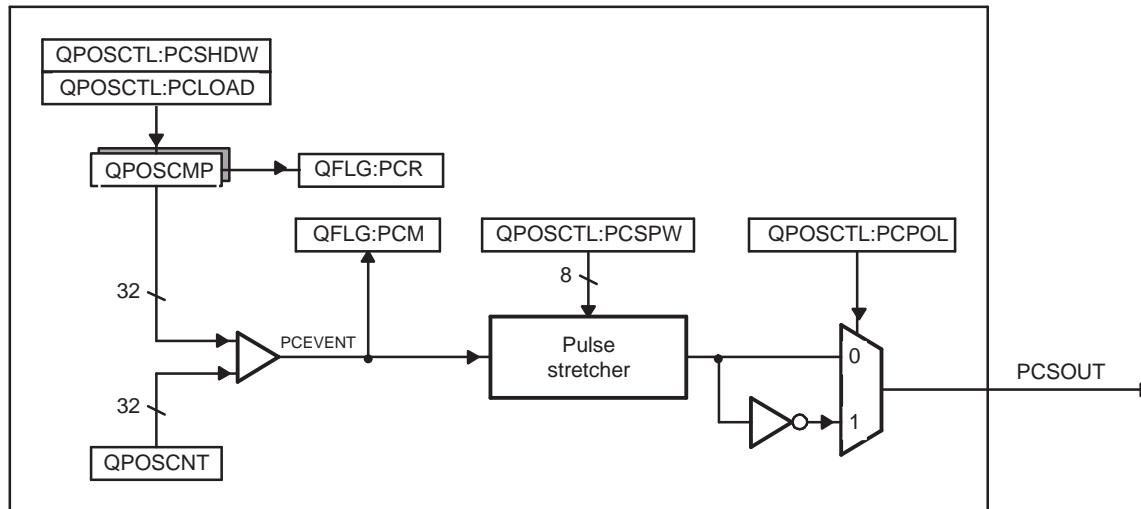
The strobe event initialization interrupt flag (QFLG[SEI]) is set when the position counter is initialized with a value in the QPOSINIT register.

**Software Initialization (SWI)**— The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit, which will automatically be cleared after initialization.

#### 15.4.2.4.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. [Figure 15-140](#) shows a diagram. The position-compare (QPOS\_CMP) register is shadowed and shadow mode can be enabled or disabled using the QPOSCTL[PSSHDW] bit. If the shadow mode is not enabled, the CPU writes directly to the active position compare register.

**Figure 15-140. eQEP Position-compare Unit**



In shadow mode, you can configure the position-compare unit (QPOSCTL[PCLOAD]) to load the shadow register value into the active register on the following events and to generate the position-compare ready (QFLG[PCR]) interrupt after loading.

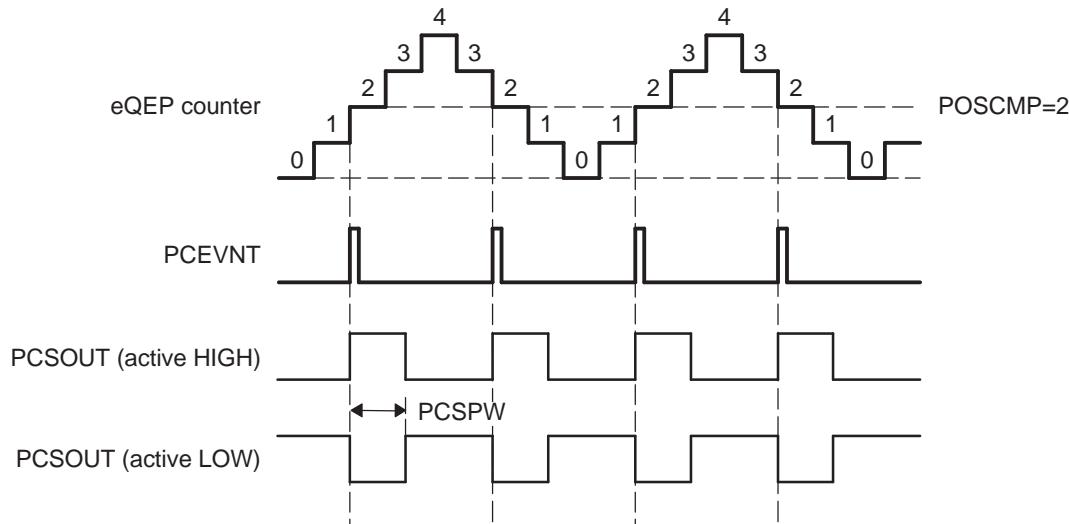
- Load on compare match
- Load on position-counter zero event

The position-compare match (QFLG[PCM]) is set when the position-counter value (QPOSCNT) matches with the active position-compare register (QPOS CMP) and the position-compare sync output of the programmable pulse width is generated on compare match to trigger an external device.

For example, if QPOS CMP = 2, the position-compare unit generates a position-compare event on 1 to 2 transitions of the eQEP position counter for forward counting direction and on 3 to 2 transitions of the eQEP position counter for reverse counting direction (see [Figure 15-141](#)).

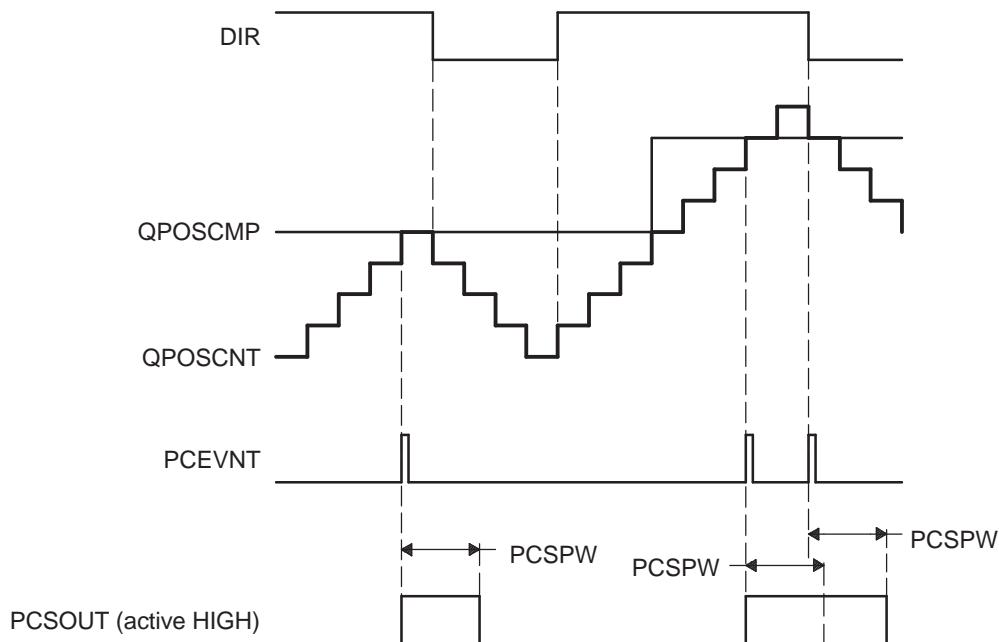
[Figure 15-163](#) shows the layout of the eQEP Position-Compare Control Register (QPOSCTL) and [Table 15-138](#) describes the QPOSCTL bit fields.

**Figure 15-141. eQEP Position-compare Event Generation Points**



The pulse stretcher logic in the position-compare unit generates a programmable position-compare sync pulse output on the position-compare match. In the event of a new position-compare match while a previous position-compare pulse is still active, then the pulse stretcher generates a pulse of specified duration from the new position-compare event as shown in [Figure 15-142](#).

**Figure 15-142. eQEP Position-compare Sync Output Pulse Stretcher**



### 15.4.2.5 eQEP Edge Capture Unit

The eQEP peripheral includes an integrated edge capture unit to measure the elapsed time between the unit position events as shown in [Figure 15-143](#). This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (3)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see [Figure 15-144](#))
- $\Delta T$  - Elapsed time between unit position events
- $v(k)$  - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS[UPEVNT] to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement and clear the flag by writing 1.

Time measurement ( $\Delta T$ ) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

- CPU read of QPOSCNT register
- Unit time-out event

If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

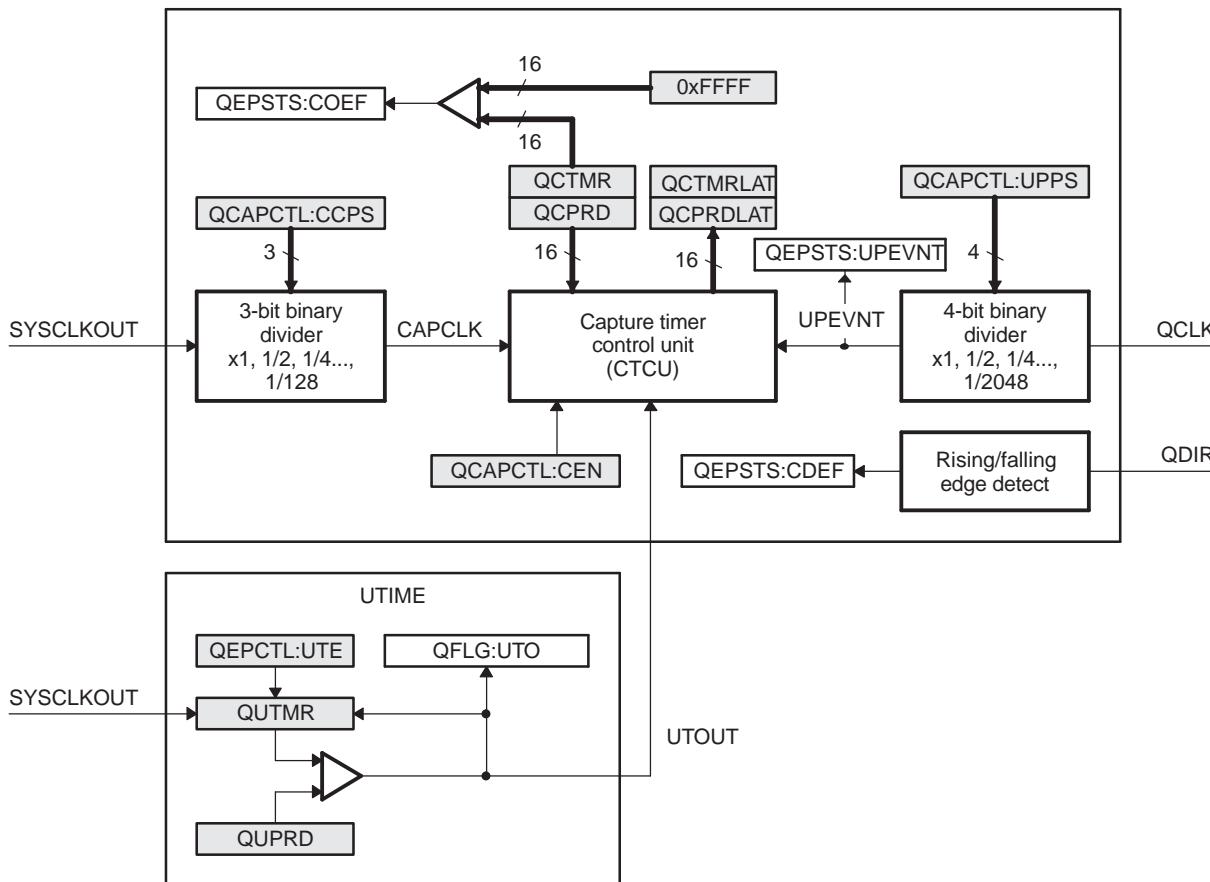
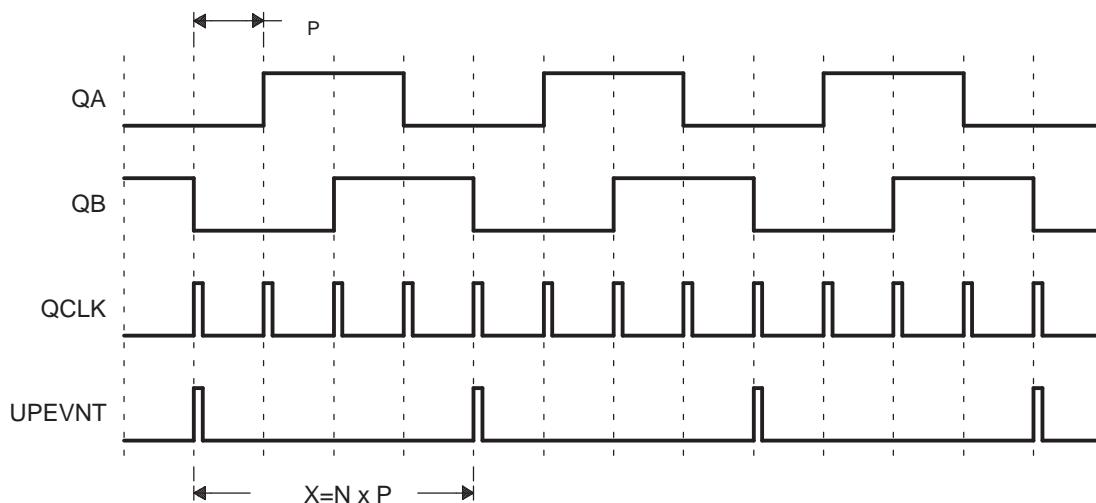
If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

[Figure 15-145](#) shows the capture unit operation along with the position counter.

---

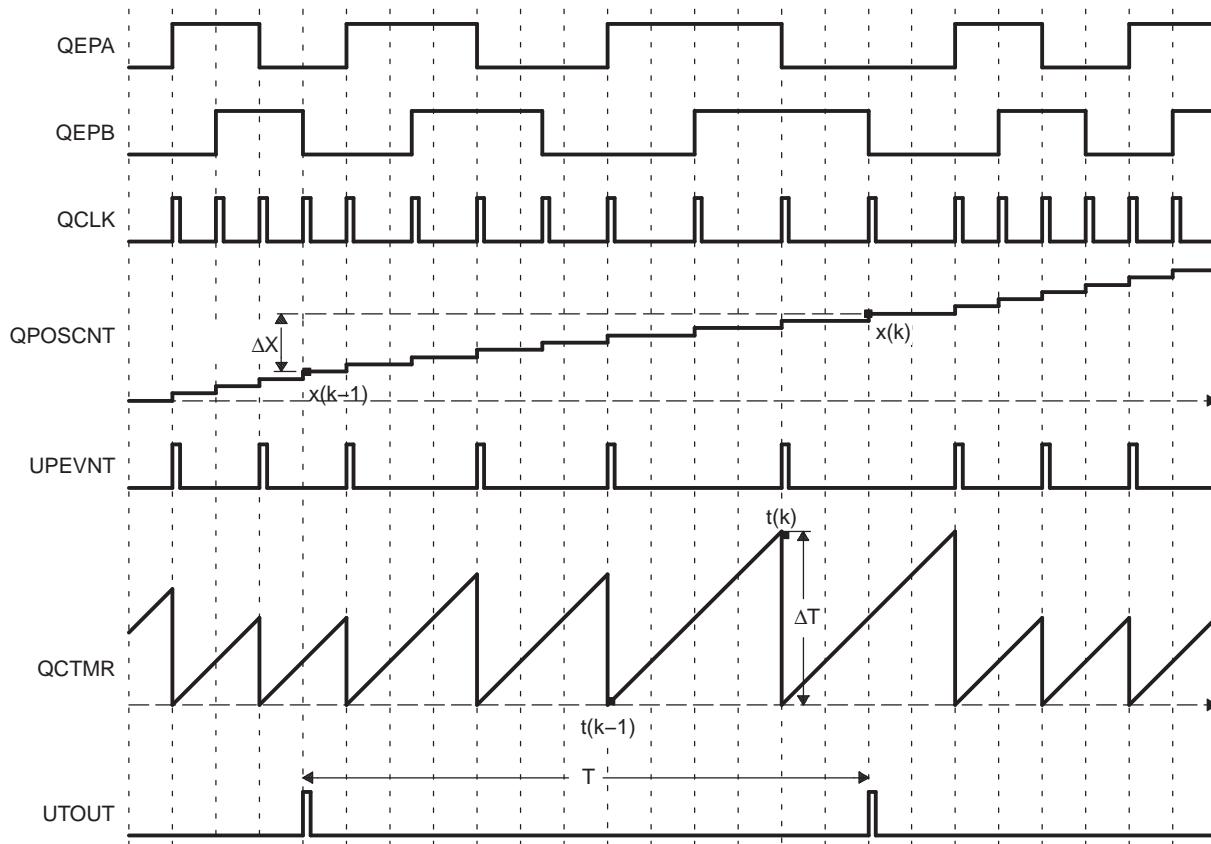
**NOTE:** The QCAPCTL register should not be modified dynamically (such as switching CAPCLK prescaling mode from QCLK/4 to QCLK/8). The capture unit must be disabled before changing the prescaler.

---

**Figure 15-143. eQEP Edge Capture Unit**

**Figure 15-144. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)**


N - Number of quadrature periods selected using QCAPCTL[UPPS] bits

Figure 15-145. eQEP Edge Capture Unit - Timing Details



Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k - 1)}{T} = \frac{\Delta X}{T} \quad (4)$$

where

$v(k)$ : Velocity at time instant k

$x(k)$ : Position at time instant k

$x(k-1)$ : Position at time instant k - 1

T: Fixed unit time or inverse of velocity calculation rate

$\Delta X$ : Incremental position movement in unit time

X: Fixed unit position

$\Delta T$ : Incremental time elapsed for unit position movement

$t(k)$ : Time instant "k"

$t(k-1)$ : Time instant "k - 1"

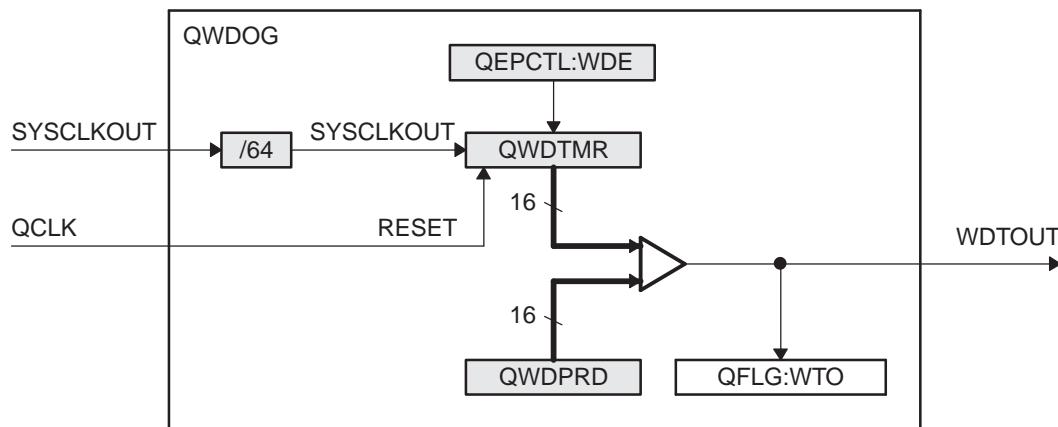
Unit time (T) and unit period (X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOS LAT and QCPRD LAT registers.

| Parameter  | Relevant Register to Configure or Read the Information                  |
|------------|---|
| T          | Unit Period Register (QUPRD)  |
| $\Delta X$ | Incremental Position = QPOSLAT(k) - QPOSLAT(K - 1)                      |
| X          | Fixed unit position defined by sensor resolution and ZCAPCTL[UPPS] bits |
| $\Delta T$ | Capture Period Latch (QCPRDLAT)   |

### 15.4.2.6 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

**Figure 15-146. eQEP Watchdog Timer**

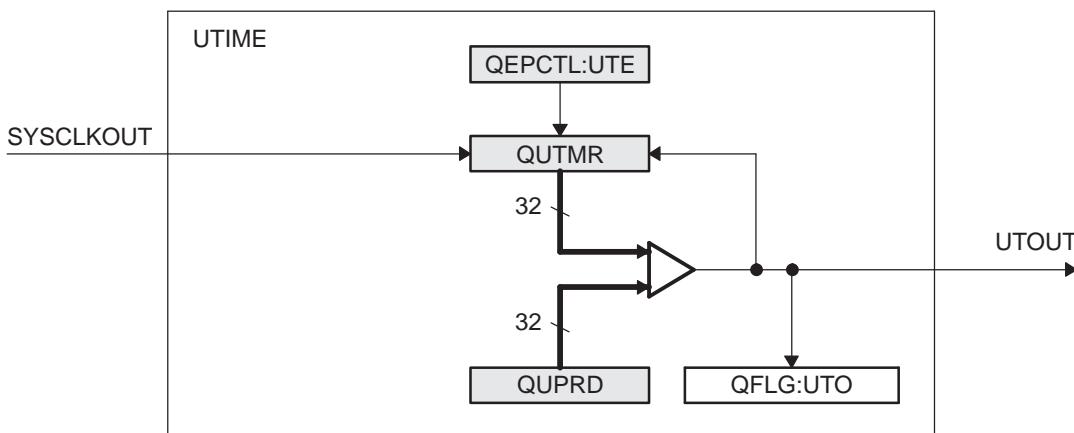


### 15.4.2.7 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD).

The eQEP peripheral can be configured to latch the position counter, capture timer, and capture period values on a unit time out event so that latched values are used for velocity calculation as described in Section [Section 15.4.2.5](#).

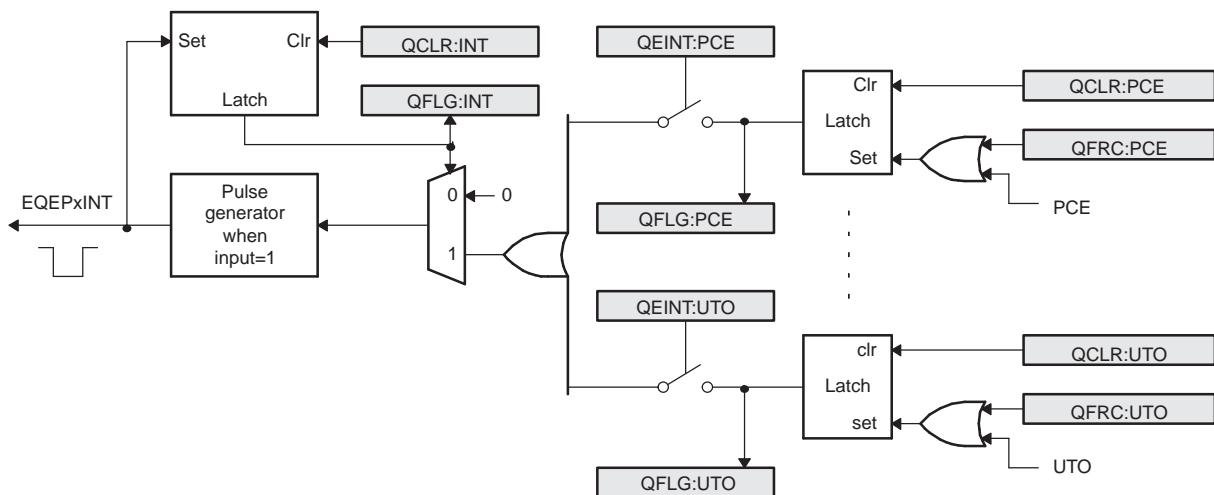
**Figure 15-147. eQEP Unit Time Base**



### 15.4.2.8 eQEP Interrupt Structure

[Figure 15-148](#) shows how the interrupt mechanism works in the EQEP module.

**Figure 15-148. EQEP Interrupt Generation**



Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL, and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated only to the interrupt controller if any of the interrupt events is enabled, the flag bit is 1 and the INT flag bit is 0. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

### 15.4.3 eQEP Registers

Table 15-123 lists the registers with their memory locations, sizes, and reset values.

**Table 15-123. eQEP Registers**

| Offset | Acronym    | Register Description                          | Size(x16)/ #shadow | Section                           |
|--------|------------|---|--------------------|-----------------------------------|
| 0h     | QPOS_CNT   | eQEP Position Counter Register                | 2/0                | <a href="#">Section 15.4.3.1</a>  |
| 4h     | QPOS_INIT  | eQEP Position Counter Initialization Register | 2/0                | <a href="#">Section 15.4.3.2</a>  |
| 8h     | QPOS_MAX   | eQEP Maximum Position Count Register          | 2/0                | <a href="#">Section 15.4.3.3</a>  |
| Ch     | QPOS_CMP   | eQEP Position-Compare Register                | 2/1                | <a href="#">Section 15.4.3.4</a>  |
| 10h    | QPOS_INDEX | eQEP Index Position Latch Register            | 2/0                | <a href="#">Section 15.4.3.5</a>  |
| 14h    | QPOSSLAT   | eQEP Strobe Position Latch Register           | 2/0                | <a href="#">Section 15.4.3.6</a>  |
| 18h    | QPOS_LAT   | eQEP Position Counter Latch Register          | 2/0                | <a href="#">Section 15.4.3.7</a>  |
| 1Ch    | QUTMR      | eQEP Unit Timer Register                      | 2/0                | <a href="#">Section 15.4.3.8</a>  |
| 20h    | QUPRD      | eQEP Unit Period Register                     | 2/0                | <a href="#">Section 15.4.3.9</a>  |
| 24h    | QWDTMR     | eQEP Watchdog Timer Register                  | 1/0                | <a href="#">Section 15.4.3.10</a> |
| 26h    | QWDPRD     | eQEP Watchdog Period Register                 | 1/0                | <a href="#">Section 15.4.3.11</a> |
| 28h    | QDECCTL    | eQEP Decoder Control Register                 | 1/0                | <a href="#">Section 15.4.3.12</a> |
| 2Ah    | QEPCCTL    | eQEP Control Register                         | 1/0                | <a href="#">Section 15.4.3.13</a> |
| 2Ch    | QCAPCTL    | eQEP Capture Control Register                 | 1/0                | <a href="#">Section 15.4.3.14</a> |
| 2Eh    | QPOS_CTL   | eQEP Position-Compare Control Register        | 1/0                | <a href="#">Section 15.4.3.15</a> |
| 30h    | QEINT      | eQEP Interrupt Enable Register                | 1/0                | <a href="#">Section 15.4.3.16</a> |
| 32h    | QFLG       | eQEP Interrupt Flag Register                  | 1/0                | <a href="#">Section 15.4.3.17</a> |
| 34h    | QCLR       | eQEP Interrupt Clear Register                 | 1/0                | <a href="#">Section 15.4.3.18</a> |
| 36h    | QFRC       | eQEP Interrupt Force Register                 | 1/0                | <a href="#">Section 15.4.3.19</a> |
| 38h    | QEPESTS    | eQEP Status Register                          | 1/0                | <a href="#">Section 15.4.3.20</a> |
| 3Ah    | QCTMR      | eQEP Capture Timer Register                   | 1/0                | <a href="#">Section 15.4.3.21</a> |
| 3Ch    | QCPRD      | eQEP Capture Period Register                  | 1/0                | <a href="#">Section 15.4.3.22</a> |
| 3Eh    | QCTMRLAT   | eQEP Capture Timer Latch Register             | 1/0                | <a href="#">Section 15.4.3.23</a> |
| 40h    | QCPRDLAT   | eQEP Capture Period Latch Register            | 1/0                | <a href="#">Section 15.4.3.24</a> |
| 5Ch    | REVID      | eQEP Revision ID Register                     | 2/0                | <a href="#">Section 15.4.3.25</a> |

### 15.4.3.1 eQEP Position Counter Register (QPOSCNT)

**Figure 15-149. eQEP Position Counter Register (QPOSCNT)**

|    |         |   |
|----|---------|---|
| 31 | QPOSCNT | 0 |
|    | R/W-0   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-124. eQEP Position Counter Register (QPOSCNT) Field Descriptions**

| Bits | Name    | Value        | Description  |
|------|---------|--------------|--|
| 31-0 | QPOSCNT | 0xFFFF FFFFh | This 32-bit position counter register counts up/down on every eQEP pulse based on direction input. This counter acts as a position integrator whose count value is proportional to position from a give reference point. |

### 15.4.3.2 eQEP Position Counter Initialization Register (QPOSINIT)

**Figure 15-150. eQEP Position Counter Initialization Register (QPOSINIT)**

|    |          |   |
|----|----------|---|
| 31 | QPOSINIT | 0 |
|    | R/W-0    |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-125. eQEP Position Counter Initialization Register (QPOSINIT) Field Descriptions**

| Bits | Name     | Value        | Description  |
|------|----------|--------------|--|
| 31-0 | QPOSINIT | 0xFFFF FFFFh | This register contains the position value that is used to initialize the position counter based on external strobe or index event. The position counter can be initialized through software. |

### 15.4.3.3 eQEP Maximum Position Count Register (QPOSMAX)

**Figure 15-151. eQEP Maximum Position Count Register (QPOSMAX)**

|    |         |   |
|----|---------|---|
| 31 | QPOSMAX | 0 |
|    | R/W-0   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-126. eQEP Maximum Position Count Register (QPOSMAX) Field Descriptions**

| Bits | Name    | Value        | Description  |
|------|---------|--------------|--|
| 31-0 | QPOSMAX | 0xFFFF FFFFh | This register contains the maximum position counter value. |

#### 15.4.3.4 eQEP Position-Compare Register (QPOSCMP)

**Figure 15-152. eQEP Position-Compare Register (QPOSCMP)**

|    |         |   |
|----|---------|---|
| 31 | QPOSCMP | 0 |
|    | R/W-0   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-127. eQEP Position-Compare Register (QPOSCMP) Field Descriptions**

| Bits | Name    | Value        | Description  |
|------|---------|--------------|--|
| 31-0 | QPOSCMP | 0xFFFF FFFFh | The position-compare value in this register is compared with the position counter (QPOSCNT) to generate sync output and/or interrupt on compare match. |

#### 15.4.3.5 eQEP Index Position Latch Register (QPOSILAT)

**Figure 15-153. eQEP Index Position Latch Register (QPOSILAT)**

|    |          |   |
|----|----------|---|
| 31 | QPOSILAT | 0 |
|    | R-0      |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-128. eQEP Index Position Latch Register (QPOSILAT) Field Descriptions**

| Bits | Name     | Value        | Description  |
|------|----------|--------------|--|
| 31-0 | QPOSILAT | 0xFFFF FFFFh | The position-counter value is latched into this register on an index event as defined by the QEPCTL[IEL] bits. |

#### 15.4.3.6 eQEP Strobe Position Latch Register (QPOSSLAT)

**Figure 15-154. eQEP Strobe Position Latch Register (QPOSSLAT)**

|    |          |   |
|----|----------|---|
| 31 | QPOSSLAT | 0 |
|    | R-0      |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-129. eQEP Strobe Position Latch Register (QPOSSLAT) Field Descriptions**

| Bits | Name     | Value        | Description  |
|------|----------|--------------|--|
| 31-0 | QPOSSLAT | 0xFFFF FFFFh | The position-counter value is latched into this register on strobe event as defined by the QEPCTL[SEL] bits. |

### 15.4.3.7 eQEP Position Counter Latch Register (QPOSLAT)

**Figure 15-155. eQEP Position Counter Latch Register (QPOSLAT)**

|    |         |     |
|----|---------|-----|
| 31 | QPOSLAT | 0   |
|    |         | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-130. eQEP Position Counter Latch Register (QPOSLAT) Field Descriptions**

| Bits | Name    | Value        | Description  |
|------|---------|--------------|--|
| 31-0 | QPOSLAT | 0xFFFF FFFFh | The position-counter value is latched into this register on unit time out event. |

### 15.4.3.8 eQEP Unit Timer Register (QUTMR)

**Figure 15-156. eQEP Unit Timer Register (QUTMR)**

|    |       |       |
|----|-------|-------|
| 31 | QUTMR | 0     |
|    |       | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-131. eQEP Unit Timer Register (QUTMR) Field Descriptions**

| Bits | Name  | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | QUTMR | 0xFFFF FFFFh | This register acts as time base for unit time event generation. When this timer value matches with unit time period value, unit time event is generated. |

### 15.4.3.9 eQEP Unit Period Register (QUPRD)

**Figure 15-157. eQEP Unit Period Register (QUPRD)**

|    |       |       |
|----|-------|-------|
| 31 | QUPRD | 0     |
|    |       | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-132. eQEP Unit Period Register (QUPRD) Field Descriptions**

| Bits | Name  | Value        | Description  |
|------|-------|--------------|--|
| 31-0 | QUPRD | 0xFFFF FFFFh | This register contains the period count for unit timer to generate periodic unit time events to latch the eQEP position information at periodic interval and optionally to generate interrupt. |

### 15.4.3.10 eQEP Watchdog Timer Register (QWDTMR)

**Figure 15-158. eQEP Watchdog Timer Register (QWDTMR)**

|    |        |   |
|----|--------|---|
| 15 | QWDTMR | 0 |
|    | R/W-0  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-133. eQEP Watchdog Timer Register (QWDTMR) Field Descriptions**

| Bits | Name   | Value        | Description   |
|------|--------|--------------|---|
| 15-0 | QWDTMR | 0xFFFF FFFFh | This register acts as time base for watch dog to detect motor stalls. When this timer value matches with watch dog period value, watch dog timeout interrupt is generated. This register is reset upon edge transition in quadrature-clock indicating the motion. |

### 15.4.3.11 eQEP Watchdog Period Register (QWDPRD)

**Figure 15-159. eQEP Watchdog Period Register (QWDPRD)**

|    |        |   |
|----|--------|---|
| 15 | QWDPRD | 0 |
|    | R/W-0  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-134. eQEP Watchdog Period Register (QWDPRD) Field Description**

| Bits | Name   | Value   | Description  |
|------|--------|---------|--|
| 15-0 | QWDPRD | 0xFFFFh | This register contains the time-out count for the eQEP peripheral watch dog timer. When the watchdog timer value matches the watchdog period value, a watchdog timeout interrupt is generated. |

**15.4.3.12 QEP Decoder Control Register (QDECCTL)**
**Figure 15-160. QEP Decoder Control Register (QDECCTL)**

| 15    | 14    | 13    | 12    | 11       | 10    | 9     | 8     |
|-------|-------|-------|-------|----------|-------|-------|-------|
| QSRC  |       | SOEN  | SPSEL | XCR      | SWAP  | IGATE | QAP   |
| R/W-0 |       | R/W-0 | R/W-0 | R/W-0    | R/W-0 | R/W-0 | R/W-0 |
| 7     | 6     | 5     | 4     |          |       |       | 0     |
| QBP   | QIP   | QSP   |       | Reserved |       |       |       |
| R/W-0 | R/W-0 | R/W-0 |       | R-0      |       |       |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-135. eQEP Decoder Control Register (QDECCTL) Field Descriptions**

| Bits  | Name     | Value | Description   |
|-------|----------|-------|---|
| 15-14 | QSRC     | 0-3h  | Position-counter source selection   |
|       |          | 0     | Quadrature count mode (QCLK = iCLK, QDIR = iDIR)  |
|       |          | 1h    | Direction-count mode (QCLK = xCLK, QDIR = xDIR)   |
|       |          | 2h    | UP count mode for frequency measurement (QCLK = xCLK, QDIR = 1)   |
|       |          | 3h    | DOWN count mode for frequency measurement (QCLK = xCLK, QDIR = 0)   |
| 13    | SOEN     | 0     | Sync output-enable  |
|       |          | 1     | Disable position-compare sync output  |
|       |          | 1     | Enable position-compare sync output   |
| 12    | SPSEL    | 0     | Sync output pin selection   |
|       |          | 0     | Index pin is used for sync output   |
|       |          | 1     | Strobe pin is used for sync output  |
| 11    | XCR      | 0     | External clock rate   |
|       |          | 0     | 2x resolution: Count the rising/falling edge  |
|       |          | 1     | 1x resolution: Count the rising edge only   |
| 10    | SWAP     | 0     | Swap quadrature clock inputs. This swaps the input to the quadrature decoder, reversing the counting direction. |
|       |          | 0     | Quadrature-clock inputs are not swapped   |
|       |          | 1     | Quadrature-clock inputs are swapped   |
| 9     | IGATE    | 0     | Index pulse gating option   |
|       |          | 0     | Disable gating of Index pulse   |
|       |          | 1     | Gate the index pin with strobe  |
| 8     | QAP      | 0     | QEPA input polarity   |
|       |          | 0     | No effect   |
|       |          | 1     | Negates QEPA input  |
| 7     | QBP      | 0     | QEPB input polarity   |
|       |          | 0     | No effect   |
|       |          | 1     | Negates QEPB input  |
| 6     | QIP      | 0     | QEPI input polarity   |
|       |          | 0     | No effect   |
|       |          | 1     | Negates QEPI input  |
| 5     | QSP      | 0     | QEPS input polarity   |
|       |          | 0     | No effect   |
|       |          | 1     | Negates QEPS input  |
| 4-0   | Reserved | 0     | Always write as 0   |

### 15.4.3.13 eQEP Control Register (QEPCCTL)

**Figure 15-161. eQEP Control Register (QEPCCTL)**

| 15         | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
|------------|-------|-------|-------|-------|-------|-------|-------|
| FREE, SOFT |       | PCRM  |       | SEI   |       | IEI   |       |
| R/W-0      |       | R/W-0 |       | R/W-0 |       | R/W-0 |       |
| 7          | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| SWI        | SEL   | IEL   | R/W-0 | PHEN  | QCLM  | UTE   | WDE   |
| R/W-0      | R/W-0 | R/W-0 |       | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 15-136. eQEP Control Register (QEPCCTL) Field Descriptions**

| Bits  | Name       | Value | Description  |
|-------|------------|-------|--|
| 15-14 | FREE, SOFT | 0-3h  | Emulation Control Bits<br><b>QPOSCNT behavior:</b><br>0 Position counter stops immediately on emulation suspend.<br>1h Position counter continues to count until the rollover.<br>2h-3h Position counter is unaffected by emulation suspend.   |
|       |            | 0     | <b>QWDTMR behavior:</b><br>0 Watchdog counter stops immediately.<br>1 Watchdog counter counts until WD period match roll over.<br>2h-3h Watchdog counter is unaffected by emulation suspend.   |
|       |            | 1     | <b>QUTMR behavior:</b><br>0 Unit timer stops immediately.<br>1h Unit timer counts until period rollover.<br>2h-3h Unit timer is unaffected by emulation suspend.   |
|       |            | 2h-3h | <b>QCTMR behavior:</b><br>0 Capture Timer stops immediately.<br>1h Capture Timer counts until next unit period event.<br>2h-3h Capture Timer is unaffected by emulation suspend.   |
| 13-12 | PCRM       | 0-3h  | Position counter reset mode<br>0 Position counter reset on an index event<br>1h Position counter reset on the maximum position<br>2h Position counter reset on the first index event<br>3h Position counter reset on a unit time event   |
| 11-10 | SEI        | 0-3h  | Strobe event initialization of position counter<br>0 Does nothing (action disabled)<br>1h Does nothing (action disabled)<br>2h Initializes the position counter on rising edge of the QEPS signal<br><b>Clockwise Direction:</b> Initializes the position counter on the rising edge of QEPS strobe<br><b>Counter Clockwise Direction:</b> Initializes the position counter on the falling edge of QEPS strobe |
| 9-8   | IEI        | 0-3h  | Index event initialization of position counter<br>0 Do nothing (action disabled)<br>1h Do nothing (action disabled)<br>2h Initializes the position counter on the rising edge of the QEPI signal (QPOSCNT = QPOSINIT)<br>3h Initializes the position counter on the falling edge of QEPI signal (QPOSCNT = QPOSINIT)   |
| 7     | SWI        | 0     | Software initialization of position counter<br>0 Do nothing (action disabled)<br>1 Initialize position counter, this bit is cleared automatically  |

**Table 15-136. eQEP Control Register (QEPCCTL) Field Descriptions (continued)**

| Bits | Name | Value | Description   |
|------|------|-------|---|
| 6    | SEL  | 0     | Strobe event latch of position counter<br><br>The position counter is latched on the rising edge of QEPS strobe (QPOSSLAT = POSCCNT). Latching on the falling edge can be done by inverting the strobe input using the QSP bit in the QDECCTL register.   |
|      |      | 1     | <b>Clockwise Direction:</b> Position counter is latched on rising edge of QEPS strobe<br><b>Counter Clockwise Direction:</b> Position counter is latched on falling edge of QEPS strobe   |
| 5-4  | IEL  | 0-3h  | Index event latch of position counter (software index marker)<br><br>0 Reserved<br>1h Latches position counter on rising edge of the index signal<br>2h Latches position counter on falling edge of the index signal<br>3h Software index marker. Latches the position counter and quadrature direction flag on index event marker. The position counter is latched to the QPOSILAT register and the direction flag is latched in the QEPSTS[QDLF] bit. This mode is useful for software index marking. |
| 3    | PHEN | 0     | Quadrature position counter enable/software reset<br><br>Reset the eQEP peripheral internal operating flags/read-only registers. Control/configuration registers are not disturbed by a software reset.   |
|      |      | 1     | eQEP position counter is enabled  |
| 2    | QCLM | 0     | eQEP capture latch mode<br><br>Latch on position counter read by CPU. Capture timer and capture period values are latched into QCTMRLAT and QCPRDLAT registers when CPU reads the QPOSCNT register.   |
|      |      | 1     | Latch on unit time out. Position counter, capture timer and capture period values are latched into QPOSLAT, QCTMRLAT and QCPRDLAT registers on unit time out.   |
| 1    | UTE  | 0     | eQEP unit timer enable<br><br>Disable eQEP unit timer   |
|      |      | 1     | Enable unit timer   |
| 0    | WDE  | 0     | eQEP watchdog enable<br><br>Disable the eQEP watchdog timer   |
|      |      | 1     | Enable the eQEP watchdog timer  |

### 15.4.3.14 eQEP Capture Control Register (QCAPCTL)

**Figure 15-162. eQEP Capture Control Register (QCAPCTL)**

|          |    |          |       |      |       |
|----------|----|----------|-------|------|-------|
| 15       | 14 |          |       |      | 8     |
| CEN      |    | Reserved |       |      |       |
| R/W-0    |    |          | R-0   |      |       |
| 7        | 6  | 4        | 3     | 0    |       |
| Reserved |    | CCPS     |       | UPPS |       |
|          |    | R-0      | R/W-0 |      | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-137. eQEP Capture Control Register (QCAPCTL) Field Descriptions**

| Bits | Name     | Value | Description                        |
|------|----------|-------|------------------------------------|
| 15   | CEN      | 0     | Enable eQEP capture                |
|      |          | 1     | eQEP capture unit is disabled      |
| 14-7 | Reserved | 0     | eQEP capture unit is enabled       |
| 6-4  | CCPS     | 0-7h  | eQEP capture timer clock prescaler |
|      |          | 0     | CAPCLK = SYSCLKOUT/1               |
|      |          | 1h    | CAPCLK = SYSCLKOUT/2               |
|      |          | 2h    | CAPCLK = SYSCLKOUT/4               |
|      |          | 3h    | CAPCLK = SYSCLKOUT/8               |
|      |          | 4h    | CAPCLK = SYSCLKOUT/16              |
|      |          | 5h    | CAPCLK = SYSCLKOUT/32              |
|      |          | 6h    | CAPCLK = SYSCLKOUT/64              |
|      |          | 7h    | CAPCLK = SYSCLKOUT/128             |
| 3-0  | UPPS     | 0-Fh  | Unit position event prescaler      |
|      |          | 0     | UPEVNT = QCLK/1                    |
|      |          | 1h    | UPEVNT = QCLK/2                    |
|      |          | 2h    | UPEVNT = QCLK/4                    |
|      |          | 3h    | UPEVNT = QCLK/8                    |
|      |          | 4h    | UPEVNT = QCLK/16                   |
|      |          | 5h    | UPEVNT = QCLK/32                   |
|      |          | 6h    | UPEVNT = QCLK/64                   |
|      |          | 7h    | UPEVNT = QCLK/128                  |
|      |          | 8h    | UPEVNT = QCLK/256                  |
|      |          | 9h    | UPEVNT = QCLK/512                  |
|      |          | Ah    | UPEVNT = QCLK/1024                 |
|      |          | Bh    | UPEVNT = QCLK/2048                 |
|      |          | Ch-Fh | Reserved                           |

**15.4.3.15 eQEP Position-Compare Control Register (QPOSCTL)**
**Figure 15-163. eQEP Position-Compare Control Register (QPOSCTL)**

| 15     | 14     | 13    | 12    | 11    | 8 |
|--------|--------|-------|-------|-------|---|
| PCSHDW | PCLOAD | PCPOL | PCE   | PCSPW |   |
| R/W-0  | R/W-0  | R/W-0 | R/W-0 | R/W-0 |   |
| 7      |        |       |       |       | 0 |
|        |        |       | PCSPW |       |   |
|        |        |       | R/W-0 |       |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-138. eQEP Position-Compare Control Register (QPOSCTL) Field Descriptions**

| Bit  | Name   | Value   | Description   |
|------|--------|---------|---|
| 15   | PCSHDW | 0       | Position-compare shadow enable<br>Shadow disabled, load Immediate |
|      |        | 1       | Shadow enabled  |
| 14   | PCLOAD | 0       | Position-compare shadow load mode<br>Load on QPOSCTN = 0          |
|      |        | 1       | Load when QPOSCTN = QPOSCTMP                                      |
| 13   | PCPOL  | 0       | Polarity of sync output<br>Active HIGH pulse output               |
|      |        | 1       | Active LOW pulse output   |
| 12   | PCE    | 0       | Position-compare enable/disable<br>Disable position compare unit  |
|      |        | 1       | Enable position compare unit                                      |
| 11-0 | PCSPW  | 0-FFFh  | Select-position-compare sync output pulse width                   |
|      |        | 0       | 1 × 4 × SYSCLKOUT cycles  |
|      |        | 1h      | 2 × 4 × SYSCLKOUT cycles  |
|      |        | 2h-FFFh | 3 × 4 × SYSCLKOUT cycles to 4096 × 4 × SYSCLKOUT cycles           |

**15.4.3.16 eQEP Interrupt Enable Register (QEINT)**
**Figure 15-164. eQEP Interrupt Enable Register (QEINT)**

| 15       | 12    | 11    | 10    | 9     | 8     |
|----------|-------|-------|-------|-------|-------|
| Reserved | UTO   | IEL   | SEL   | PCM   |       |
| R-0      | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7        | 6     | 5     | 4     | 3     | 2     |
| PCR      | PCO   | PCU   | WTO   | QDC   | PHE   |
| R/W-0    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|          |       |       |       |       | R-0   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-139. eQEP Interrupt Enable Register (QEINT) Field Descriptions**

| Bits  | Name     | Value | Description                                  |
|-------|----------|-------|--|
| 15-12 | Reserved | 0     | Always write as 0                            |
| 11    | UTO      | 0     | Unit time out interrupt enable               |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 10    | IEL      | 0     | Index event latch interrupt enable           |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 9     | SEL      | 0     | Strobe event latch interrupt enable          |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 8     | PCM      | 0     | Position-compare match interrupt enable      |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 7     | PCR      | 0     | Position-compare ready interrupt enable      |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 6     | PCO      | 0     | Position counter overflow interrupt enable   |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 5     | PCU      | 0     | Position counter underflow interrupt enable  |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 4     | WTO      | 0     | Watchdog time out interrupt enable           |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 3     | QDC      | 0     | Quadrature direction change interrupt enable |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 2     | PHE      | 0     | Quadrature phase error interrupt enable      |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 1     | PCE      | 0     | Position counter error interrupt enable      |
|       |          | 1     | Interrupt is disabled                        |
|       |          | 1     | Interrupt is enabled                         |
| 0     | Reserved | 0     | Reserved                                     |

### 15.4.3.17 eQEP Interrupt Flag Register (QFLG)

**Figure 15-165. eQEP Interrupt Flag Register (QFLG)**

| 15  | Reserved |     |     | 12  | 11  | 10  | 9   | 8   |
|-----|----------|-----|-----|-----|-----|-----|-----|-----|
|     | R-0      |     |     |     | R-0 | R-0 | R-0 | R-0 |
| 7   | 6        | 5   | 4   | 3   | 2   | 1   | 0   |     |
| PCR | PCO      | PCU | WTO | QDC | PHE | PCE | INT |     |
| R-0 | R-0      | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -n = value after reset

**Table 15-140. eQEP Interrupt Flag Register (QFLG) Field Descriptions**

| Bits  | Name     | Value | Description   |
|-------|----------|-------|---|
| 15-12 | Reserved | 0     | Always write as 0   |
| 11    | UTO      | 0     | Unit time out interrupt flag  |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | Set by eQEP unit timer period match   |
| 10    | IEL      | 0     | Index event latch interrupt flag  |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set after latching the QPOSCNT to QPOSILAT  |
| 9     | SEL      | 0     | Strobe event latch interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set after latching the QPOSCNT to QPOSSLAT  |
| 8     | PCM      | 0     | eQEP compare match event interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set on position-compare match   |
| 7     | PCR      | 0     | Position-compare ready interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set after transferring the shadow register value to the active position compare register. |
| 6     | PCO      | 0     | Position counter overflow interrupt flag  |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set on position counter overflow.   |
| 5     | PCU      | 0     | Position counter underflow interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set on position counter underflow.  |
| 4     | WTO      | 0     | Watchdog timeout interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | Set by watch dog timeout  |
| 3     | QDC      | 0     | Quadrature direction change interrupt flag  |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | This bit is set during change of direction  |
| 2     | PHE      | 0     | Quadrature phase error interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | Set on simultaneous transition of QEPA and QEPB   |
| 1     | PCE      | 0     | Position counter error interrupt flag   |
|       |          | 1     | No interrupt generated  |
|       |          | 1     | Position counter error  |

**Table 15-140. eQEP Interrupt Flag Register (QFLG) Field Descriptions (continued)**

| Bits | Name | Value  | Description   |
|------|------|--------|---|
| 0    | INT  | 0<br>1 | Global interrupt status flag<br>No interrupt generated<br>Interrupt was generated |

#### 15.4.3.18 eQEP Interrupt Clear Register (QCLR)

**Figure 15-166. eQEP Interrupt Clear Register (QCLR)**

| 15    | 12       | 11    | 10    | 9     | 8     |
|-------|----------|-------|-------|-------|-------|
|       | Reserved | UTO   | IEL   | SEL   | PCM   |
| 15    | R-0      | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7     | 6        | 5     | 4     | 3     | 2     |
| PCR   | PCO      | PCU   | WTO   | QDC   | PHE   |
| R/W-0 | R/W-0    | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|       |          |       |       | 1     | 0     |
|       |          |       |       | INT   |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-141. eQEP Interrupt Clear Register (QCLR) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 15-12 | Reserved | 0      | Always write as 0s   |
| 11    | UTO      | 0<br>1 | Clear unit time out interrupt flag<br>No effect<br>Clears the interrupt flag               |
| 10    | IEL      | 0<br>1 | Clear index event latch interrupt flag<br>No effect<br>Clears the interrupt flag           |
| 9     | SEL      | 0<br>1 | Clear strobe event latch interrupt flag<br>No effect<br>Clears the interrupt flag          |
| 8     | PCM      | 0<br>1 | Clear eQEP compare match event interrupt flag<br>No effect<br>Clears the interrupt flag    |
| 7     | PCR      | 0<br>1 | Clear position-compare ready interrupt flag<br>No effect<br>Clears the interrupt flag      |
| 6     | PCO      | 0<br>1 | Clear position counter overflow interrupt flag<br>No effect<br>Clears the interrupt flag   |
| 5     | PCU      | 0<br>1 | Clear position counter underflow interrupt flag<br>No effect<br>Clears the interrupt flag  |
| 4     | WTO      | 0<br>1 | Clear watchdog timeout interrupt flag<br>No effect<br>Clears the interrupt flag            |
| 3     | QDC      | 0<br>1 | Clear quadrature direction change interrupt flag<br>No effect<br>Clears the interrupt flag |

**Table 15-141. eQEP Interrupt Clear Register (QCLR) Field Descriptions (continued)**

| Bit | Field | Value | Description   |
|-----|-------|-------|---|
| 2   | PHE   | 0     | Clear quadrature phase error interrupt flag   |
|     |       | 1     | No effect   |
|     |       | 1     | Clears the interrupt flag   |
| 1   | PCE   | 0     | Clear position counter error interrupt flag   |
|     |       | 1     | No effect   |
|     |       | 1     | Clears the interrupt flag   |
| 0   | INT   | 0     | Global interrupt clear flag   |
|     |       | 1     | No effect   |
|     |       | 1     | Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1. |

**15.4.3.19 eQEP Interrupt Force Register (QFRC)**
**Figure 15-167. eQEP Interrupt Force Register (QFRC)**

| 15       | 12    | 11    | 10    | 9     | 8     |
|----------|-------|-------|-------|-------|-------|
| Reserved | UTO   | IEL   | SEL   | PCM   |       |
| R-0      | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7        | 6     | 5     | 4     | 3     | 2     |
| PCR      | PCO   | PCU   | WTO   | QDC   | PHE   |
| R/W-0    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|          |       |       |       |       | R-0   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-142. eQEP Interrupt Force Register (QFRC) Field Descriptions**

| Bit   | Field    | Value | Description                                 |
|-------|----------|-------|---|
| 15-12 | Reserved | 0     | Always write as 0s                          |
| 11    | UTO      | 0     | Force unit time out interrupt               |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 10    | IEL      | 0     | Force index event latch interrupt           |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 9     | SEL      | 0     | Force strobe event latch interrupt          |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 8     | PCM      | 0     | Force position-compare match interrupt      |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 7     | PCR      | 0     | Force position-compare ready interrupt      |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 6     | PCO      | 0     | Force position counter overflow interrupt   |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 5     | PCU      | 0     | Force position counter underflow interrupt  |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 4     | WTO      | 0     | Force watchdog time out interrupt           |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 3     | QDC      | 0     | Force quadrature direction change interrupt |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 2     | PHE      | 0     | Force quadrature phase error interrupt      |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 1     | PCE      | 0     | Force position counter error interrupt      |
|       |          | 0     | No effect                                   |
|       |          | 1     | Force the interrupt                         |
| 0     | Reserved | 0     | Always write as 0                           |

### 15.4.3.20 eQEP Status Register (QEPSTS)

**Figure 15-168. eQEP Status Register (QEPSTS)**

|        |      |     |      |       |       |       |      | 15       | 8 |
|--------|------|-----|------|-------|-------|-------|------|----------|---|
|        |      |     |      |       |       |       |      | Reserved |   |
|        |      |     |      |       |       |       |      | R-0      |   |
| 7      | 6    | 5   | 4    | 3     | 2     | 1     | 0    |          |   |
| UPEVNT | FIDF | QDF | QDLF | COEF  | CDEF  | FIMF  | PCEF |          |   |
| R-0    | R-0  | R-0 | R-0  | R/W-1 | R/W-1 | R/W-1 | R-0  |          |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-143. eQEP Status Register (QEPSTS) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Always write as 0  |
| 7    | UPEVNT   | 0     | Unit position event flag   |
|      |          | 1     | No unit position event detected  |
|      |          | 1     | Unit position event detected. Write 1 to clear.  |
| 6    | FDF      | 0     | Direction on the first index marker. Status of the direction is latched on the first index event marker. |
|      |          | 1     | Counter-clockwise rotation (or reverse movement) on the first index event                                |
|      |          | 1     | Clockwise rotation (or forward movement) on the first index event  |
| 5    | QDF      | 0     | Quadrature direction flag  |
|      |          | 0     | Counter-clockwise rotation (or reverse movement)   |
|      |          | 1     | Clockwise rotation (or forward movement)   |
| 4    | QDLF     | 0     | eQEP direction latch flag. Status of direction is latched on every index event marker.                   |
|      |          | 0     | Counter-clockwise rotation (or reverse movement) on index event marker                                   |
|      |          | 1     | Clockwise rotation (or forward movement) on index event marker   |
| 3    | COEF     | 0     | Capture overflow error flag  |
|      |          | 0     | Sticky bit, cleared by writing 1   |
|      |          | 1     | Overflow occurred in eQEP Capture timer (QEPCTMR)  |
| 2    | CDEF     | 0     | Capture direction error flag   |
|      |          | 0     | Sticky bit, cleared by writing 1   |
|      |          | 1     | Direction change occurred between the capture position event.  |
| 1    | FIMF     | 0     | First index marker flag  |
|      |          | 0     | Sticky bit, cleared by writing 1   |
|      |          | 1     | Set by first occurrence of index pulse   |
| 0    | PCEF     | 0     | Position counter error flag. This bit is not sticky and it is updated for every index event.             |
|      |          | 1     | No error occurred during the last index transition.  |
|      |          |       | Position counter error   |

### 15.4.3.21 eQEP Capture Timer Register (QCTMR)

**Figure 15-169. eQEP Capture Timer Register (QCTMR)**

|     |       |   |
|-----|-------|---|
| 15  | QCTMR | 0 |
| R/W |       |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-144. eQEP Capture Time Register (QCTMR) Field Descriptions**

| Bits | Name  | Value   | Description   |
|------|-------|---------|---|
| 15-0 | QCTMR | 0-FFFFh | This register provides time base for edge capture unit. |

### 15.4.3.22 eQEP Capture Period Register (QCPRD)

**Figure 15-170. eQEP Capture Period Register (QCPRD)**

|     |       |   |
|-----|-------|---|
| 15  | QCPRD | 0 |
| R/W |       |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-145. eQEP Capture Period Register (QCPRD) Field Descriptions**

| Bits | Name  | Value   | Description   |
|------|-------|---------|---|
| 15-0 | QCPRD | 0-FFFFh | This register holds the period count value between the last successive eQEP position events |

### 15.4.3.23 eQEP Capture Timer Latch Register (QCTMRLAT)

**Figure 15-171. eQEP Capture Timer Latch Register (QCTMRLAT)**

|    |          |   |
|----|----------|---|
| 15 | QCTMRLAT | 0 |
| R  |          |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-146. eQEP Capture Timer Latch Register (QCTMRLAT) Field Descriptions**

| Bits | Name     | Value   | Description   |
|------|----------|---------|---|
| 15-0 | QCTMRLAT | 0-FFFFh | The eQEP capture timer value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. |

#### 15.4.3.24 eQEP Capture Period Latch Register (QCPRDLAT)

**Figure 15-172. eQEP Capture Period Latch Register (QCPRDLAT)**

|     |          |   |
|-----|----------|---|
| 15  | QCPRDLAT | 0 |
| R/W |          |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 15-147. eQEP Capture Period Latch Register (QCPRDLAT) Field Descriptions**

| Bits | Name     | Value   | Description  |
|------|----------|---------|--|
| 15-0 | QCPRDLAT | 0-FFFFh | eQEP capture period value can be latched into this register on two events viz., unit timeout event, reading the eQEP position counter. |

#### 15.4.3.25 eQEP Revision ID Register (REVID)

**Figure 15-173. eQEP Revision ID Register (REVID)**

|              |     |   |
|--------------|-----|---|
| 31           | REV | 0 |
| R-44D3 1103h |     |   |

LEGEND: R = Read only; -n = value after reset

**Table 15-148. eQEP Revision ID Register (REVID) Field Descriptions**

| Bits | Name | Value      | Description      |
|------|------|------------|------------------|
| 31-0 | REV  | 44D3 1103h | eQEP revision ID |

## **Universal Serial Bus (USB)**

This chapter describes the USB of the device.

| Topic                             | Page |
|-----------------------------------|------|
| 16.1 Introduction .....           | 1678 |
| 16.2 Integration .....            | 1681 |
| 16.3 Functional Description ..... | 1684 |
| 16.4 Supported Use Cases .....    | 1746 |
| 16.5 USB Registers .....          | 1747 |

## 16.1 Introduction

The USB controller provides a low-cost connectivity solution for numerous consumer portable devices by providing a mechanism for data transfer between USB devices with a line/bus speed up to 480 Mbps. The device USB subsystem has two independent USB 2.0 Modules built around two OTG controllers. The OTG supplement feature, the support for a dynamic role change, is also supported. Each port has the support for a dual-role feature allowing for additional versatility enabling operation capability as a host or peripheral. Both ports have identical capabilities and operate independent of each other.

Each USB controller is built around the Mentor USB OTG controller (musbmhdrc) and TI PHY. Each USB controller has user configurable 32K Bytes of Endpoint FIFO and has the support for 15 'Transmit' endpoints and 15 'Receive' endpoints in addition to Endpoint 0. The USB subsystem makes use of the CPPI 4.1 DMA for accelerating data movement via a dedicated DMA hardware.

The two USB modules share the CPPI DMA controller and accompanying Queue Manager, Interrupt Pacer, Power Management module, and PHY/UTMI clock.

Within the descriptions of the USB subsystem that would follow, the term USB controller or USB PHY is used to mean/refer to any of the two USB controllers or PHYs existing within the USB subsystem. The term USB module is used to mean/refer to any of the two USB modules. USB0 is used to refer to one of the USB modules and USB1 is used to refer to the other USB module.

### 16.1.1 Acronyms, Abbreviations, and Definitions

|         |   |
|---------|---|
| AHB     | Advanced High-performance Bus                 |
| CBA     | Common Bus Architecture                       |
| CDC     | Change Data Capture                           |
| CPPI    | Communications Port Programming Interface     |
| CPU     | Central Processing Unit                       |
| DFT     | Design for Test                               |
| DMA     | Direct Memory Access                          |
| DV      | Design Verification                           |
| EOI     | End of Interrupt                              |
| EOP     | End of Packet                                 |
| FIFO    | First-In First-Out                            |
| FS      | Full-Speed USB data rate                      |
| HNP     | Host Negotiation                              |
| HS      | High-Speed USB data rate                      |
| INTD    | Interrupt Distributor                         |
| IP      | Intellectual Property                         |
| ISO     | Isochronous transfer type                     |
| LS      | Low-Speed USB data rate                       |
| MHz     | Megahertz                                     |
| MOP     | Middle of Packet                              |
| OCP     | Open Core Protocol                            |
| OCP HD  | OCP High Performance                          |
| OCP MMR | OCP Memory Mapped Registers                   |
| OTG     | On-The-Go                                     |
| PDR     | Physical Design Requirements                  |
| PHY     | Physical Layer Device                         |
| PPU     | Packet Processing Unit                        |
| RAM     | Random Access Memory                          |
| RNDIS   | Remote Network Driver Interface Specification |

|       |   |
|-------|---|
| RX    | Receive   |
| SCR   | Switched Central Resource   |
| SOC   | System On a Chip  |
| SOP   | Start of Packet   |
| SRP   | Session Resume  |
| TX    | Transmit  |
| USB   | Universal Serial Bus  |
| USB0  | One of the two USB 2.0 Compliant USB Module                       |
| USB1  | One of the two USB2.0 Compliant USB Module                        |
| USBSS | USB Subsystem (contains USB0 and USB1)                            |
| UTMI  | USB 2.0 Transceiver Macrocell Interface                           |
| XDMA  | Transfer DMA (DMA other than CPPI DMA used within the Controller) |

### 16.1.2 USB Features

The main features of the USB subsystem are:

- Contains 2 usb20otg\_f controller modules with the following features:
  - Built around the Mentor USB 2.0 OTG core (musbmhdrc)
  - Supports USB 2.0 peripheral at speeds HS (480 Mb/s) and FS (12 Mb/s)
  - Supports USB 2.0 host or OTG at speeds HS (480 Mb/s), FS (12 Mb/s), and LS (1.5 Mb/s)
  - Supports all modes of transfers (control, bulk, interrupt, and isochronous)
  - Supports high bandwidth ISO mode
  - Supports 16 Transmit (TX) and 16 Receive (RX) endpoints including endpoint 0
  - Supports USB OTG extensions for Session Resume (SRP) and Host Negotiation (HNP)
  - Includes a 32K endpoint FIFO RAM, and supports programmable FIFO sizes
  - Includes RNDIS mode for accelerating RNDIS type protocols using short packet termination over USB
  - Includes CDC Linux mode for accelerating CDC type protocols using short packet termination over USB
  - Includes an RNDIS like mode for terminating RNDIS type protocols without using short packet termination for support of MSC applications
- Includes two USB2.0 OTG PHYs
- Interfaces to the CPU via 3 OCP interfaces:
  - Master OCP HP interface for the DMA
  - Master OCP HP interface for the Queue manager
  - Slave OCP MMR interface
- Includes a CPPI 4.1 compliant DMA controller sub-module with 30 RX and 30 TX simultaneous data connections
- Includes a CPPI 4.1 DMA scheduler
- DMA supports CPPI host descriptor formats
- DMA supports stall on buffer starvation
- Supports data buffer sizes up to 4M bytes
- CPPI FIFO interface per TX/RX endpoint
- Provides a CPPI Queue Manager module with 92 queues for queuing/de-queuing packets.
- DMA pacing logic for interrupts
- Loopback MGC test using the UTMI interfaces

### 16.1.3 Unsupported USB OTG and PHY Features

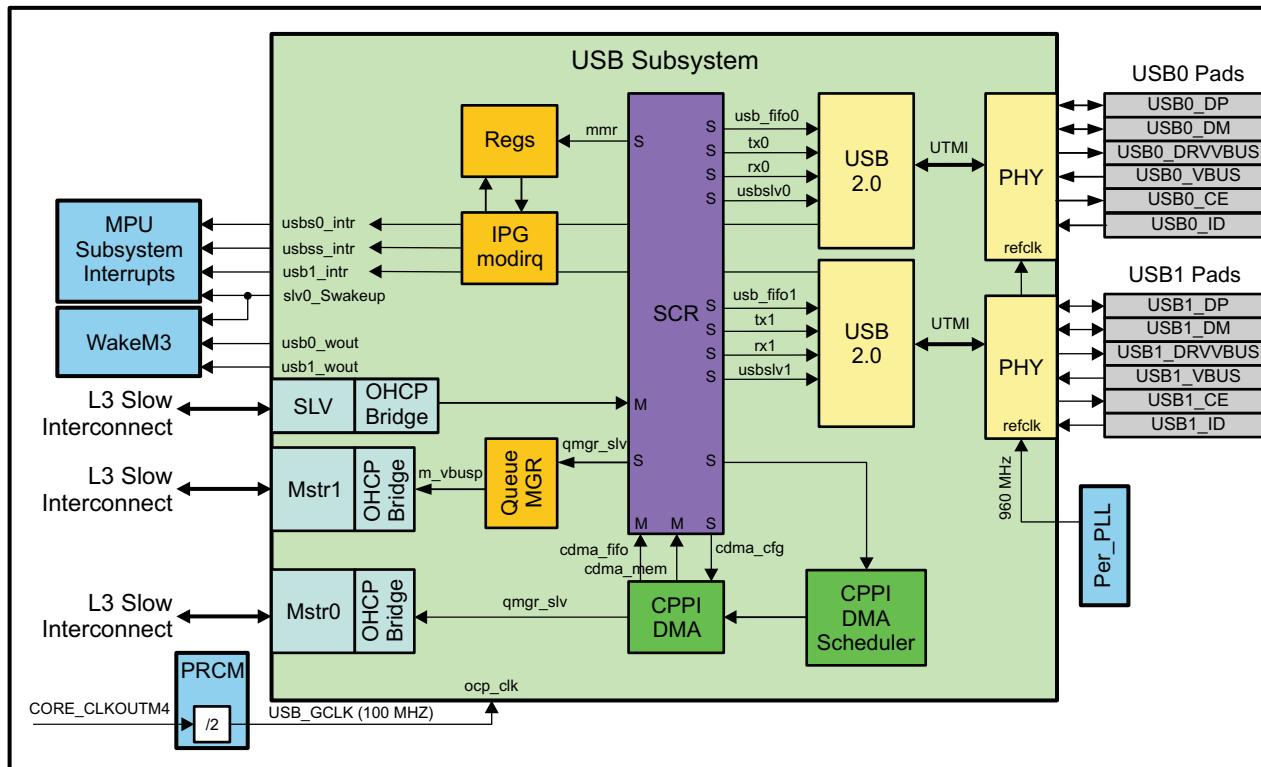
This device supports USBOTG module host, device and OTG features.

## 16.2 Integration

This device implements the USB2.0 OTG dual port module and PHY for interfacing to USB as a peripheral or host.

Figure 16-1 shows the integration of the USB module on this device.

**Figure 16-1. USB Integration**



### 16.2.1 USB Connectivity Attributes

The USB module itself has a very large number of interrupt outputs. For ease of integration, these outputs are all routed to a pair of interrupt aggregators. These aggregator blocks generate a single interrupt on the first occurrence of any interrupt from the USB or DMA logic of the module.

**Table 16-1. USB Connectivity Attributes**

| Attributes          | Type                                     |
|---------------------|--|
| Power domain        | Peripheral Domain                        |
| Clock domain        | PD_PER_L3S_GCLK (OCP)<br>CLKDCOLDO (PHY) |
| Reset signals       | DEF_DOM_RST_N<br>USB POR_N               |
| Idle/Wakeup signals | Smart idle<br>Wakeup                     |

**Table 16-1. USB Connectivity Attributes (continued)**

| Attributes        | Type  |
|-------------------|---|
| Interrupt request | 4 interrupts<br>usbss (USBSSINT) to MPU subsystem<br>usb0 (USBINT0) to MPU subsystem<br>usb1 (USBINT1) to MPU subsystem<br>slv0p_Swakeup (USBWAKEUP) to MPU subsystem and WakeM3<br>2 Wakeup Events to WakeM3<br>usb0_wuout<br>usb1_wuout |
| DMA request       | None  |
| Physical address  | L3 Slow slave port  |

### 16.2.2 USB Clock and Reset Management

Each USB controller has a PHY module that generates the UTMI clock. The UTMI clock is fixed in the UTMI specification for an 8-bit interface at 60 MHz (480 Mb/s). The PHYs require a low-jitter 960-MHz source clock.

**Table 16-2. USB Clock Signals**

| Clock Signal                                 | Max Freq | Reference / Source | Comments                     |
|--|----------|--------------------|------------------------------|
| ocp_clk<br>OCP / Functional clock            | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l3s_gclk<br>From PRCM |
| phy0_other_refclk960m<br>Phy reference clock | 960 MHz  | CLKDCOLDO          | clkdcoldo_po<br>From Per PLL |
| phy1_other_refclk960m<br>Phy reference clock | 960 MHz  | CLKDCOLDO          | clkdcoldo_po<br>From Per PLL |

### 16.2.3 USB Pin List

The USB external interface signals are shown in [Table 16-3](#).

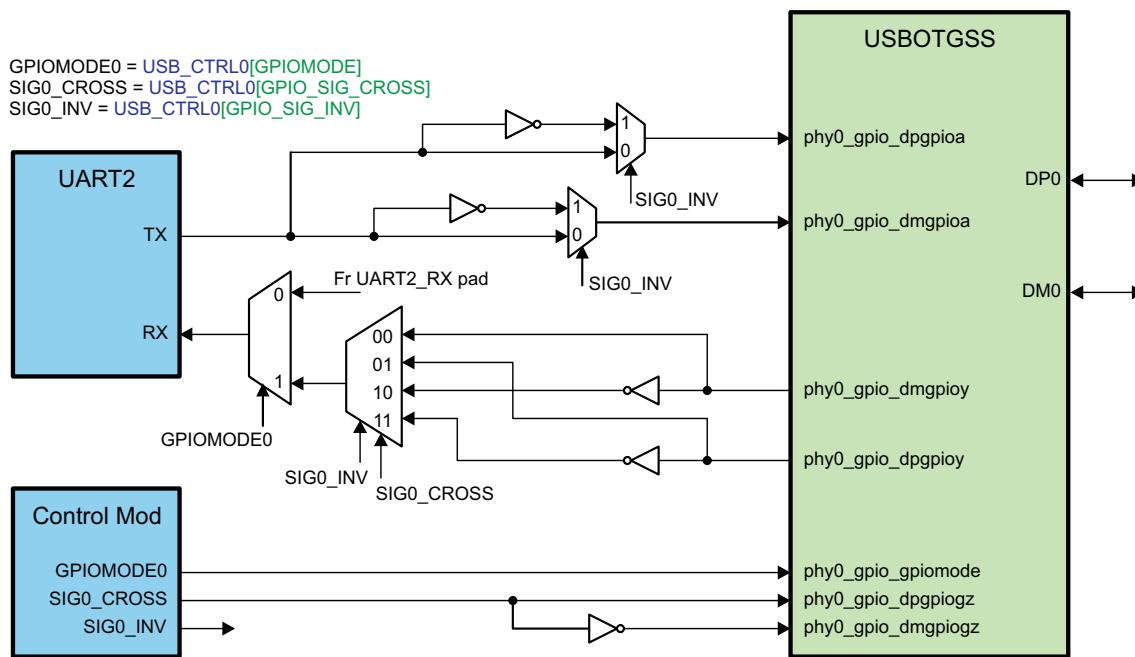
**Table 16-3. USB Pin List**

| Pin                    | Type           |             | Description                                |
|------------------------|----------------|-------------|--|
| USBx_DP <sup>(1)</sup> | USB            | Analog I/O  | USBx data differential pair                |
| USBx_DM <sup>(1)</sup> | GPIO           | Digital I/O |  |
| USBx_DRVVBUS           | Digital output |             | USBx VBUS supply control                   |
| USBx_VBUS              | Analog input   |             | USBx VBUS (input only for voltage sensing) |
| USBx_ID                | Analog input   |             | USBx OTG identification                    |
| USBx_CE                | Digital output |             | USBx Phy charge enable                     |

<sup>(1)</sup> Analog in USB mode; CMOS in GPIO mode.

### 16.2.4 USB GPIO Details

The USB module supports configuration of the DP and DM pins as pass-through GPIOs. In this device, the GPIO mode is used to provide UART over USB interface functionality. Chip level logic allows connection of the UART TX/RX data signals to either DP or DM in either normal or inverted state as shown in [Figure 16-2](#). The diagram shows the UART2 / USB0 port implementation. This logic is also replicated for the UART3 / USB1 ports.

**Figure 16-2. USB GPIO Integration**


### 16.2.5 USB Unbonded PHY Pads

The USBOTGSS includes two USB PHY modules. On packages where only 1 PHY is bonded out to pins (e.g. 13x13 package), the following procedures must be followed in order to ensure that the unbonded PHY pads do not cause issues with USBOTGSS operation:

- The USB Controller corresponding to the unbonded PHY must be placed in Host Mode by setting Bit2 of the USB Core DEVCTL register.
- The unbonded PHY must be placed in the SUSPEND state by setting Bits[1:0] of the USB Core POWER register.
- The Control Module USB\_CTRLx register corresponding to the unbonded PHY must have the following bits programmed as shown:
  - CHGDET\_DIS = 1
  - CM\_PWRDN = 1
  - GPIO MODE = 0

## 16.3 Functional Description

The USB controller within the device supports 15 Transmit endpoints and 15 Receive endpoints in addition to Endpoint 0. (The use of these endpoints for IN and OUT transactions depends on whether the USB controller is being used as a device/peripheral or as a host. When used as a peripheral, IN transactions are processed through TX endpoints and OUT transactions are processed through Rx endpoints. When used as a host, IN transactions are processed through Rx endpoints and OUT transactions are processed through TX endpoints.)

These additional endpoints can be individually configured in software to handle either Bulk transfers (which also allows them to handle Interrupt transfers), Isochronous transfers or Control transfers. Further, the endpoints can also be allocated to different target device functions on the fly – maximizing the number of devices that can be simultaneously supported.

Each endpoint requires a chunk of memory allocated within the FIFO to be associated with it. The USB module has a single block of FIFO RAM (32 Kbytes in size) to be shared by all endpoints. The FIFO for Endpoint 0 is required to be 64 bytes deep and will buffer 1 packet. The first 64 Bytes of the FIFO RAM is reserved by hardware for Endpoint 0 usage and for this reason user software does not need (nor has the capability) to allocate FIFO for Endpoint 0. The rest of the FIFO RAM is user configurable with regard to the other endpoint FIFOs, which may be from 8 to 8192 bytes in size and can buffer either 1 or 2 packets.

Separate FIFOs may be associated with each endpoint: alternatively a TX endpoint and the Rx endpoint with the same Endpoint number can be configured to use the same FIFO, for example to reduce the size of RAM block needed, provided they can never be active at the same time.

The role (host or device) that the USB controller assumes is chosen by user firmware programming the respective USB controller MODE register defined within the USB subsystem space. Note that some USB pins have not been bonded out and the functions of these pins are controlled by user software via dedicated registers.

The user has access to the controller through the OCP slave interface via the CPU. The user can process USB transactions entirely from the CPU or can also use the DMA to perform data transfer. The CPPI DMA can be used to service Endpoints 1 to 15 not Endpoint 0. CPU access method is used to service Endpoint 0 transactions.

### 16.3.1 VBUS Voltage Sourcing Control

When any of the USB controllers assumes the role of a host, the USB is required to supply a 5V power source to an attached device through its VBUS line. In order to achieve this task, the USB controller requires the use of an external power logic (or charge pump) capable of sourcing 5V power. A USB\_DRVVBUS is used as a control signal to enable/disable this external power logic to either source or disable power on the VBUS line. The control on the USB\_DRVVBUS is automatic and is handled by the USB controller. The control should be transparent to the user so long as the proper hardware connection and software initialization are in place. The USB controller drives the USB\_DRVVBUS signal high when it assumes the role of a host while the controller is in session. When assuming the role of a device, the controller drives the USB\_DRVVBUS signal low disabling the external charge pump/power logic; hence, no power is driven on the VBUS line (in this case, power is expected to be sourced by the external host).

Note that both USBs are self-powered and the device does not rely on the voltage on the VBUS line sourced by an external host for controller operation when assuming the role of a device. The power on the VBUS is used to identify the presence of a Host. It is also used to power up the pull-up on the D+ line. The USB PHY would continually monitor the voltage on the VBUS and report the status to USB controller.

### 16.3.2 Pullup/PullDown Resistors

Since the USB controllers are dual role controllers, capable of assuming a role of a host or device, the necessary required pull-up/pull-down resistors can not exist external to the device. These pull-up/pull-down resistors exist internal to the device, within the PHY to be more specific, and are enabled and disabled based on the role the controller assumes allowing dynamic hardware configuration.

When assuming the role of a host, the data lines are pulled low by the PHY enabling the internal 15KOhms resistors. When assuming the role of a device the required 1.5Kohm pull-up resistor on the D+ line is enabled automatically to signify the USB capability to the external host as a FS device (HS operation is negotiated during reset bus condition).

### 16.3.3 Role Assuming Method

For an OTG controller, the usual method followed by the controller to assume the role of a host or a device is governed by the state of the ID pin which in turn is controlled by the USB cable connector type. The device has bonded out these ID pins and allows the control to be handled directly from the connector, i.e. the USB controller would assume the role based on the cable end inserted to the mini or micro A/B connector. An alternate method where the ID pin signal is bypassed and the firmware controls the role the controller assumes exists via register access allowing additional control for systems that do not require the use of the USB ID pin. In this case the USBnMODE [Bit7=IDDIG\_MUX] needs to be programmed with a value of 1 so that Register programming will be used.

Two registers, USB0 Mode Register at offset 10E8h and USB1 Mode Register at offset 18E8h, are used for a user to select the role the USB controller assumes. The user is required to program the corresponding register prior to the USB controller is in session.

### 16.3.4 Clock, PLL, and PHY Initialization

Prior to configuring the USB Module Registers, the USB SubSystem and PHY are required to be released from reset, enable interconnects and controllers clocks as well as configure the USB PHY with appropriate setting. Not all registers related for this task are captured within this document. Please refer to the PRCM for clocking and Control Module for PHY configuration registers definitions.

### 16.3.5 Indexed and Non-Indexed Register Spaces

The USB controller provides two mechanisms for accessing endpoint control and status registers; Indexed and Non-Indexed method.

**Indexed Endpoint Control/Status Register Space.** This register space can be thought of as a region populated with Proxy Registers. This register space is a memory-mapped region at offset 1410h to 141Fh for USB0 and 1C10h to 1C1Fh for USB1. The endpoint register space mapped at this region is selected by programming the corresponding INDEX register (@ offsets 140Eh and 1C0Eh for USB0 and USB1 respectively) of the controller.

By programming the INDEX register with the corresponding endpoint number, the control and status register corresponding to that particular endpoint is accessible from this Indexed Region. In other words, Index Register region behaves as a proxy to access a selected endpoint registers.

**Non-indexed Endpoint Control/Status Register Space.** These regions are dedicated endpoint registers memory-mapped residing at offsets 1500h to 16FFh for USB0 and 1D00h to 1EFFh for USB1. Registers at offset 1500h to 150Fh belong to Endpoint 0; registers at offset 1510h to 151Fh belong to Endpoint 1... and lastly registers at offset 16F0h to 16FFh belong to Endpoint 15 for USB0. Similarly registers at offset 1D00h to 1E0Fh belong to Endpoint 0; registers at offset 1D10h to 1D1Fh belong to Endpoint 1... and finally registers at offset 1DF0h to 1DFFh belong to Endpoint 15 for USB1.

This allows the user/firmware to access an endpoint register region directly from its unique space, as specified within a non-indexed region or from a proxy space by programming the corresponding Index register.

Note: The control and status registers/regions apply to both Tx and Rx of a given Endpoint. That is, Endpoint 1 Tx and Endpoint 1 Rx registers occupy the same register space.

For detailed information about the USB controller registers, see [Section 16.5](#).

### 16.3.6 Dynamic FIFO Sizing

Each USB module supports a total of 32K bytes of FIFO RAM to dynamically allocate FIFO to all endpoints in use. Each endpoint requires a FIFO to be associated with it. There is one set of register per endpoint available for FIFO allocation, excluding endpoint 0. FIFO configuration registers are Indexed registers and cannot be accessed directly; configuration takes place via accessing proxy registers. The INDEX register at addresses 140Eh or 1C0Eh must be set to the appropriate endpoint. In other words, the Endpoint FIFO configuration registers do not have non-indexed regions.

The allocation of FIFO space to the different endpoints requires programming for each Tx and Rx endpoint with the following information:

- The start address of the FIFO within the RAM block
- The maximum size of packet to be supported
- Whether double-buffering is required

(These last two together define the amount of space that needs to be allocated to the FIFO.)

It is the responsibility of the firmware to ensure that all the Tx and Rx endpoints (other than endpoint 0) that are active in the current USB configuration have a block of FIFO RAM allocated for their use. The maximum FIFO size for endpoint 0 required is 64 bytes deep and which is large enough to buffer one packet. For this reason, the first 64 bytes of FIFO memory is reserved for Endpoint 0 usage and this resource allocation is implied to be available at all times and no FIFO allocation is required for Endpoint 0. For endpoints other than endpoint 0, the FIFO RAM interface is configurable and must have a minimum size of 8 bytes and should be capable of buffering either 1 or 2 packets. Separate FIFOs may be associated with each endpoint: alternatively a Tx endpoint and the Rx endpoint with the same Endpoint number can be configured to use the same FIFO, to reduce the size of RAM block needed, provided they can never be active at the same time.

**NOTE:** The option of dynamically setting FIFO sizes only applies to Endpoints 1-15. Endpoint 0 FIFO has a fixed size (64 bytes) and a fixed location which is the first 64 Bytes of FIFO (FIFO addresses 0-63).

### 16.3.7 USB Controller Host and Peripheral Modes Operation

The two USB modules can be used in a range of different environments. They can be used as either a high-speed or a full-speed USB peripheral device attached to a conventional USB host (such as a PC) in a point-to-point type of arrangement. As a host, the USB modules can also be used with another peripheral device of any speed (high-, full-, or low-speed) in a point-to-point arrangement or can be used as the host to a range of peripheral devices in a multi-point setup; one-to-many via hub. Note that the role each USB controller is assuming is independent of each other. Two options (h/w or s/w) exist for a USB role assumption/configuration.

For a USB Peripheral configuration the user has the option of using the cable end to select the role by attaching the mini or micro b side of the cable (can consider this as a h/w option) or use the optional method where the firmware is required to program the respective USB Mode Register IDDIG bit field with a value of '1' prior to the USB controller goes into session (can consider this as a s/w option).

Similarly, for a USB host configuration the user has the option of using the cable end to select the role by attaching the mini or micro a side of the cable (can consider this as a h/w option) or use the optional method where the firmware is required to program the respective USB Mode Register IDDIG bit field with a value of '0' value prior to the USB controller going into session (can consider this as a s/w option).

When using the s/w option, the USB ID pin state is ignored/bypassed (this is a similar configuration where an ID pin does not exist) and the USB2.0 controller role adaptation of a host or peripheral (device) is dependent upon the state of the respective USB Mode Register IDDIG field. If the IDDIG field is programmed by the user with a value of '1' prior to the USB going into session, it would assume the role of a peripheral/device. However, if the IDDIG field is programmed with the value of '0', the USB controller will assume the role of a host. What this means is that the USB cable end, connector, will not be able to control the role of the OTG controller and the user needs to be aware of the firmware program setting prior to performing a USB connection.

The procedure for the USB2.0 OTG controller determining its operating modes (usb controller assuming a role of a host or a peripheral) starts when the USB 2.0 controller goes into a session. The USB 2.0 controller is in session when either it senses a voltage ( $\geq 4.4V$ ) on the USBx\_VBUSIN pin and the controller sets its DEVCTL[SESSION] bit field or when the firmware sets the DEVCTL[SESSION] bit; assuming that it will operating as a host.

When the DEVCTL[SESSION] bit is set, the controller will start sensing the state of the Iddig signal, which in turn is controlled either by the USBx\_ID pin (h/w option) or by the IDDIG bit field of the MODE Register (s/w option). If the state of the iddig signal is found to be low (IDDIG is programmed with '0') then the USB2.0 controller will assume the role of a host and when it finds the iddig signal to be high (IDDIG is programmed with '1') then it will assume the role of a device. Note that iddig is an internal signal that could be driven to high or low via firmware from dedicated registers.

Upon the USB controller determining its role as a host, it will drive the USBx\_DRVVBUS pin high to enable the external power logic so that it starts sourcing the required 5V power (must be  $\geq 4.4V$  but to account for the voltage drop on the cable it is suggested to be in the neighborhood of 4.75V). The USB2.0 controller will then wait for the voltage of the USBx\_VBUSIN to go high. Upon sampling a little time later, if it does not see the voltage on the USBx\_VBUSIN pin to be within a Vbus Valid range ( $\geq 4.4V$ ), it will generate a Vbus error interrupt. Assuming that the voltage level of the USBx\_VBUSIN is found to be within the Vbus Valid range, the USB 2.0 host controller will wait for a device to connect; that is for it to see one of its data lines USB0/1\_DP/DM be pulled high.

When assuming the role of a peripheral, assuming that the firmware has set the POWER[SOFTCONN] bit, and the USBx\_ID pin is floating (h/w option) or USBx Mode Register IDDIG bit field is programmed with '1' (s/w option) and an external host is sourcing power on the USBx\_VBUSIN line then the USB2.0 controller will set the DEVCTL[SESSION] bit instructing the controller to go into session. Note that the voltage on the USBx\_VBUSIN pin must be within Vbus Valid range (i.e. greater or equal to 4.4V). When the controller goes into session, it will force the USB2.0 Controller to sense the state of the iddig signal. Once it senses iddig signal to be high, it will enable its 1.5Kohm pull-up resistor to signify the external host that it is a Full-Speed device. Note that even when operating as a High-Speed peripheral; the USB controller has to first come up as Full-Speed and then later transition to High-Speed. The USB2.0 controller will then waits for a reset signal from the external host. Then after, if High-Speed option has been selected it will negotiate for a High-Speed operation and if its request has been accepted by the host, it will enable its precision 45ohm termination resistors (to stop signal reflection) on its data lines and disable the 1.5Kohm resistor.

### 16.3.8 Protocol Description(s)

This section describes the implementation of the USB protocol(s) by the USB modules.

#### 16.3.8.1 USB Controller Peripheral Mode Operation

The USB controller assumes the role of a peripheral when the USBx\_ID pin is floating or USB Mode Register[iddig=bit8] is set to 1 (provided that iddig\_mux, which is bit7 of USBnMODE is also set to 1) by the user application prior to the controller goes into session. When the USB controller go into session it will assume the role of a device.

Soft connect –After a POR or USB Module soft reset, the SOFTCONN bit of POWER register (bit 6) is cleared to 0. The controller will therefore appear disconnected until the software has set the SOFTCONN bit to 1. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB. Once the SOFTCONN bit has been set, the software can also simulate a disconnect by clearing this bit to 0.

Entry into suspend mode –When operating as a peripheral device, the controller monitors activity on the bus and when no activity has occurred for 3 ms, it goes into Suspend mode. If the Suspend interrupt has been enabled, an interrupt will be generated at this time.

At this point, the controller can then be left active (and hence able to detect when Resume signaling occurs on the USB), or the application may arrange to disable the controller by stopping its clock. However, the controller will not then be able to detect Resume signaling on the USB if clocks are not running. If such is the case, external hardware will be needed to detect Resume signaling (by monitoring the DM and DP signals), so that the clock to the controller can be restarted.

Resume Signaling –When resume signaling occurs on the bus, first the clock to the controller must be restarted if necessary. Then the controller will automatically exit Suspend mode. If the Resume interrupt is enabled, an interrupt will be generated.

Initiating a remote wakeup –If the software wants to initiate a remote wakeup while the controller is in Suspend mode, it should set the POWER[RESUME] bit to 1. The software should leave then this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) before resetting it to 0.

**NOTE:** No resume interrupt will be generated when the software initiates a remote wakeup.

Reset Signaling –When reset signaling or bus condition occurs on the bus, the controller performs the following actions:

- Clears FADDR register to 0
- Clears INDEX register to 0
- Flushes all endpoint FIFOs
- Clears all controller control/status registers
- Generates a reset interrupt
- Enables all interrupts at the core level.

If the HSENA bit within the POWER register (bit 5) is set, the controller also tries to negotiate for high-speed operation. Whether high-speed operation is selected is indicated by HSMODE bit of POWER register (bit 4). When the application software receives a reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

##### 16.3.8.1.1 Control Transactions:Peripheral Mode

Endpoint 0 is the main control endpoint of the core. The software is required to handle all the standard device requests that may be sent or received via endpoint 0. These are described in Universal Serial Bus Specification, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transactions per request/command. To accommodate this, the software needs to take a state machine approach to command decoding and handling.

The Standard Device Requests received by a USB peripheral device can be divided into three categories: Zero Data Requests (in which all the information is included in the command; no additional data is required), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of actions that the software must perform to process these different types of device request.

**NOTE:** The Setup packet associated with any standard device request should include an 8-byte command. Any setup packet containing a command field other than 8 bytes will be automatically rejected by the controller.

#### 16.3.8.1.1.1 Zero Data Requests: Peripheral Mode

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred. Examples of Zero Data standard device requests are:

- SET\_FEATURE
- CLEAR\_FEATURE
- SET\_ADDRESS
- SET\_CONFIGURATION
- SET\_INTERFACE

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI\_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO, decoded and the appropriate action taken.

For example, if the command is SET\_ADDRESS, the 7-bit address value contained in the command should be written to the FADDR register at the completion of the command. The PERI\_CSR0 register should be written by setting the SERV\_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) and also setting the DATAEND bit (bit 3) (indicating that no further data is expected for this request). The interval between setting SERV\_RXPKTRDY bit and DATAEND bit should be very small to avoid getting a SETUPEND error condition. It is highly recommended to set both bits at the same time.

When the host moves to the status stage of the request, a second endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software. The second interrupt is just a confirmation that the request completed successfully. For SET\_ADDRESS command, the address should be written to FADDR register at the completion of the command, i.e. when the status stage interrupt is received.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI\_CSR0 register should be written to set the SERV\_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host moves to the status stage of the request, the controller will send a STALL packet telling the host that the request was not executed. A second endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI\_CSR0) will be set.

If the host sends more data after the DATAEND bit has been set, then the controller will send a STALL packet automatically. An endpoint 0 interrupt will be generated and the SENTSTALL bit (bit 2 of PERI\_CSR0) will be set.

**NOTE:** DMA is not supported for endpoint 0; endpoint 0 is always serviced via CPU.

#### 16.3.8.1.1.2 Write Requests: Peripheral Mode

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a Write standard device request is: SET\_DESCRIPTOR.

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI\_CSR0 will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the PERI\_CSR0 register should then be written to set the SERV\_RXPKTRDY bit (bit 6) (indicating that the command has been read from the FIFO) but in this case the DATAEND bit (bit 3) should not be set (indicating that more data is expected).

When a second endpoint 0 interrupt is received, the PERI\_CSR0 register should be read to check the endpoint status. The RXPKTRDY bit of PERI\_CSR0 should be set to indicate that a data packet has been received. The COUNT0 register should then be read to determine the size of this data packet. The data packet can then be read from the endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the wLength field in the command) is greater than the maximum packet size for endpoint 0, further data packets will be sent. In this case, PERI\_CSR0 should be written to set the SERV\_RXPKTRDY bit, but the DATAEND bit should not be set.

When all the expected data packets have been received, the PERI\_CSR0 register should be written to set the SERV\_RXPKTRDY bit and to set the DATAEND bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software, the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI\_CSR0 register should be written to set the SERV\_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host sends more data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI\_CSR0 (bit 2) will be set.

If the host sends more data after the DATAEND has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI\_CSR0 (bit 2) will be set.

#### **16.3.8.1.1.3 Read Requests: Peripheral Mode**

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of Read Standard Device Requests are:

- GET\_CONFIGURATION
- GET\_INTERFACE
- GET\_DESCRIPTOR
- GET\_STATUS
- SYNCH\_FRAME

The sequence of events will begin, as with all requests, when the software receives an endpoint 0 interrupt. The RXPKTRDY bit of PERI\_CSR0 (bit 0) will also have been set. The 8-byte command should then be read from the endpoint 0 FIFO and decoded. The PERI\_CSR0 register should then be written to set the SERV\_RXPKTRDY bit (bit 6) (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for endpoint 0, only the maximum packet size should be written to the FIFO. The PERI\_CSR0 register should then be written to set the TXPKTRDY bit (bit 1) (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the PERI\_CSR0 register should be written to set the TXPKTRDY bit and to set the DATAEND bit (bit 3) (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software: the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the PERI\_CSR0 register should be written to set the SERV\_RXPKTRDY bit (bit 6) and to set the SENDSTALL bit (bit 5). When the host requests data, the controller will send a STALL to tell the host that the request was not executed. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI\_CSR0 (bit 2) will be set.

If the host requests more data after DATAEND (bit 3) has been set, then the controller will send a STALL. An endpoint 0 interrupt will be generated and the SENTSTALL bit of PERI\_CSR0 (bit 2) will be set.

#### 16.3.8.1.1.4 Endpoint 0 States: Peripheral Mode

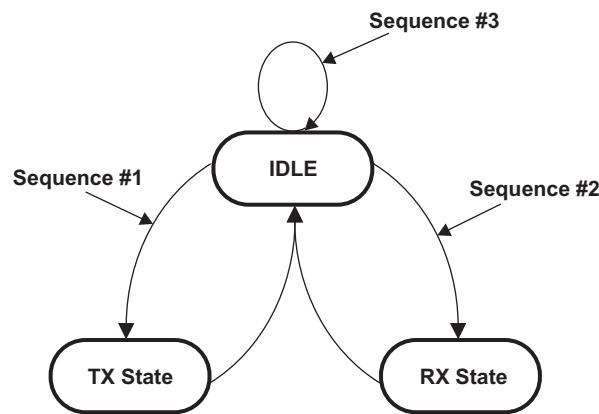
When the USB controller is operating as a peripheral device, the endpoint 0 control needs three modes – IDLE, TX and RX – corresponding to the different phases of the control transfer and the states endpoint 0 enters for the different phases of the transfer (described in later sections).

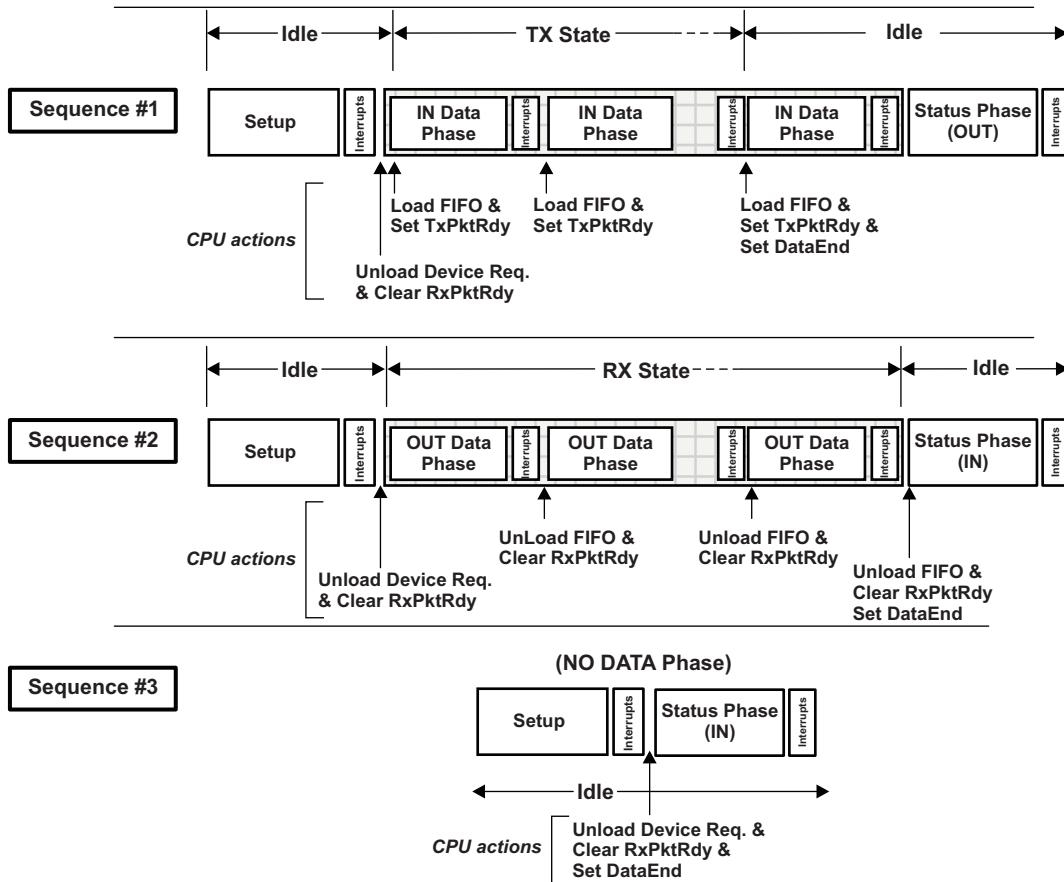
The default mode on power-up or reset should be IDLE. RXPKTRDY bit of PERI\_CSR0 (bit 0) becoming set when endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the controller decodes the descriptor to find whether there is a data phase and, if so, the direction of the data phase of the control transfer (in order to set the FIFO direction). See [Figure 16-3](#).

Depending on the direction of the data phase, endpoint 0 goes into either TX state or RX state. If there is no Data phase, endpoint 0 remains in IDLE state to accept the next device request.

The actions that the CPU needs to take at the different phases of the possible transfers (for example, loading the FIFO, setting TXPKTRDY) are indicated in [Figure 16-4](#).

**Figure 16-3. CPU Actions at Transfer Phases**



**Figure 16-4. Sequence of Transfer**


#### 16.3.8.1.1.4.1 Endpoint 0 Interrupt Generating Events: Peripheral Mode

An Endpoint 0 interrupt is generated when any of the below events occur and Interrupt Service handler should determine the right event and launch the proper handler

The following events will cause Endpoint 0 Interrupt to be generated while the controller is operating assuming the role of a device/peripheral:

- The controller sets the RXPKTRDY bit of PERI\_CSR0 (bit 0) after a valid token has been received and data has been written to the FIFO (legal status condition).
- The controller clears the TXPKTRDY bit of PERI\_CSR0 (bit 1) after the packet of data in the FIFO has been successfully transmitted to the host (legal status condition).
- The controller sets the SENTSTALL bit of PERI\_CSR0 (bit 2) after a control transaction is ended due to a protocol violation (illegal status condition).
- The controller sets the SETUPEND bit of PERI\_CSR0 (bit 4) because a control transfer has ended before DATAEND (bit 3 of PERI\_CSR0) is set (illegal status condition).

In order to determine the cause of the interrupt, the detail below recommends the order of cause determination and execution.

Whenever the endpoint 0 service routine is entered, the software must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the SENTSTALL bit would be set. If the control transfer ends due to a premature end of control transfer, the SETUPEND bit would be set. In either case, the software should abort processing the current control transfer and set the state to IDLE.

Once the software has determined that the interrupt was not generated by an illegal bus state, the next action taken depends on the endpoint state.

If endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the controller receiving data from the bus. The service routine must check for this by testing the RXPKTRDY bit of PERI\_CSR0 (bit 0). If this bit is set, then the controller has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the controller must take. Depending on the command contained within the SETUP packet, endpoint 0 will enter one of three states:

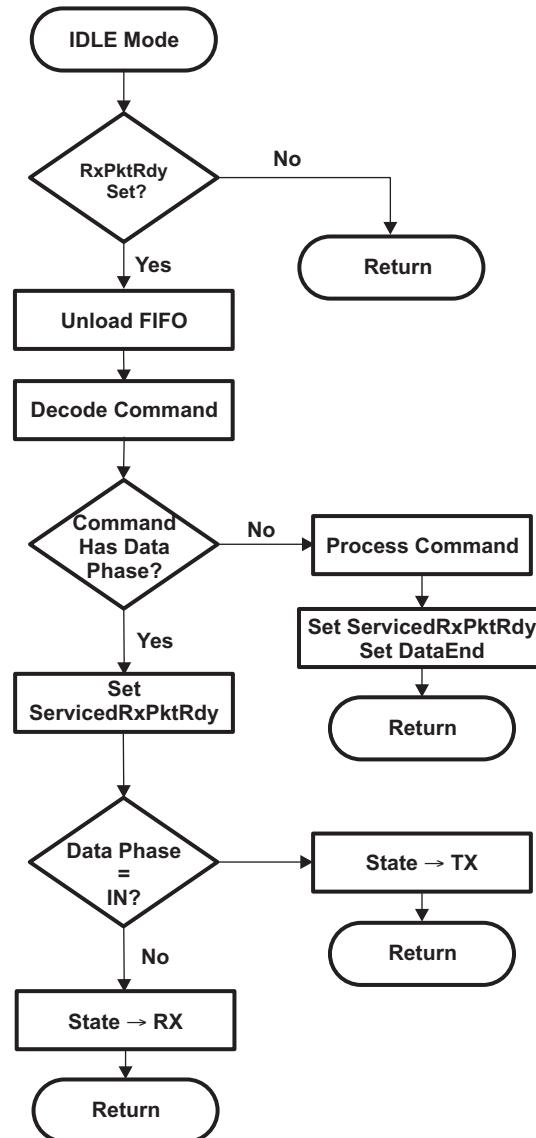
- If the command is a single packet transaction (SET\_ADDRESS, SET\_INTERFACE etc.) without any data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (SET\_DESCRIPTOR etc.), the endpoint will enter RX state.
- If the command has an IN data phase (GET\_DESCRIPTOR etc.), the endpoint will enter TX state.
- If the endpoint 0 is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The software must respond to this either by placing more data in the FIFO if the host is still expecting more data or by setting the DATAEND bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, endpoint 0 should be returned to IDLE state to await the next control transaction (status stage).
- If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The software must respond by unloading the received data from the FIFO. The software must then determine whether it has received all of the expected data. If it has, the software should set the DATAEND bit and return endpoint 0 to IDLE state. If more data is expected, the firmware should set the SERV\_RXPKTRDY bit of PERI\_CSR0 (bit 6) to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

#### 16.3.8.1.1.4.2 Idle Mode: Control Transfer of Peripheral Mode

IDLE mode is the mode the Endpoint 0 control needs to select at power-on or reset and is the mode to which the Endpoint 0 control should return when the RX and TX modes are terminated.

It is also the mode in which the SETUP phase of control transfer is handled (as outlined in [Figure 16-5](#)).

**Figure 16-5. Flow Chart of Setup Stage of a Control Transfer in Peripheral Mode**



#### 16.3.8.1.1.4.3 TX Mode: Control Transfer of Peripheral Mode

When the endpoint is in TX state, all arriving IN tokens need to be treated as part of a data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received while the endpoint is in the TX state, this will cause a SETUPEND condition to occur as the core expects only IN tokens.

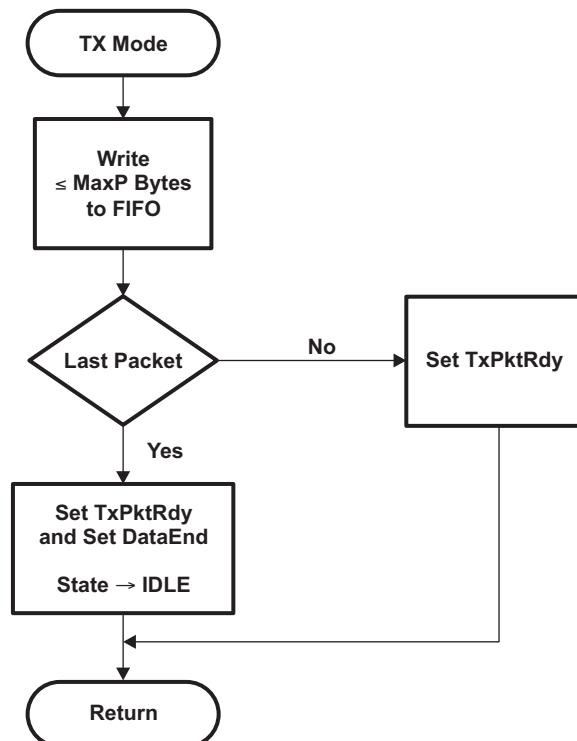
Three events can cause TX mode to be terminated before the expected amount of data has been sent as shown in [Figure 16-6](#):

1. The host sends an invalid token causing a SETUPEND condition (bit 4 of PERI\_CSR0 is set).
2. The firmware sends a packet containing less than the maximum packet size for Endpoint 0.
3. The firmware sends an empty data packet.

Until the transaction is terminated, the firmware simply needs to load the FIFO when it receives an interrupt which indicates that a packet has been sent from the FIFO. An interrupt is generated when TXPKTRDY is cleared.

When the firmware forces the termination of a transfer (by sending a short or empty data packet), it should set the DATAEND bit of PERI\_CSR0 (bit 3) to indicate to the core that the data phase is complete and that the core should next receive an acknowledge packet.

**Figure 16-6. Flow Chart of Transmit Data Stage of a Control Transfer in Peripheral Mode**



#### 16.3.8.1.1.4.4 Rx Mode: Control Transfer of Peripheral Mode

In RX mode, all arriving data should be treated as part of a data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, a SETUPEND condition will occur as the controller expects only OUT tokens.

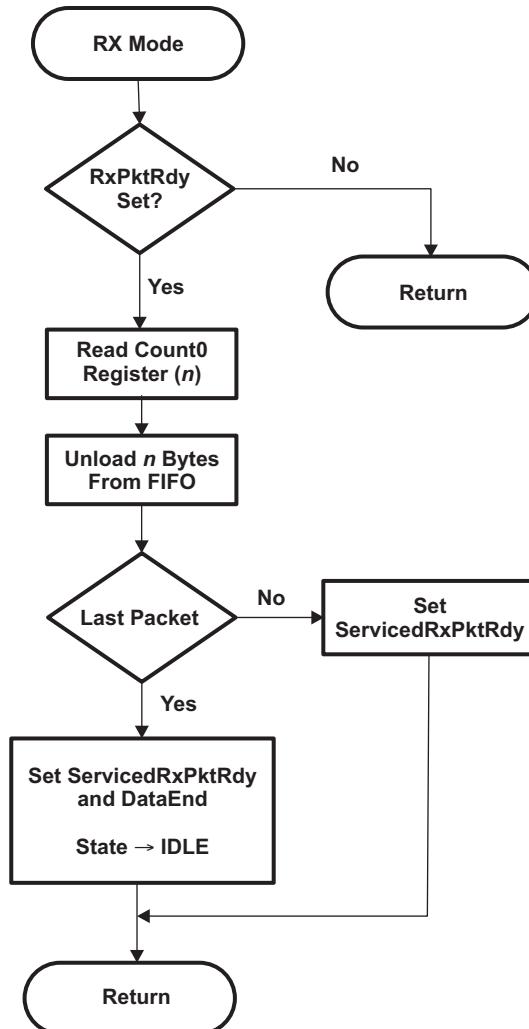
Three events can cause RX mode to be terminated before the expected amount of data has been received as shown in [Figure 16-7](#):

1. The host sends an invalid token causing a SETUPEND condition (setting bit 4 of PERI\_CSR0).
2. The host sends a packet which contains less than the maximum packet size for endpoint 0.
3. The host sends an empty data packet.

Until the transaction is terminated, the software unloads the FIFO when it receives an interrupt that indicates new data has arrived (setting RXPKTRDY bit of PERI\_CSR0) and to clear RXPKTRDY by setting the SERV\_RXPKTRDY bit of PERI\_CSR0 (bit 6).

When the software detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DATAEND bit (bit 3 of PERI\_CSR0) to indicate to the controller that the data phase is complete and that the core should receive an acknowledge packet next.

**Figure 16-7. Flow Chart of Receive Data Stage of a Control Transfer in Peripheral Mode**



#### **16.3.8.1.1.4.5 Error Handling: Control Transfer of Peripheral Mode**

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the software wishes to abort the transfer (for example, because it cannot process the command).

The controller automatically detects protocol errors and sends a STALL packet to the host under the following conditions:

- The host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DATAEND bit (bit 3 of PERI\_CSR0) has been set.
- The host requests more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DATAEND bit in the PERI\_CSR0 register has been set.
- The host sends more than Max Packet Size data bytes in an OUT data packet.
- The host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the controller has sent the STALL packet, it sets the SENTSTALL bit (bit 2 of PERI\_CSR0) and generates an interrupt. When the software receives an endpoint 0 interrupt with the SENTSTALL bit set, it should abort the current transfer, clear the SENTSTALL bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SETUPEND bit (bit 4 of PERI\_CSR0) will be set and an endpoint 0 interrupt generated. When the software receives an endpoint 0 interrupt with the SETUPEND bit set, it should abort the current transfer, set the SERV\_SETUPEND bit (bit 7 of PERI\_CSR0), and return to the IDLE state. If the RXPKTRDY bit (bit 0 of PERI\_CSR0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the SENDSTALL bit (bit 5 of PERI\_CSR0). The controller will then send a STALL packet to the host, set the SENTSTALL bit (bit 2 of PERI\_CSR0) and generate an endpoint 0 interrupt.

#### **16.3.8.1.1.5 Additional Conditions: Control Transfer of Peripheral Mode**

When working as a peripheral device, the controller automatically responds to certain conditions on the USB bus or actions by the host. The details are:

- Stall Issued to Control Transfers
  1. The host sends more data during an OUT Data phase of a Control transfer than was specified in the device request during the SETUP phase. This condition is detected by the controller when the host sends an OUT token (instead of an IN token) after the software has unloaded the last OUT packet and set DATAEND.
  2. The host requests more data during an IN data phase of a Control transfer than was specified in the device request during the SETUP phase. This condition is detected by the controller when the host sends an IN token (instead of an OUT token) after the software has cleared TXPKTRDY and set DATAEND in response to the ACK issued by the host to what should have been the last packet.
  3. The host sends more than MaxPktSize data with an OUT data token.
  4. The host sends the wrong PID for the OUT Status phase of a Control transfer.
  5. The host sends more than a zero length data packet for the OUT Status phase.

- Zero Length Out Data Packets In Control Transfer

A zero length OUT data packet is used to indicate the end of a Control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred (i.e., after the software has set DataEnd). If, however, the host sends a zero length OUT data packet before the entire length of device request has been transferred, this signals the premature end of the transfer. In this case, the controller will automatically flush any IN token loaded by software ready for the Data phase from the FIFO and set SETUPEND bit (bit 4 of PERI\_CSR0).

### 16.3.8.1.2 Bulk Transfer: Peripheral Mode

Bulk transactions are handled by endpoints other than endpoint 0. It is used to handle non-periodic, large bursty communication typically used for a transfer that use any available bandwidth and can also be delayed until bandwidth is available.

#### 16.3.8.1.2.1 Bulk IN Transactions: Peripheral Mode

A Bulk IN transaction is used to transfer non-periodic data from the USB peripheral device to the host.

The following optional features are available for use with a Tx endpoint used in peripheral mode for Bulk IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention

##### 16.3.8.1.2.1.1 Bulk IN Transaction Setup: Peripheral Mode

In configuring a TX endpoint for bulk transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint and the PERI\_TXCSR register (DMAEN and DMAMODE bit fields should be set when using DMA). [Table 16-4](#) displays the PERI\_TXCSR setting when used for Bulk transfer.

**Table 16-4. PERI\_TXCSR Register Bit Configuration for Bulk IN Transactions**

| Bit Field | Bit Name    | Description  |
|-----------|-------------|--|
| Bit 15    | AUTOSET     | Cleared to 0 if using DMA. For CPU Mode use, if AUTOSET bit is set, the TXPKTRDY bit will be automatically set when data of the maximum packet size is loaded into the FIFO. |
| Bit 14    | ISO         | Cleared to 0 for bulk mode operation.  |
| Bit 13    | MODE        | Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint)   |
| Bit 12    | DMAEN       | Set to 1 to enable DMA usage; not needed if CPU is being used to service the Tx Endpoint   |
| Bit 11    | FRCDATATOOG | Cleared to 0 to allow normal data toggle operations.   |
| Bit 10    | DMAMODE     | Set to 1 when DMA is used to service Tx FIFO.  |

When the endpoint is first configured (following a SET\_CONFIGURATION or SET\_INTERFACE command on Endpoint 0), the lower byte of PERI\_TXCSR should be written to set the CLRDATATOOG bit (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state.

Also if there are any data packets in the FIFO, indicated by the FIFONOTEMPTY bit (bit 1 of PERI\_TXCSR) being set, they should be flushed by setting the FLUSHFIFO bit (bit 3 of PERI\_TXCSR).

**NOTE:** It may be necessary to set this bit twice in succession if double buffering is enabled.

#### **16.3.8.1.2.1.2 Bulk IN Operation: Peripheral Mode**

When data is to be transferred over a Bulk IN pipe, a data packet needs to be loaded into the FIFO and the PERI\_TXCSR register written to set the TXPKTRDY bit (bit 0). When the packet has been sent, the TXPKTRDY bit will be cleared by the USB controller and an interrupt generated to signify to the user/application that the next packet can be loaded into the FIFO. If double packet buffering is enabled, then after the first packet has been loaded and the TXPKTRDY bit set, the TXPKTRDY bit will immediately be cleared (will not wait for the loaded first packet to be driven out) by the USB controller and an interrupt generated so that a second packet can be loaded into the FIFO. The software should operate in the same way, loading a packet when it receives an interrupt, regardless of whether double packet buffering is enabled or not.

In the general case, the packet size must not exceed the size specified by the lower 11 bits of the TXMAXP register. This part of the register defines the payload (packet size) for transfers over the USB and is required by the USB Specification to be either 8, 16, 32, 64 (Full-Speed or High-Speed) or 512 bytes (High-Speed only).

The host may determine that all the data for a transfer has been sent by knowing the total amount of data that is expected. Alternatively it may infer that all the data has been sent when it receives a packet which is smaller than the maximum packet size configuration for that particular endpoint (TXMAXP[10-0]). In the latter case, if the total size of the data block is a multiple of this payload, it will be necessary for the function to send a null packet after all the data has been sent. This is done by setting TXPKTRDY when the next interrupt is received, without loading any data into the FIFO.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using the CPPI DMA. A separate section detailing the use of the DMA is discussed within a latter section. Suffix is to say that the PERI\_TXCSR (DMAEN and DMAMODE) bit fields need to be set and the PERI\_TXCSR[AUTOSET] bit cleared at the core level when using the DMA with an endpoint configured for any IN transaction/transfer.

#### **16.3.8.1.2.1.3 Error Handling of Bulk IN Transfer: Peripheral Mode**

If the software wants to shut down the Bulk IN pipe, it should set the SENDSTALL bit (bit 4 of PERI\_TXCSR). When the controller receives the next IN token, it will send a STALL to the host, set the SENTSTALL bit (bit 5 of PERI\_TXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 5 of PERI\_TXCSR) set, it should clear the SENTSTALL bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk IN pipe.

**NOTE:** If the host failed to receive the STALL packet for some reason, it will send another IN token, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk IN pipe.

When a pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATATOG bit in the PERI\_TXCSR register (bit 6).

#### **16.3.8.1.2.2 Bulk OUT Transfer: Peripheral Mode**

A Bulk OUT transaction is used to transfer non-periodic data from the host to the function controller.

The following optional features are available for use with an Rx endpoint used in peripheral mode for Bulk OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of the RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

**NOTE:** When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

#### 16.3.8.1.2.2.1 Bulk OUT Transfer Setup: Peripheral Mode

In configuring an Rx endpoint for Bulk OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. When using DMA for Rx Endpoints, the PERI\_RXCSR needs to have its DMAEN bit set and have its AUTOCLEAR and DMAMODE bits cleared. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI\_RXCSR register should be set as shown in [Table 16-5](#).

**Table 16-5. PERI\_RXCSR Register Bit Configuration for Bulk OUT Transactions**

| Bit Field | Bit Name  | Description   |
|-----------|-----------|---|
| Bit 15    | AUTOCLEAR | Cleared to 0 if using DMA. In CPU Mode of usage, if the CPU sets AUTOCLEAR bit, the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. |
| Bit 14    | ISO       | Cleared to 0 to enable Bulk protocol.   |
| Bit 13    | DMAEN     | Set to 1 if a DMA request is required for this Rx endpoint.   |
| Bit 12    | DISNYET   | Cleared to 0 to allow normal PING flow control. This will affect only high speed transactions.  |
| Bit 11    | DMAMODE   | Always clear this bit to 0.   |

When the Rx endpoint is first configured (following a SET\_CONFIGURATION or SET\_INTERFACE command on Endpoint 0), the lower byte of PERI\_RXCSR should be written to set the CLRDATAOTOG bit (bit 7). This will ensure that the data toggle (which is handled automatically by the USB controller) starts in the correct state.

Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of PERI\_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit (bit 4 of PERI\_RXCSR).

**NOTE:** It may be necessary to set this bit twice in succession if double buffering is enabled.

#### 16.3.8.1.2.2.2 Bulk OUT Operation: Peripheral Mode

When a data packet is received by a Bulk Rx endpoint, the RXPKTRDY bit (bit 0 of PERI\_RXCSR) is set and an interrupt is generated. The software should read the RXCOUNT register for the endpoint to determine the size of the data packet. The data packet should be read from the FIFO, and then the RXPKTRDY bit should be cleared.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host). When a block of data larger than wMaxPacketSize needs to be sent to the function, it will be sent as multiple packets. All the packets will be wMaxPacketSize in size, except the last packet which will contain the residue. The software may use an application specific method of determining the total size of the block and hence when the last packet has been received. Alternatively it may infer that the entire block has been received when it receives a packet which is less than wMaxPacketSize in size. (If the total size of the data block is a multiple of wMaxPacketSize, a null data packet will be sent after the data to signify that the transfer is complete.)

In the general case, the application software will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

### **16.3.8.1.2.2.3 Bulk OUT Error Handling: Peripheral Mode**

If the software wants to shut down the Bulk OUT pipe, it should set the SENDSTALL bit (bit 5 of PERI\_RXCSR). When the controller receives the next packet it will send a STALL to the host, set the SENTSTALL bit (bit 6 of PERI\_RXCSR) and generate an interrupt.

When the software receives an interrupt with the SENTSTALL bit (bit 6 of PERI\_RXCSR) set, it should clear this bit. It should however leave the SENDSTALL bit set until it is ready to re-enable the Bulk OUT pipe.

**NOTE:** If the host failed to receive the STALL packet for some reason, it will send another packet, so it is advisable to leave the SENDSTALL bit set until the software is ready to re-enable the Bulk OUT pipe. When a Bulk OUT pipe is re-enabled, the data toggle sequence should be restarted by setting the CLRDATAOG bit (bit 7) in the PERI\_RXCSR register.

### **16.3.8.1.3 Interrupt Transfer: Peripheral Mode**

An Interrupt IN transaction uses the same protocol as a Bulk IN transaction and the configuration and operation mentioned on prior sections apply to Interrupt transfer too with minor changes. Similarly, an Interrupt OUT transaction uses almost the same protocol as a Bulk OUT transaction and can be used the same way.

Tx endpoints in the USB controller have one feature for Interrupt IN transactions that they do not support in Bulk IN transactions. In Interrupt IN transactions, the endpoints support continuous toggle of the data toggle bit.

This feature is enabled by setting the FRCDATATOOG bit in the PERI\_TXCSR register (bit 11). When this bit is set, the controller will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

Another difference is that interrupt endpoints do not support PING flow control. This means that the controller should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DISNYET bit in the PERI\_RXCSR register (bit 12) should be set to disable the transmission of NYET handshakes in high-speed mode.

Though DMA can be used with an interrupt OUT endpoint, it generally offers little benefit as interrupt endpoints are usually expected to transfer all their data in a single packet.

### **16.3.8.1.4 Isochronous Transfer: Peripheral Mode**

Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device.

#### **16.3.8.1.4.1 Isochronous IN Transactions: Peripheral Mode**

An Isochronous IN transaction is used to transfer periodic data from the function controller to the host.

The following optional features are available for use with a Tx endpoint used in Peripheral mode for Isochronous IN transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

**NOTE:** Double packet buffering is generally advisable for isochronous transactions in order to avoid underrun errors as described in later section.

- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature allows the DMA controller to load packets into the FIFO without processor intervention.

However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI\_TXCSR register needs to be accessed following every packet to check for Underrun errors.

When DMA is enabled and DMAMODE bit of PERI\_TXCSR is set, endpoint interrupt will not be generated for completion of packet transfer. Endpoint interrupt will be generated only in the error conditions.

#### 16.3.8.1.4.1.1 Isochronous IN Transfer Setup: Peripheral Mode

In configuring a Tx endpoint for Isochronous IN transactions, the TXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint) and the PERI\_TXCSR register should be set as shown in [Table 16-6](#).

**Table 16-6. PERI\_TXCSR Register Bit Configuration for Isochronous IN Transactions**

| Bit Field | Bit Name   | Description  |
|-----------|------------|--|
| Bit 14    | ISO        | Set to 1 to enable Isochronous transfer protocol.  |
| Bit 13    | MODE       | Set to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint). |
| Bit 12    | DMAEN      | Set to 1 if DMA Requests have to be enabled.   |
| Bit 11    | FRCDATATOG | Ignored in Isochronous mode.   |
| Bit 10    | DMAMODE    | Set to 1 when DMA is enabled.  |

#### 16.3.8.1.4.1.2 Isochronous IN Transfer Operation: Peripheral Mode

An isochronous endpoint does not support data retries, so if data underrun is to be avoided, the data to be sent to the host must be loaded into the FIFO before the IN token is received. The host will send one IN token per frame (or microframe in High-speed mode), however the timing within the frame (or microframe) can vary. If an IN token is received near the end of one frame and then at the start of the next frame, there will be little time to reload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is sent to the host and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the PERI\_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame(/microframe) before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using either the SOF interrupt or the external SOF\_PULSE signal from the controller to trigger the loading of the next data packet. The SOF\_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF\_PULSE when the SOF packet has been lost.) The interrupts may still be used to set the TXPKTRDY bit in PERI\_TXCSR (bit 0) and to check for data overruns/underruns.

Starting up a double-buffered Isochronous IN pipe can be a source of problems. Double buffering requires that a data packet is not transmitted until the frame(/microframe) after it is loaded. There is no problem if the function loads the first data packet at least a frame(/microframe) before the host sets up the pipe (and therefore starts sending IN tokens). But if the host has already started sending IN tokens by the time the first packet is loaded, the packet may be transmitted in the same frame(/microframe) as it is loaded, depending on whether it is loaded before, or after, the IN token is received. This potential problem can be avoided by setting the ISOUPDATE bit in the POWER register (bit 7). When this bit is set, any data packet loaded into an Isochronous Tx endpoint FIFO will not be transmitted until after the next SOF packet has been received, thereby ensuring that the data packet is not sent too early.

#### 16.3.8.1.4.1.3 Isochronous IN Error Handling: Peripheral Mode

If the endpoint has no data in its FIFO when an IN token is received, it will send a null data packet to the host and set the UNDERRUN bit in the PERI\_TXCSR register (bit 2). This is an indication that the software is not supplying data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the software is loading one packet per frame(/microframe) and it finds that the TXPKTRDY bit in the PERI\_TXCSR register (bit 0) is set when it wants to load the next packet, this indicates that a data packet has not been sent (perhaps because an IN token from the host was corrupted). It is up to the application how it handles this condition: it may choose to flush the unsent packet by setting the FLUSHFIFO bit in the PERI\_TXCSR register (bit 3), or it may choose to skip the current packet.

#### **16.3.8.1.4.2 Isochronous OUT Transactions: Peripheral Mode**

An isochronous OUT transaction is used to transfer periodic data from the host to the function controller.

Following optional features are available for use with an Rx endpoint used in Peripheral mode for Isochronous OUT transactions:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4). **NOTE:** Double packet buffering is generally advisable for Isochronous transactions in order to avoid overrun errors.
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention. However, this feature is not particularly useful with Isochronous endpoints because the packets transferred are often not maximum packet size and the PERI\_RXCSR register needs to be accessed following every packet to check for Overrun or CRC errors. When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

#### **16.3.8.1.4.2.1 Isochronous OUT Setup: Peripheral Mode**

In configuring an Rx endpoint for Isochronous OUT transactions, the RXMAXP register must be written with the maximum packet size (in bytes) for the endpoint. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint. In addition, the relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint) and the PERI\_RXCSR register should be set as shown in [Table 16-7](#).

**Table 16-7. PERI\_RXCSR Register Bit Configuration for Isochronous OUT Transactions**

| Bit Field | Bit Name  | Description   |
|-----------|-----------|---|
| Bit 15    | AUTOCLEAR | Cleared to 0 if using DMA. In CPU Mode of usage, if the CPU sets AUTOCLEAR bit, the RXPKTRDY bit will be automatically cleared when a packet of RXMAXP bytes has been unloaded from the Receive FIFO. |
| Bit 14    | ISO       | Set to 1 to configure endpoint usage for Isochronous transfer.  |
| Bit 13    | DMAEN     | Set to 1 if a DMA request is required for this Rx endpoint.   |
| Bit 12    | DISNYET   | Ignored in Isochronous Mode.  |
| Bit 11    | DMAMODE   | Always clear this bit to 0.   |

#### 16.3.8.1.4.2.2 Isochronous OUT Operation: Peripheral Mode

An Isochronous endpoint does not support data retries so, if a data overrun is to be avoided, there must be space in the FIFO to accept a packet when it is received. The host will send one packet per frame (or microframe in High-speed mode); however, the time within the frame can vary. If a packet is received near the end of one frame(/microframe) and another arrives at the start of the next frame, there will be little time to unload the FIFO. For this reason, double buffering of the endpoint is usually necessary.

An interrupt is generated whenever a packet is received from the host and the software may use this interrupt to unload the packet from the FIFO and clear the RXPKTRDY bit in the PERI\_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), depending on when the host has scheduled the transaction, the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame(/microframe) before unloading the FIFO. This can be done by using either the SOF interrupt or the external SOF\_PULSE signal from the controller to trigger the unloading of the data packet. The SOF\_PULSE is generated once per frame(/microframe) when a SOF packet is received. (The controller also maintains an external frame(/microframe) counter so it can still generate a SOF\_PULSE when the SOF packet has been lost.) The interrupts may still be used to clear the RXPKTRDY bit in PERI\_RXCSR and to check for data overruns/underruns.

#### 16.3.8.1.4.2.3 Isochronous OUT Error Handling: Peripheral Mode

If there is no space in the FIFO to store a packet when it is received from the host, the OVERRUN bit in the PERI\_RXCSR register (bit 2) will be set. This is an indication that the software is not unloading data fast enough for the host. It is up to the application to determine how this error condition is handled.

If the controller finds that a received packet has a CRC error, it will still store the packet in the FIFO and set the RXPKTRDY bit (bit 0 of PERI\_RXCSR) and the DATAERROR bit (bit 3 of PERI\_RXCSR). It is left up to the application to determine how this error condition is handled.

The number of USB packets sent in any microframe will depend on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated number of packets have not been received by the end of a microframe, the INCOMPRX bit (bit 8) bit in the PERI\_RXCSR register will be set to indicate that the data in the FIFO is incomplete. Equally, if a packet of the wrong data type is received, then the PID Error bit is the PERI\_RXCSR register will be set. In each case, an interrupt will, however, still be generated to allow the data that has been received to be read from the FIFO.

**Note:** The circumstances in which a PID Error or INCOMPRX is reported depends on the precise sequence of packets received.

When the core is operating in peripheral mode, the details are in [Table 16-8](#).

**Table 16-8. Isochronous OUT Error Handling: Peripheral Mode**

| No. Packet(s) Expected | Data Packet(s) Received | Response                     |
|------------------------|-------------------------|------------------------------|
| 1                      | DATA0                   | OK                           |
|                        | DATA1                   | PID Error set                |
|                        | DATA2                   | PID Error set                |
|                        | MDATA                   | PID Error set                |
| 2                      | DATA0                   | OK                           |
|                        | DATA1                   | INCOMPRX Set                 |
|                        | DATA2                   | INCOMPRX Set + PID Error set |
|                        | MDATA                   | INCOMPRX Set                 |
|                        | MDATA DATA0             | PID Error Set                |
| 3                      | MDATA DATA1             | OK                           |
|                        | MDATA DATA2             | PID Error Set                |
|                        | MDATA MDATA             | PID Error Set                |
|                        | DATA0                   | OK                           |
|                        | DATA1                   | INCOMPRX Set                 |

**Table 16-8. Isochronous OUT Error Handling: Peripheral Mode (continued)**

| No. Packet(s) Expected | Data Packet(s) Received | Response      |
|------------------------|-------------------------|---------------|
|                        | DATA2                   | INCOMPRX Set  |
|                        | MDATA                   | INCOMPRX Set  |
|                        | MDATA DATA0             | PID Error set |
|                        | MDATA DATA1             | OK            |
|                        | MDATA DATA2             | INCOMPRX Set  |
|                        | MDATA MDATA             | INCOMPRX Set  |
|                        | MDATA MDATA DATA0       | PID Error set |
|                        | MDATA MDATA DATA1       | PID Error set |
|                        | MDATA MDATA DATA2       | OK            |
|                        | MDATA MDATA MDATA       | PID Error set |

### 16.3.8.2 USB Controller Host Mode Operation

The USB controller assumes the role of a host when the USBx\_ID pin state is grounded or USB Mode Register[iddig=bit8] is cleared to 0 (provided that iddig\_mux, which is bit7 of USBnMODE is also set to 1) by the user application prior to the controller goes into session. When the USB controller go into session, application/firmware sets the DEVCTL[SESSION] bit to 1, it will assume the role of a host.

- Entry into Suspend mode. When operating as a host, the USB controller can be prompted to go into Suspend mode by setting the SUSPENDM bit in the POWER register. When this bit is set, the controller will complete the current transaction then stop the transaction scheduler and frame counter. No further transactions will be started and no SOF packets will be generated. If the ENSUSPM bit (bit 0 of POWER register) is set, the UTMI+ PHY will go into low-power mode when the controller goes into suspend mode.
- Sending Resume Signaling. When the application requires the controller to leave suspend mode, it needs to clear the SUSPENDM bit in the POWER register, set the RESUME bit and leave it set for 20ms. While the RESUME bit is high, the controller will generate Resume signaling on the bus. After 20 ms, the CPU should clear the RESUME bit, at which point the frame counter and transaction scheduler will be started.
- Responding to Remote Wake-up. If Resume signaling is detected from the target while the controller is in suspend mode, the UTMI+ PHY will be brought out of low-power mode and UTMI clock is restarted. The controller will then exit suspend mode and automatically set the RESUME bit in the POWER register (bit 2) to '1' to take over generating the Resume signaling from the target. If the resume interrupt is enabled, an interrupt will be generated.
- Reset Signaling. If the RESET bit in the POWER register (bit 3) is set while the controller is in Host mode, it will generate Reset signaling on the bus. If the HSENAB bit in the POWER register (bit 5) was set, it will also try to negotiate for high-speed operation. The software should keep the RESET bit set for at least 20 ms to ensure correct resetting of the target device. After the software has cleared the bit, the controller will start its frame counter and transaction scheduler. Whether high-speed operation is selected will be indicated by HSMODE bit of POWER register (bit 4).

### 16.3.8.2.1 Control Transactions: Host Mode

Host control transactions are conducted through Endpoint 0 and the software is required to handle all the Standard Device Requests that may be sent or received via Endpoint 0 (as described in Universal Serial Bus Specification, Revision 2.0). Endpoint 0 can only be serviced via CPU and DMA mode can not be used.

There are three categories of Standard Device Requests to be handled: Zero Data Requests (in which all the information is included in the command); Write Requests (in which the command will be followed by additional data); and Read Requests (in which the device is required to send data back to the host).

1. Zero Data Requests comprise a SETUP command followed by an IN Status Phase.
2. Write Requests comprise a SETUP command, followed by an OUT Data Phase which is in turn followed by an IN Status Phase.
3. Read Requests comprise a SETUP command, followed by an IN Data Phase which is in turn followed by an OUT Status Phase.

A timeout may be set to limit the length of time for which the host controller will retry a transaction which is continually NAKed by the target. This limit can be between 2 and 215 frames/microframes and is set through the NAKLIMIT0 register.

The following sections describe the actions that the CPU needs to take in issuing these different types of request through looking at the steps to take in the different phases of a control transaction.

**Note:** Before initiating any transactions as a Host, the FADDR register needs to be set to address the peripheral device. When the device is first connected, FADDR should be set to zero. After a SET\_ADDRESS command is issued, FADDR should be set the target's new address.

#### 16.3.8.2.1.1 Setup Phase of Control Transaction: Host Mode

For the SETUP Phase of a control transaction (Figure 16-8), the software driving the USB host device needs to:

1. Load the 8 bytes of the required Device request command into the Endpoint 0 FIFO
2. Set SETUPPKT and TXPKTRDY (bits 3 and 1 of HOST\_CSR0, respectively).

**NOTE:** These bits must be set together.

The controller then proceeds to send a SETUP token followed by the 8-byte command/request to Endpoint 0 of the addressed device, retrying as necessary. *Note:* On errors, the controller retries the transaction three times.

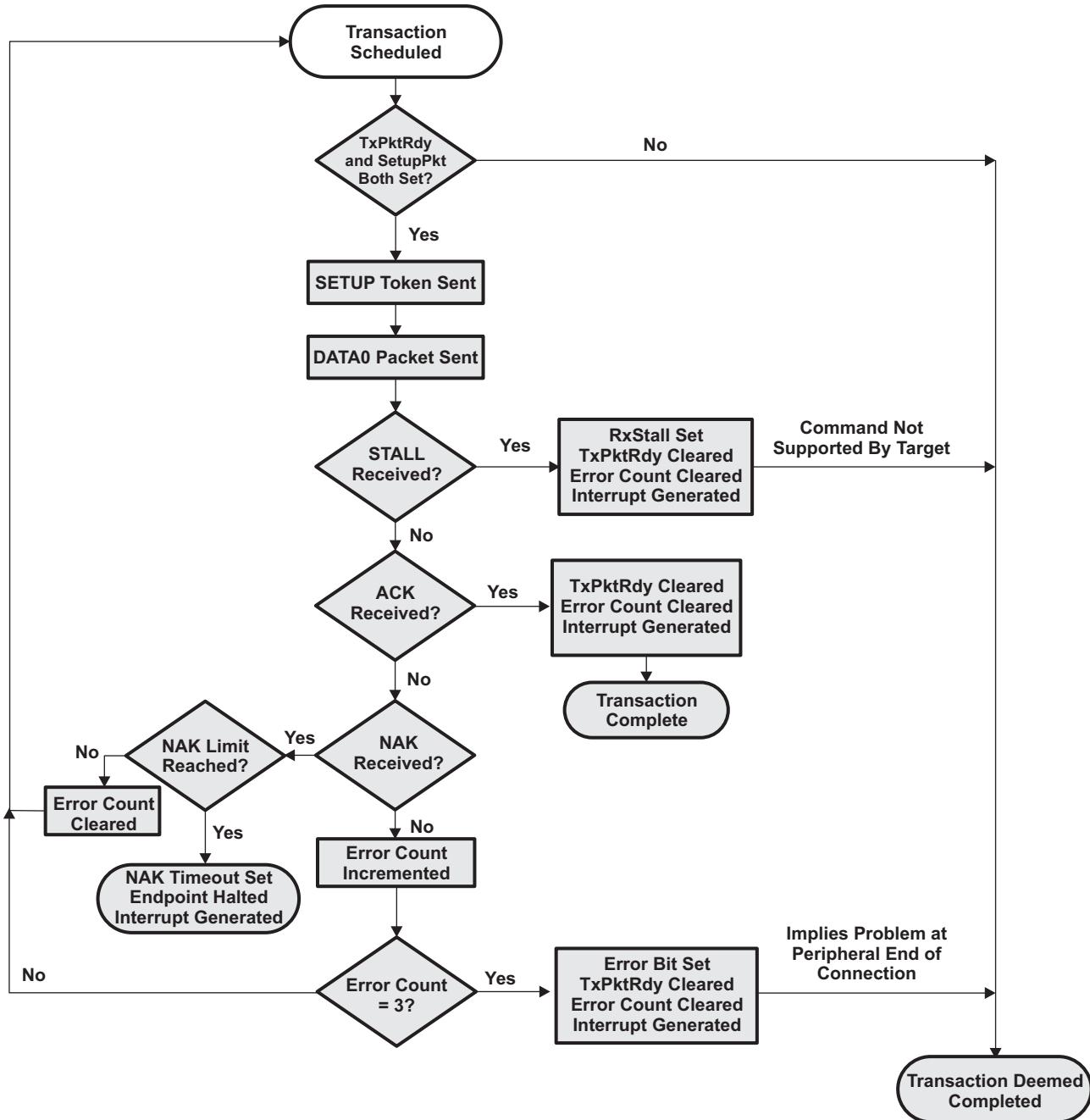
3. At the end of the attempt to send the 8-byte request data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST\_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK\_TIMEOUT bit (bit 7) has been set.

If RXSTALL is set, it indicates that the target did not accept the command (for example, because it is not supported by the target device) and so has issued a STALL response. If ERROR is set, it means that the controller has tried to send the SETUP Packet and the following data packet three times without getting any response.

If NAK\_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the SETUP packet, for longer than the time set in HOST\_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK\_TIMEOUT bit.

4. If none of RXSTALL, ERROR or NAK\_TIMEOUT is set, the SETUP Phase has been correctly ACKed and the software should proceed to the following IN Data Phase, OUT Data Phase or IN Status Phase specified for the particular Standard Device Request.

**Figure 16-8. Flow Chart of Setup Stage of a Control Transfer in Host Mode**



#### 16.3.8.2.1.2 Data Phase (IN Data Phase) of a Control Transaction: Host Mode

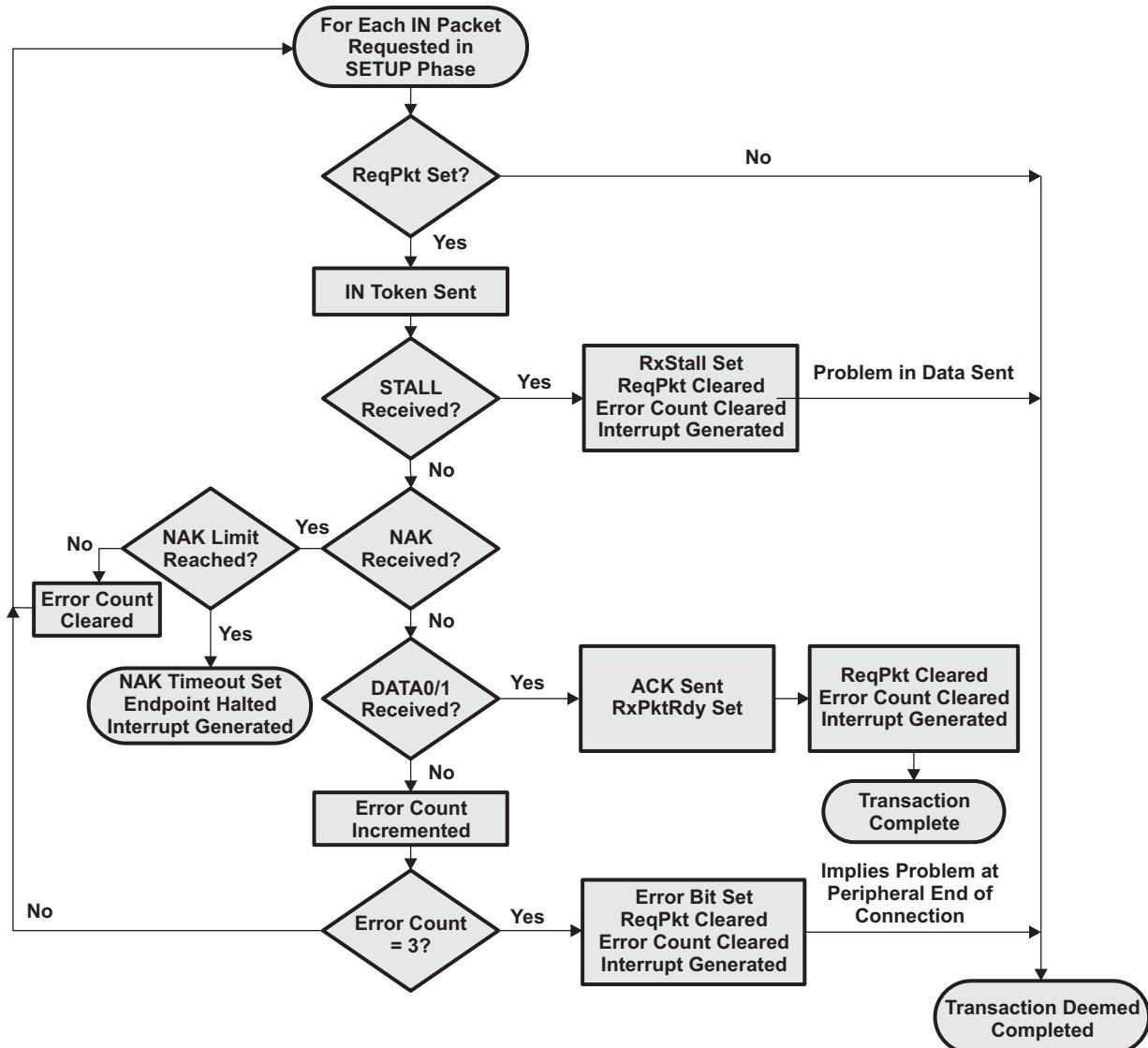
For the IN Data Phase of a control transaction (Figure 16-9), the software driving the USB host device needs to

1. Set REQPKT bit of HOST\_CSR0 (bit 5)
  2. Wait while the controller sends the IN token and receives the required data back.
  3. When the controller generates the Endpoint 0 interrupt, read HOST\_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK\_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.
- If RXSTALL is set, it indicates that the target has issued a STALL response.

- If ERROR is set, it means that the controller has tried to send the required IN token three times without getting any response. If NAK\_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in HOST\_NAKLIMIT0. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by clearing REQPKT before clearing the NAK\_TIMEOUT bit
4. If RXPKTRDY has been set, the software should read the data from the Endpoint 0 FIFO, then clear RXPKTRDY.
  5. If further data is expected, the software should repeat Steps 1-4.

When all the data has been successfully received, the CPU should proceed to the OUT Status Phase of the Control Transaction.

**Figure 16-9. Flow Chart of Data Stage (IN Data Phase) of a Control Transfer in Host Mode**

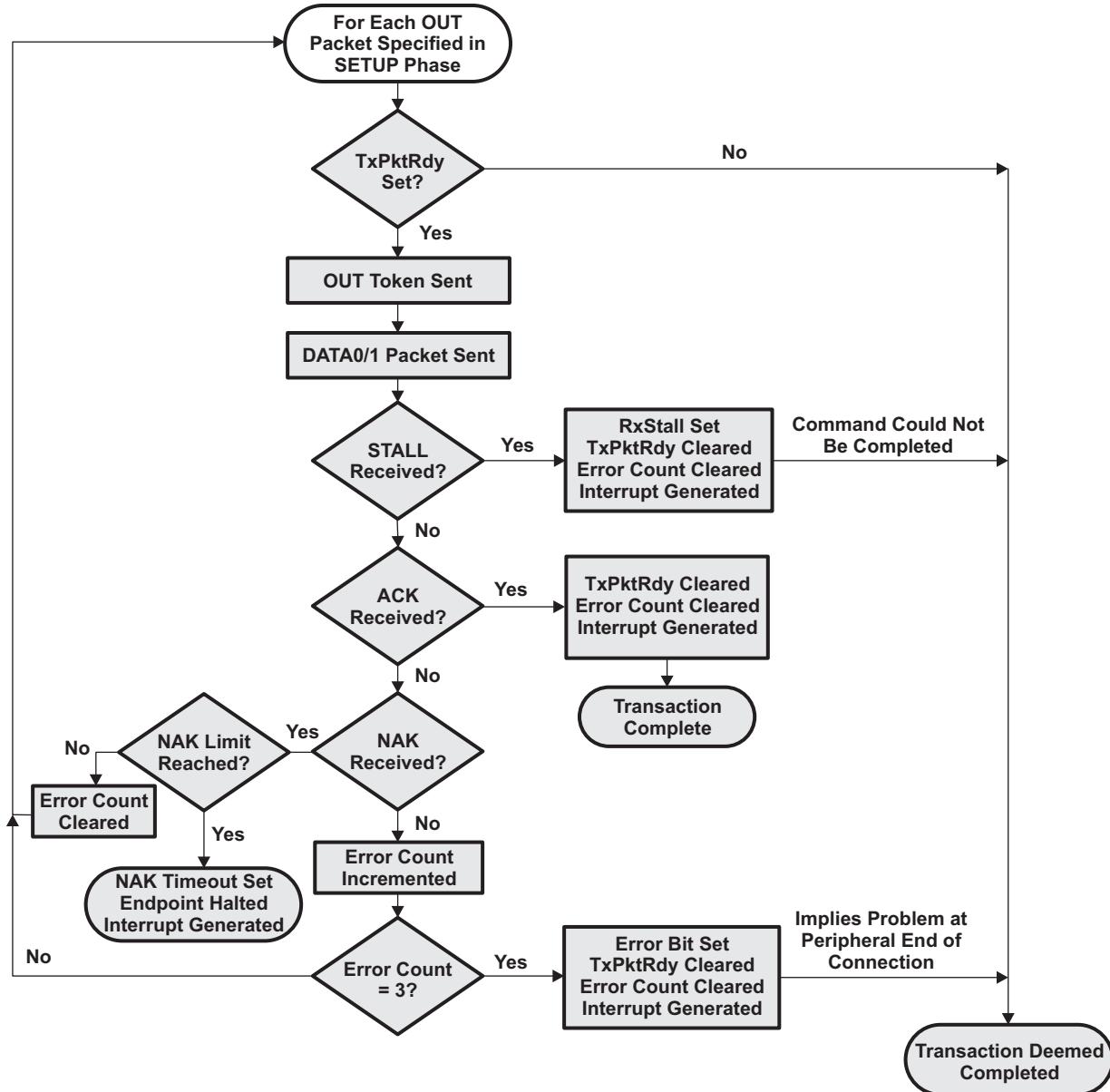


#### 16.3.8.2.1.3 Data Phase (OUT Data Phase) of a Control Transaction: Host Mode

For the OUT Data Phase of a control transaction (Figure 16-10), the software driving the USB host device needs to:

1. Load the data to be sent into the endpoint 0 FIFO.
2. Set the TXPKTRDY bit of HOST\_CSR0 (bit 1). The controller then proceeds to send an OUT token followed by the data from the FIFO to Endpoint 0 of the addressed device, retrying as necessary.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST\_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK\_TIMEOUT bit (bit 7) has been set.  
If RXSTALL bit is set, it indicates that the target has issued a STALL response.  
If ERROR bit is set, it means that the controller has tried to send the OUT token and the following data packet three times without getting any response. If NAK\_TIMEOUT is set, it means that the controller has received a NAK response to each attempt to send the OUT token, for longer than the time set in the HOST\_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK\_TIMEOUT bit.  
If none of RXSTALL, ERROR or NAKLIMIT is set, the OUT data has been correctly ACKed.
4. If further data needs to be sent, the software should repeat Steps 1-3.  
When all the data has been successfully sent, the software should proceed to the IN Status Phase of the Control Transaction.

**Figure 16-10. Flow Chart of Data Stage (OUT Data Phase) of a Control Transfer in Host Mode**



#### 16.3.8.2.1.4 Status Phase (Status for OUT Data Phase or Setup Phase) of a Control Transaction: Host Mode

IN Status Phase of a control transfer exists for a Zero Data Request or for a Write Request of a control transfer. The IN Status Phase follows the Setup Stage, if no Data Stage of a control transfer exists, or OUT Data Phase of a Data Stage of a control transfer.

For the IN Status Phase of a control transaction (Figure 16-11), the software driving the USB Host device needs to:

1. Set the STATUSPKT and REQPKT bits of HOST\_CSR0 (bit 6 and bit 5, respectively).
  2. Wait while the controller sends an IN token and receives a response from the USB peripheral device.
  3. When the controller generates the Endpoint 0 interrupt, read HOST\_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4), the NAK\_TIMEOUT bit (bit 7) or RXPKTRDY bit (bit 0) has been set.
- If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a

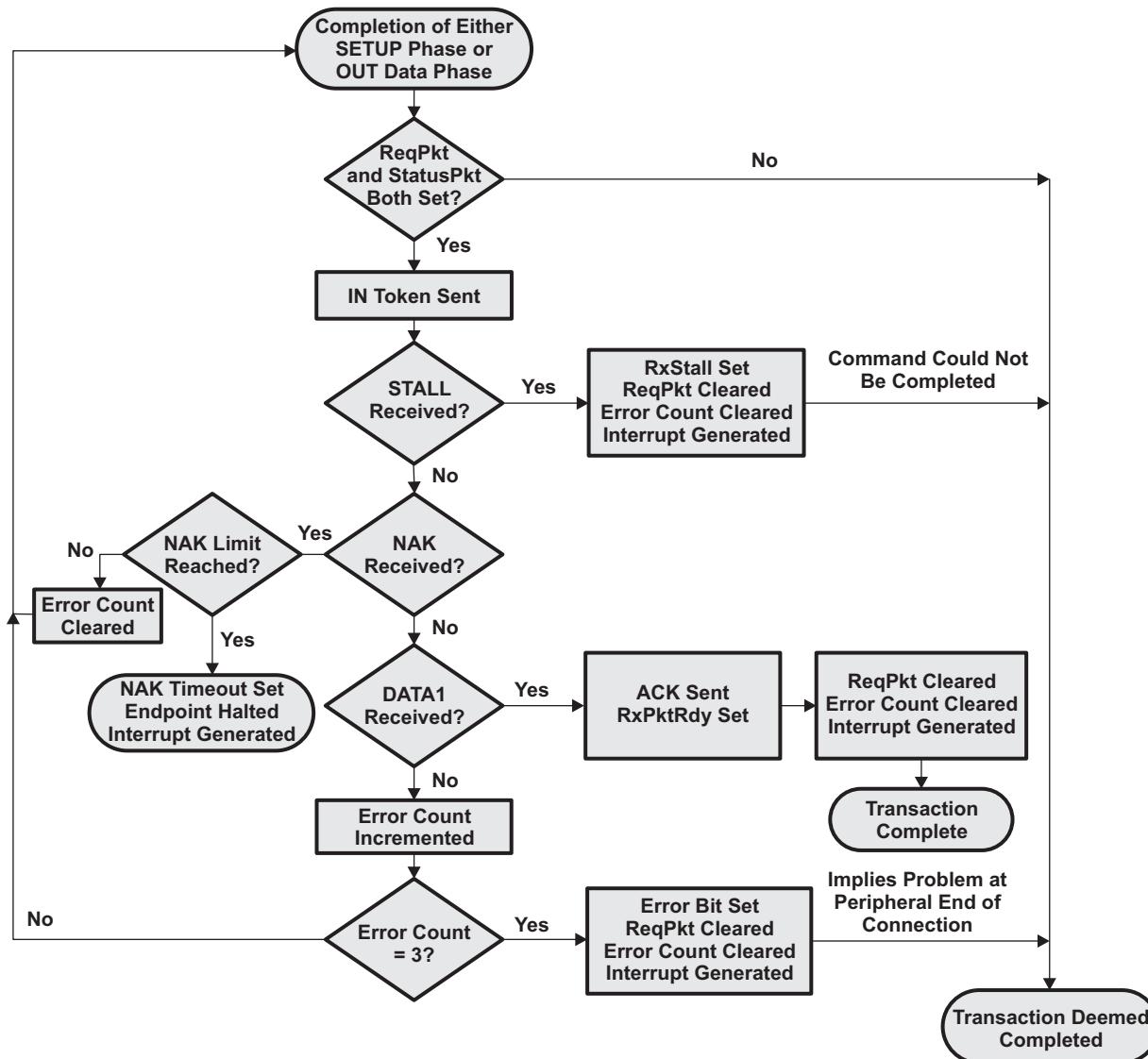
STALL response.

If ERROR bit is set, it means that the controller has tried to send the required IN token three times without getting any response.

If NAK\_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST\_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by clearing REQPKT bit and STATUSPKT bit before clearing the NAK\_TIMEOUT bit.

4. The CPU should clear the STATUSPPKT bit of HOST\_CSR0 together with (i.e., in the same write operation as) RxPktRdy if this has been set.

**Figure 16-11. Flow Chart of Status Stage of Zero Data Request or Write Request of a Control Transfer in Host Mode**



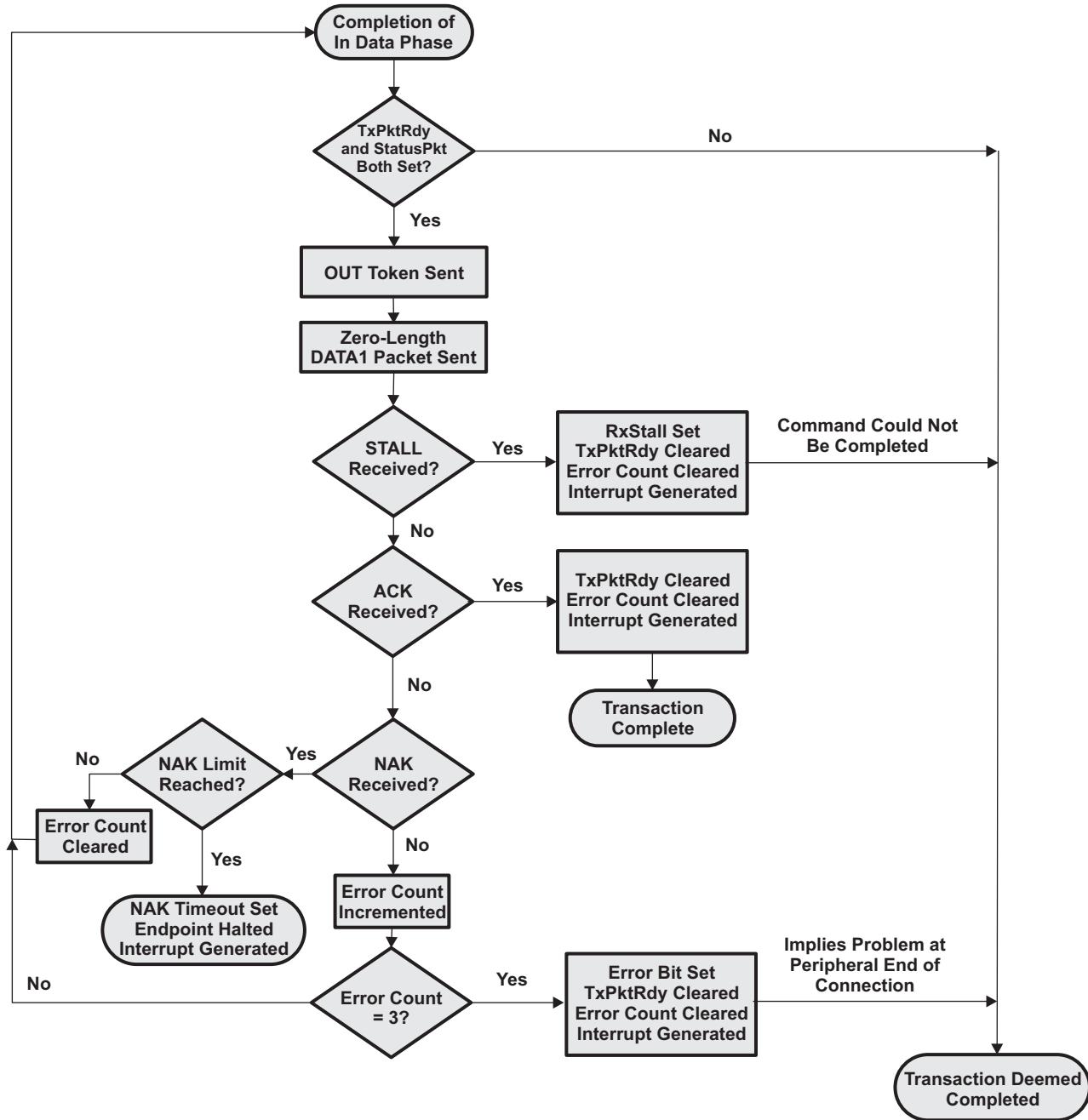
#### 16.3.8.2.1.5 Status Phase for a Read Request of a Control Transaction: Host Mode

OUT Status Phase of a control transfer exist for a Read Request or a control transfer where data was received by the host controller. The OUT Status phase follows the IN Data Stage of a control transfer.

For the OUT Status Phase of a control transaction (Figure 16-12), the CPU driving the host device needs to:

1. Set STATUSPKT and TXPKTRDY bits of HOST\_CSR0 (bit 6 and bit 1, respectively).  
NOTE: These bits need to be set together
2. Wait while the controller sends the OUT token and a zero-length DATA1 packet.
3. At the end of the attempt to send the data, the controller will generate an Endpoint 0 interrupt. The software should then read HOST\_CSR0 to establish whether the RXSTALL bit (bit 2), the ERROR bit (bit 4) or the NAK\_TIMEOUT bit (bit 7) has been set.  
If RXSTALL bit is set, it indicates that the target could not complete the command and so has issued a STALL response.  
If ERROR bit is set, it means that the controller has tried to send the STATUS Packet and the following data packet three times without getting any response.  
If NAK\_TIMEOUT bit is set, it means that the controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the HOST\_NAKLIMIT0 register. The controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK\_TIMEOUT bit.
4. If none of RXSTALL, ERROR or NAK\_TIMEOUT bits is set, the STATUS Phase has been correctly ACKed.

**Figure 16-12. Chart of Status Stage of a Read Request of a Control Transfer in Host Mode**



### 16.3.8.2.2 Bulk Transfer: Host Mode

Bulk transactions are handled by endpoints other than endpoint 0. It is used to handle non-periodic, large bursty communication typically used for a transfer that use any available bandwidth and can also be delayed until bandwidth is available.

#### 16.3.8.2.2.1 Bulk IN Transactions: Host Mode

A Bulk IN transaction may be used to transfer non-periodic data from the external USB peripheral to the host.

The following optional features are available for use with an Rx endpoint used in host mode to receive the data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention.

When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions. For more information see section on CPPI DMA

#### **16.3.8.2.2.1.1 Bulk IN Setup: Host Mode**

Before initiating any Bulk IN Transactions in Host mode:

- The HOST\_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 10 (binary value) in the PROT field for bulk transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST\_RXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or set to zero if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).

**Note:** Whether interrupt is handled from PDR level or Core level, interrupt is required to be enabled at the core level. See details on Core Interrupt registers, CTRLR register, and sections on Interrupt Handling

- The AUTOREQ bit field should be used only when servicing transfers using CPU mode and must be cleared when using DMA Mode. For DMA Mode, dedicated registers USB0/1 Req Registers exist and the HOST\_RXCSR[AUTOREQ] should be cleared.

When DMA is enabled, the following bits of HOST\_RXCSR register should be set as:

- Clear AUTOCLEAR.
- Set DMAEN (bit 13) to 1 if a DMA request is required for this endpoint.
- Clear DSINYET (bit 12) to 0 to allow normal PING flow control. This will affect only High Speed transactions.
- Clear DMAMODE (bit 11) to 0.

**Note:** If DMA is enabled, the USB0/1 Auto Req register can be set for generating IN tokens automatically after receiving the data. Set the bit field RXn\_AUTOREQ (where n is the endpoint number) with binary value 01 or 11.

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN and DATATOG bits of HOST\_RXCSR (bit 10 and bit 9) to toggle the current setting or by setting the CLRDATATOG bit of HOST\_RXCSR (bit 7). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also if there are any data packets in the FIFO (indicated by the RXPKTRDY bit (bit 0 of HOST\_RXCSR) being set), they should be flushed by setting the FLUSHFIFO bit of HOST\_RXCSR (bit 4).

**NOTE:** It may be necessary to set this bit twice in succession if double buffering is enabled.

#### 16.3.8.2.2.1.2 Bulk IN Operation: Host Mode

When Bulk data is required from the USB peripheral device, the software should set the REQPKT bit in the corresponding HOST\_RXCSR register (bit 5). The controller will then send an IN token to the selected peripheral endpoint and waits for data to be returned.

If data is correctly received, RXPKTRDY bit of HOST\_RXCSR (bit 0) is set. If the USB peripheral device responds with a STALL, RXSTALL bit (bit 6 of HOST\_RXCSR) is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the POLINTVL\_NAKLIMIT set in the HOST\_RXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting the ERROR bit of HOST\_RXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST\_RXCSR register to determine whether the RXPKTRDY, RXSTALL, ERROR or DATAERR\_NAKTIMEOUT bit is set and act accordingly. If the DATAERR\_NAKTIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the DATAERR\_NAKTIMEOUT bit or to abort the transaction by clearing REQPKT bit before clearing the DATAERR\_NAKTIMEOUT bit.

The packets received should not exceed the size specified in the RXMAXP register (as this should be the value set in the wMaxPacketSize field of the endpoint descriptor sent to the host).

In the general case, the application software (if CPU is servicing the endpoint) will need to read each packet from the FIFO individually. If large blocks of data are being transferred, the overhead of calling an interrupt service routine to unload each packet can be avoided by using DMA.

**Note:** When using DMA, see [Section 16.3.9, Communications Port Programming Interface \(CPPI\) 4.1 DMA](#), for the proper configuration of the core register, HOST\_RXCSR.

#### 16.3.8.2.2.1.3 Bulk IN Error Handling: Host Mode

If the target wants to shut down the Bulk IN pipe, it will send a STALL response to the IN token. This will result in the RXSTALL bit of HOST\_RXCSR (bit 6) being set.

#### 16.3.8.2.2.2 Bulk OUT Transactions: Host Mode

A Bulk OUT transaction may be used to transfer non-periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention. For more information on using DMA, consult the section discussing CPPI DMA.

When DMA is enabled and DMAMODE bit in HOST\_TXCSR register is set, an endpoint interrupt will not be generated for completion of packet reception. An endpoint interrupt will be generated only in the error conditions.

### 16.3.8.2.2.2.1 Bulk OUT Setup: Host Mode

Before initiating any bulk OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint. (TXFUNCADDR register is available for all endpoints from EP0 to EP15.)
- The HOST\_TXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set PROT field to 10b for bulk transfer.
  - Enter Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
- The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST\_TXINTERVAL register needs to be written with the required value for the NAK limit (2-215 frames/microframes), or set to zero if the NAK timeout feature is not required.
- The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST\_TXCSR register should be set as:
  - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
  - Clear the FRCDATATOG bit (bit 11) to 0 to allow normal data toggle operations.
  - Clear/Set AUTOSET bit (bit 15). The setting of this bit depends on the desire of the user/application in automatically setting the TXPKTRDY bit when servicing transactions using CPU.

**Note:** If DMA is needed to be used in place of the CPU, the following table displays the setting of the core register HOST\_TXCSR register in Host mode. For the CPPI DMA registers settings consult the section on CPPI DMA within this document.

| Bit Field | Bit Name   | Description  |
|-----------|------------|--|
| Bit 15    | AUTOSET    | Cleared to 0 if using DMA. For CPU Mode use, if AUTOSET bit is set, the TXPKTRDY bit will be automatically set when data of the maximum packet size is loaded into the FIFO. |
| Bit 14    | ISO        | Cleared to 0 for bulk mode operation.  |
| Bit 13    | MODE       | Set to 1 to make sure the FIFO is enabled (only necessary if the FIFO is shared with an RX endpoint)   |
| Bit 12    | DMAEN      | Set to 1 to enable DMA usage; not needed if CPU is being used to service the Tx Endpoint   |
| Bit 11    | FRCDATATOG | Cleared to 0 to allow normal data toggle operations.   |
| Bit 10    | DMAMODE    | Set to 1 when DMA is used to service Tx FIFO.  |

When the endpoint is first configured, the endpoint data toggle should be cleared to 0 either by using the DATATOGWREN bit and DATATOG bit of HOST\_TXCSR (bit 9 and bit 8) to toggle the current setting or by setting the CLRDATATOG bit of HOST\_TXCSR (bit 6). This will ensure that the data toggle (which is handled automatically by the controller) starts in the correct state. Also, if there are any data packets in the FIFO (indicated by the FIFONOTEMPTY bit of HOST\_TXCSR register (bit 1) being set), they should be flushed by setting the FLUSHFIFO bit (bit 3 of HOST\_TXCSR).

**NOTE:** It may be necessary to set this bit twice in succession if double buffering is enabled.

#### 16.3.8.2.2.2.2 Bulk OUT Operation: Host Mode

When Bulk data is required to be sent to the USB peripheral device, the software should write the first packet of the data to the FIFO (or two packets if double-buffered) and set the TXPKTRDY bit in the corresponding HOST\_TXCSR register (bit 0). The controller will then send an OUT token to the selected peripheral endpoint, followed by the first data packet from the FIFO.

If data is correctly received by the peripheral device, an ACK should be received whereupon the controller will clear TXPKTRDY bit of HOST\_TXCSR (bit 0). If the USB peripheral device responds with a STALL, the RXSTALL bit (bit 5) of HOST\_TXCSR is set. If a NAK is received, the controller tries again and continues to try until either the transaction is successful or the NAK limit set in the HOST\_RXINTERVAL register is reached. If no response at all is received, two further attempts are made before the controller reports an error by setting ERROR bit in HOST\_TXCSR (bit 2).

The controller then generates the appropriate endpoint interrupt, whereupon the software should read the corresponding HOST\_TXCSR register to determine whether the RXSTALL (bit 5), ERROR (bit 2) or NAK\_TIMEOUT (bit 7) bit is set and act accordingly. If the NAK\_TIMEOUT bit is set, the controller can be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT bit or to abort the transaction by flushing the FIFO before clearing the NAK\_TIMEOUT bit.

If large blocks of data are being transferred, then the overhead of calling an interrupt service routine to load each packet can be avoided by using DMA.

#### 16.3.8.2.2.2.3 Bulk OUT Error Handling: Host Mode

If the target wants to shut down the Bulk OUT pipe, it will send a STALL response. This is indicated by the RXSTALL bit of HOST\_TXCSR register (bit 5) being set.

#### 16.3.8.2.3 Interrupt Transfer: Host Mode

When the controller is operating as the host, interactions with an Interrupt endpoint on the USB peripheral device are handled in very much the same way as the equivalent Bulk transactions (described in previous sections).

The principal difference as far as operational steps are concerned is that the PROT field of HOST\_RXTYPE and HOST\_TXTYPE (bits 5-4) need to be set (binary value) to represent an Interrupt transaction. The required polling interval also needs to be set in the HOST\_RXINTERVAL and HOST\_TXINTERVAL registers.

#### 16.3.8.2.4 Isochronous Transfer: Host Mode

Isochronous transfers are used when working with isochronous data. Isochronous transfers provide periodic, continuous communication between host and device.

An Isochronous IN transaction is used to transfer periodic data from the USB peripheral to the host.

##### 16.3.8.2.4.1 Isochronous IN Transactions: Host Mode

The following optional features are available for use with an Rx endpoint used in Host mode to receive this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO on reception from the host. This allows that one packet can be received while another is being read. Double packet buffering is enabled by setting the DPB bit of RXFIFOSZ register (bit 4).
- AutoRequest: When the AutoRequest feature is enabled, the REQPKT bit of HOST\_RXCSR (bit 5) will be automatically set when the RXPKTRDY bit is cleared. This only applies when the CPU is being used to service the endpoint. When using DMA, this bit field needs to be cleared to Zero. CPPI DMA has its own configuration registers that renders a similar task, USB0/1 Auto Req registers, that needs to be used to have a similar effect. For more information, see [Section 16.3.9, Communications Port Programming Interface \(CPPI\) 4.1 DMA](#).
- DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow the DMA controller to unload packets from the FIFO without processor intervention. However, this feature is not particularly useful with isochronous

endpoints because the packets transferred are often not maximum packet size.

- When DMA is enabled, endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

#### **16.3.8.2.4.1.1 Isochronous IN Transfer Setup: Host**

Before initiating an Isochronous IN Transactions in Host mode:

- The target function address needs to be set in the RXFUNCADDR register for the selected controller endpoint (RXFUNCADDR register is available for all endpoints from EP0 to EP4).
- The HOST\_RXTYPE register for the endpoint that is to be used needs to be programmed as:
  - Operating speed in the SPEED bit field (bits 7 and 6).
  - Set 01 (binary value) in the PROT field for isochronous transfer.
  - Endpoint Number of the target device in RENDPN field. This is the endpoint number contained in the Rx endpoint descriptor returned by the target device during enumeration.
- The RXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
- The HOST\_RXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
- The relevant interrupt enable bit in the INTRRXE register should be set (if an interrupt is required for this endpoint).
- The following bits of HOST\_RXCSR register should be set as:
  - Clear AUTOCLEAR
  - Set the DMAEN bit (bit 13) to 1 if a DMA request is required for this endpoint.
  - Clear the DISNYET it (bit 12) to 0 to allow normal PING flow control. This will only affect High Speed transactions.
  - Clear DMAMODE bit (bit 11) to 0.
- If DMA is enabled, AUTOREQ register can be set for generating IN tokens automatically after receiving the data. Set the bit field RXn\_AUTOREQ (where n is the endpoint number) with binary value 01 or 11. For detailed information on using CPPI DMA, consult related section within this document.

#### **16.3.8.2.4.1.2 Isochronous IN Operation: Host Mode**

The operation starts with the software setting REQPKT bit of HOST\_RXCSR (bit 5). This causes the controller to send an IN token to the target.

When a packet is received, an interrupt is generated which the software may use to unload the packet from the FIFO and clear the RXPKTRDY bit in the HOST\_RXCSR register (bit 0) in the same way as for a Bulk Rx endpoint. As the interrupt could occur almost any time within a frame(/microframe), the timing of FIFO unload requests will probably be irregular. If the data sink for the endpoint is going to some external hardware, it may be better to minimize the requirement for additional buffering by waiting until the end of each frame before unloading the FIFO. This can be done by using the SOF\_PULSE signal from the controller to trigger the unloading of the data packet. The SOF\_PULSE is generated once per frame(/microframe). The interrupts may still be used to clear the RXPKTRDY bit in HOST\_RXCSR.

#### **16.3.8.2.4.1.3 Isochronous IN Error Handling: Host Mode**

If a CRC or bit-stuff error occurs during the reception of a packet, the packet will still be stored in the FIFO but the DATAERR\_NAKTIMEOUT bit of HOST\_RXCSR (bit 3) is set to indicate that the data may be corrupt.

**Note:** The number of USB packets sent in any microframe will depend on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated numbers of packets have not been received by the end of a microframe, the INCOMPRX bit in the HOST\_RXCSR register will be set to indicate that the data in the FIFO is incomplete. Equally, if a packet of the wrong data type is received, then the PID Error bit is the HOST\_RXCSR register will be set. In each case, an interrupt will, however, still be generated to allow the data that has been received to be read from the FIFO.

#### 16.3.8.2.4.2 Isochronous OUT Transactions: Host Mode

An Isochronous OUT transaction may be used to transfer periodic data from the host to the USB peripheral.

Following optional features are available for use with a Tx endpoint used in Host mode to transmit this data:

- Double packet buffering: When enabled, up to two packets can be stored in the FIFO awaiting transmission to the peripheral device. Double packet buffering is enabled by setting the DPB bit of TXFIFOSZ register (bit 4).

DMA: If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature can be used to allow the DMA controller to load packets into the FIFO without processor intervention.

However, this feature is not particularly useful with isochronous endpoints because the packets transferred are often not maximum packet size.

When DMA is enabled and endpoint interrupt will not be generated for completion of packet reception. Endpoint interrupt will be generated only in the error conditions.

#### 16.3.8.2.4.2.1 Isochronous OUT Transfer Setup: Host Mode

Before initiating any Isochronous OUT transactions:

- The target function address needs to be set in the TXFUNCADDR register for the selected controller endpoint (TXFUNCADDR register is available for all endpoints from EP0 to EP4).
  - The HOST\_TXTYPE register for the endpoint that is to be used needs to be programmed as:
    - Operating speed in the SPEED bit field (bits 7 and 6).
    - Set 01 (binary value) in the PROT field for isochronous transfer.
    - Endpoint Number of the target device in TENDPN field. This is the endpoint number contained in the OUT(Tx) endpoint descriptor returned by the target device during enumeration.
  - The TXMAXP register for the controller endpoint must be written with the maximum packet size (in bytes) for the transfer. This value should be the same as the wMaxPacketSize field of the Standard Endpoint Descriptor for the target endpoint.
  - The HOST\_TXINTERVAL register needs to be written with the required transaction interval (usually one transaction per frame/microframe).
  - The relevant interrupt enable bit in the INTRTXE register should be set (if an interrupt is required for this endpoint).
  - The following bits of HOST\_TXCSR register should be set as:
    - Set the MODE bit (bit 13) to 1 to ensure the FIFO is enabled (only necessary if the FIFO is shared with an Rx endpoint).
    - Set the DMAEN bit (bit 12) to 1 if a DMA request is required for this endpoint
    - The FRCDATATOG bit (bit 11) is ignored for isochronous transactions.
    - Set the DMAMODE bit (bit 10) to 1 when DMA is enabled.
- For more details in using DMA, consult CPPI DMA section within this document.

#### 16.3.8.2.4.2.2 Isochronous OUT Transfer Operation: Host Mode

The operation starts when the software writes to the FIFO and sets TXPKTRDY bit of HOST\_TXCSR (bit 0). This triggers the controller to send an OUT token followed by the first data packet from the FIFO.

An interrupt is generated whenever a packet is sent and the software may use this interrupt to load the next packet into the FIFO and set the TXPKTRDY bit in the HOST\_TXCSR register (bit 0) in the same way as for a Bulk Tx endpoint. As the interrupt could occur almost any time within a frame, depending on when the host has scheduled the transaction, this may result in irregular timing of FIFO load requests. If the data source for the endpoint is coming from some external hardware, it may be more convenient to wait until the end of each frame before loading the FIFO as this will minimize the requirement for additional buffering. This can be done by using the SOF\_PULSE signal from the controller to trigger the loading of the next data packet. The SOF\_PULSE is generated once per frame(/microframe). The interrupts may still be used to set the TXPKTRDY bit in HOST\_TXCSR.

### 16.3.9 Communications Port Programming Interface (CPPI) 4.1 DMA

The CPPI DMA module supports the transmission and reception of USB packets. The CPPI DMA is designed to facilitate the segmentation and reassembly of CPPI compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each networking port. Multiple Tx and Rx channels are provided for all endpoints (excluding endpoint 0) within the DMA allowing multiple segmentation or reassembly operations to be effectively performed in parallel (but not actually simultaneously). The DMA controller maintains state information for each of the ports/channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

The CPPI (version 4.1) DMA controller sub-module is a common 15 port DMA controller. It supports 15 Tx and 15 Rx channels and each port attaches to the associated endpoint in the controller. Port 1 maps to endpoint 1 and Port 2 maps to endpoint 2 and so on with Port 15 mapped to endpoint 15; endpoint 0 can not utilize the DMA and the firmware is responsible to load or offload the endpoint 0 FIFO via CPU.

#### 16.3.9.1 CPPI Terminology

**Host** — The host is an intelligent system resource that configures and manages each communications control module. The host is responsible for allocating memory, initializing all data structures, and responding to port interrupts.

**Main Memory** — The area of data storage managed by the CPU. The CPPI DMA (CDMA) reads and writes CPPI packets from and to main memory. This memory can exist internal or external from the device.

**Queue Manager (QM)** — The QM is responsible for accelerating management of a variety of Packet Queues and Free Descriptor / Buffer Queues. It provides status indications to the CDMA Scheduler when queues are empty or full.

**CPPI DMA (CDMA)** — The CDMA is responsible for transferring data between the CPPI FIFO and Main Memory. It acquires free Buffer Descriptor from the QM (Receive Submit Queue) for storage of received data, posts received packets pointers to the Receive Completion Queue, transmits packets stored on the Transmit Submit Queue (Transmit Queue), and posts completed transmit packets to the Transmit Completion Queue.

**CDMA Scheduler (CDMAS)** — The CDMAS is responsible for scheduling CDMA transmit and receive operations. It uses Queue Indicators from the QM and the CDMA to determine the types of operations to schedule.

**CPPI FIFO** — The CPPI FIFO provides FIFO interfaces (for each of the 15 transmit and 15 receive endpoints).

**Transfer DMA (XDMA)** — The XDMA receives DMA requests from the Mentor USB 2.0 Core and initiates DMAs to the CPPI FIFO.

**Endpoint FIFOs** — The Endpoint FIFOs are the USB packet storage elements used by the Mentor USB 2.0 Core for packet transmission or reception. The XDMA transfers data between the CPPI FIFO and the Endpoint FIFOs for transmit operations and between the Endpoint FIFOs and the CPPI FIFO for receive operations.

**Port** — A port is the communications module (peripheral hardware) that contains the control logic for Direct Memory Access for a single transmit/receive interface or set of interfaces. Each port may have multiple communication channels that transfer data using homogenous or heterogeneous protocols. A port is usually subdivided into transmit and receive pairs which are independent of each other. Each endpoint, excluding endpoint 0, has its own dedicated port.

**Channel** — A channel refers to the sub-division of information (flows) that is transported across ports. Each channel has associated state information. Channels are used to segregate information flows based on the protocol used, scheduling requirements (example: CBR, VBR, ABR), or concurrency requirements (that is, blocking avoidance). All fifteen ports per USB Module have dedicated single channels, channel 0, associated for their use in a USB application.

**Data Buffer** — A data buffer is a single data structure that contains payload information for transmission to or reception from a port. A data buffer is a byte aligned contiguous block of memory used to store packet payload data. A data buffer may hold any portion of a packet and may be linked together (via descriptors) with other buffers to form packets. Data buffers may be allocated anywhere within the 32-bit memory space. The Buffer Length field of the packet descriptor indicates the number of valid data bytes in the buffer. There may be from 1 to 4M-1 valid data bytes in each buffer.

**Host Buffer Descriptor (Buffer Descriptor)** — A buffer descriptor is a single data structure that contains information about one or more data buffers. This type of descriptor is required when more than one descriptor is needed to define an entire packet, i.e., it either defines the middle of a packet or end of a packet.

**Start of Packet (SOP)** — Packet along side Buffer Descriptors are used to hold information about Packets. When multiple Descriptors are used to hold a single packet information or a single Descriptor is used to hold a single packet information, the first descriptor is referred to as a Packet Descriptor which is also Start-of-Packet Descriptor.

**Host Packet Descriptor (Packet Descriptor)** — A packet descriptor is another name for the first buffer descriptor within a packet. Some fields within a data buffer descriptor are only valid when it is a packet descriptor including the tags, packet length, packet type, and flags. This type of descriptor is always used to define a packet since it provides packet level information that is useful to both the ports and the Host in order to properly process the packet. It is the only descriptor used when single descriptor solely defines a packet. When multiple descriptors are needed to define a packet, the packet descriptor is the first descriptor used to define a packet.

**Free Descriptor/Buffer Queue** — A free descriptor/buffer queue is a hardware managed list of available descriptors with pre-linked empty buffers that are to be used by the receive ports for host type descriptors. Free Descriptor/Buffer Queues are implemented by the Queue Manager.

**Teardown Descriptor** — Teardown Descriptor is a special structure which is not used to describe either a packet or a buffer but is instead used to describe the completion of a channel halt and teardown event. Channel teardown is an important function because it ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the Host without the possibility of losing buffer or descriptor references (which results in a memory leak).

**Packet Queue** — A packet queue is hardware managed list of valid (i.e. populated) packet descriptors that is used for forwarding a packet from one entity to another for any number of purposes.

**NOTE:** All descriptors (regardless of type) must be allocated at addresses that are naturally aligned to the smallest power of 2 that is equal to or greater than the descriptor size.

### 16.3.9.2 Data Structures

Two data structures are mainly used to identify data buffers used by packet and buffer descriptors. A third Descriptor, Teardown Descriptor, exists. The purpose of this Descriptor is a channel halt and teardown event. Each of these Descriptor layouts as well as bit fields are shown below.

#### 16.3.9.2.1 Host Packet Descriptor/ Packet Descriptor (SOP Descriptor)

Descriptors are designed to be used when USB like application requires support for true, unlimited fragment count scatter/gather type operations. The Packet Descriptor is the first descriptor on multiple descriptors setup or the only descriptor in a single descriptors setup. The Packet Descriptor contains the following information:

- Indicator which identifies the descriptor as a packet descriptor (always 10h)
- Source and destination tags (reserved)
- Packet type
- Packet length
- Protocol specific region size
- Protocol specific control/statusbits
- Pointer to the first valid byte in the SOP data buffer

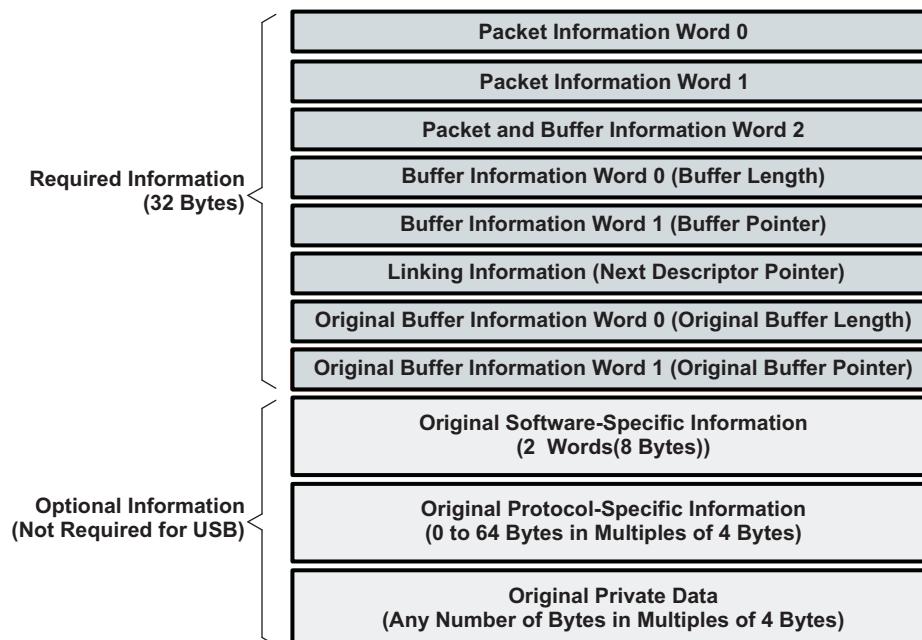
- Length of the SOP data buffer
- Pointer to the next buffer descriptor in the

Packet descriptors can vary in size by design of their defined fields from 32 bytes up to 104 bytes. Within this range, packet descriptors always contain 32 bytes of required information and may also contain 8 bytes of software specific tagging information and up to 64 bytes (indicated in 4 byte increments) of protocol specific information. How much protocol specific information (and therefore the allocated size of the descriptors) is application dependent. Port will make use of the first 32 bytes only.

From a general USB use perspective, a 32-byte descriptor size is suffix and the use of this size is expected for a normal USB usage.

The packet descriptor layout is shown in [Figure 16-13](#) and described within [Table 16-9](#) through [Table 16-16](#).

**Figure 16-13. Packet Descriptor Layout**



**Table 16-9. Packet Descriptor Word 0 (PD0) Bit Field Descriptions**

| Bits  | Field Name                         | Description   |
|-------|------------------------------------|---|
| 31-27 | Descriptor type                    | The host packet descriptor type is 16 decimal (10h). The CPU initializes this field.  |
| 26-22 | Protocol-specific valid word count | This field indicates the valid number of 32-bit words in the protocol-specific region. The CPU initializes this field. This is encoded in increments of 4 bytes as:<br>0 = 0 byte<br>1 = 4 bytes<br>...<br>16 = 64 bytes<br>17-31 = Reserved  |
| 21-0  | Packet length                      | The length of the packet in bytes. If the packet length is less than the sum of the buffer lengths, then the packet data will be truncated. A packet length greater than the sum of the buffers is an error. The valid range for the packet length is 0 to (4M - 1) bytes. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. |

**Table 16-10. Packet Descriptor Word 1 (PD1) Bit Field Descriptions**

| Bits  | Field Name                | Description   |
|-------|---------------------------|---|
| 31-27 | Source Tag; Port #        | This field indicates the port number (0-31) from which the packet originated. The DMA overwrites this field on packet reception. This is the RX Endpoint number from which the packet originated. |
| 26-21 | Source Tag; Channel #     | This field indicates the channel number within the port from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0-67.                           |
| 20-16 | Source Tag: Sub-channel # | This field indicates the sub-channel number (0-31) within the channel from which the packet originated. The DMA overwrites this field on packet reception. This field is always 0.                |
| 15-0  | Destination Tag           | This field is application-specific. This field is always 0.   |

**Table 16-11. Packet Descriptor Word 2 (PD2) Bit Field Descriptions**

| Bits  | Field Name                   | Description   |
|-------|------------------------------|---|
| 31    | Packet error                 | This bit indicates if an error occurred during reception of this packet (0 = no error occurred; 1 = error occurred). The DMA overwrites this field on packet reception. Additional information about different errors may be encoded in the protocol specific fields in the descriptor.   |
| 30-26 |                              | This field indicates the type of this packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. This field is encoded as:<br>5 = USB<br>8-31 = Reserved  |
| 25-20 | Reserved                     | Reserved  |
| 19    | Zero-length packet indicator | If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit is set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller. |
| 18-16 | Protocol-specific            | This field contains protocol-specific flags/information that can be assigned based on the packet type. Not used for USB.  |
| 15    | Return policy                | This field indicates the return policy for this packet. The CPU initializes this field.<br>0 = Entire packet (still linked together) should be returned to the queue specified in bits 11-0.<br>1 = Each buffer should be returned to the queue specified in bits 11-0 of Word 2 in their respective descriptors. The Tx DMA will return each buffer in sequence.   |
| 14    | On-chip                      | This field indicates whether or not this descriptor is in a region which is in on-chip memory space (1) or in external memory (0).  |
| 13-12 | Packet return queue mgr #    | This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only one queue manager in the USB HS/FS device controller. This field must always be 0.                                    |
| 11-0  | Packet return queue #        | This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.   |

**Table 16-12. Packet Descriptor Word 3 (PD3) Bit Field Descriptions**

| Bits  | Field Name      | Description   |
|-------|-----------------|---|
| 31-22 | Reserved        | Reserved  |
| 21-0  | Buffer 0 length | The buffer length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. |

**Table 16-13. Packet Descriptor Word 4 (PD4) Bit Field Descriptions**

| Bits | Field Name       | Description   |
|------|------------------|---|
| 31-0 | Buffer 0 pointer | The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. |

**Table 16-14. Packet Descriptor Word 5 (PD5) Bit Field Descriptions**

| Bits | Field Name              | Description  |
|------|-------------------------|--|
| 31-0 | Next descriptor pointer | The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current buffer is the last buffer in the packet. The CPU initializes this field for transmitted packets; the DMA overwrites this field on packet reception. |

**Table 16-15. Packet Descriptor Word 6 (PD6) Bit Field Descriptions**

| Bits  | Field Name               | Description   |
|-------|--------------------------|---|
| 31-22 | Reserved                 | Reserved  |
| 21-0  | Original buffer 0 length | The buffer length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information. |

**Table 16-16. Packet Descriptor Word 7 (PD7) Bit Field Descriptions**

| Bits  | Field Name                | Description  |
|-------|---------------------------|--|
| 31-22 | Reserved                  | Reserved   |
| 21-0  | Original buffer 0 pointer | The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information. |

#### 16.3.9.2.2 Host Buffer Descriptor/Buffer Descriptor (BD)

The buffer descriptor is identical in size and organization to a Packet Descriptor but does not include valid information in the packet level fields and does not include a populated region for protocol specific information. The packet level fields are not needed since the SOP descriptor contains this information and additional copy of this data is not needed/necessary.

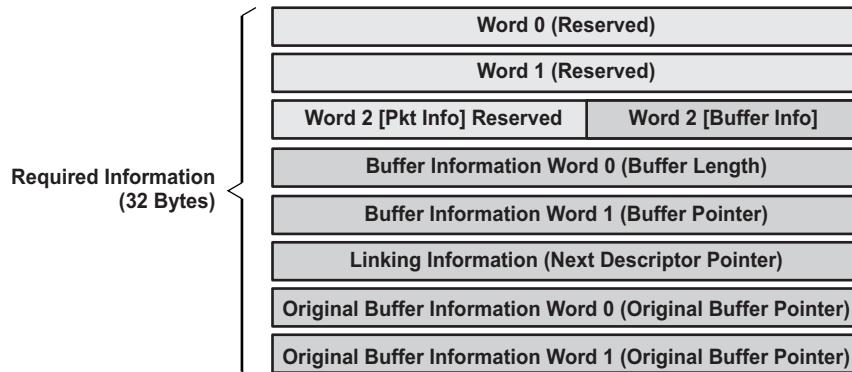
Host buffer descriptors are designed to be linked onto a host packet descriptor or another host buffer descriptor to provide support for unlimited scatter / gather type operations. Host buffer descriptors provide information about a single corresponding data buffer. Every host buffer descriptor stores the following information:

- Pointer to the first valid byte in the data
- Length of the data buffer
- Pointer to the next buffer descriptor in the packet

Buffer descriptors always contain 32 bytes of required information. Since it is a requirement that it is possible to convert a descriptor between a Buffer Descriptor and a Packet Descriptor (by filling in the appropriate fields) in practice, Buffer Descriptors will be allocated using the same sizes as Packet Descriptors. In addition, since the 5 LSBs of the Descriptor Pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is always 32 bytes

The buffer descriptor layout is shown in [Figure 16-14](#) and described within [Table 16-17](#) through [Table 16-24](#).

**Figure 16-14. Buffer Descriptor (BD) Layout**



**Table 16-17. Buffer Descriptor Word 0 (BD0) Bit Field Descriptions**

| Bits | Field Name | Description |
|------|------------|-------------|
| 31-0 | Reserved   | Reserved    |

**Table 16-18. Buffer Descriptor Word 1 (BD1) Bit Field Descriptions**

| Bits | Field Name | Description |
|------|------------|-------------|
| 31-0 | Reserved   | Reserved    |

**Table 16-19. Buffer Descriptor Word 2 (BD2) Bit Field Descriptions**

| Bits  | Field Name                | Description  |
|-------|---------------------------|--|
| 31-15 | Reserved                  | Reserved   |
| 14    | On-chip                   | This field indicates whether or not this descriptor is in a region which is on-chip memory space (1) or in external memory (0).  |
| 13-12 | Packet return queue mgr # | This field indicates which queue manager in the system the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU. There is only 1 queue manager in the USB HS/FS device controller. This field must always be 0. |
| 11-0  | Packet return queue #     | This field indicates the queue number within the selected queue manager that the descriptor is to be returned to after transmission is complete. This field is not altered by the DMA during transmission or reception and is initialized by the CPU.  |

**Table 16-20. Buffer Descriptor Word 3 (BD3) Bit Field Descriptions**

| Bits  | Field Name      | Description   |
|-------|-----------------|---|
| 31-22 | Reserved        | Reserved  |
| 21-0  | Buffer 0 length | The buffer length field indicates how many valid data bytes are in the buffer. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception. |

**Table 16-21. Buffer Descriptor Word 4 (BD4) Bit Field Descriptions**

| Bits | Field Name       | Description   |
|------|------------------|---|
| 31-0 | Buffer 0 pointer | The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception. |

**Table 16-22. Buffer Descriptor Word 5 (BD5) Bit Field Descriptions**

| Bits | Field Name               | Description  |
|------|--------------------------|--|
| 31-0 | Next description pointer | The 32-bit word aligned memory address of the next buffer descriptor in the packet. If the value of this pointer is zero, then the current descriptor is the last descriptor in the packet. The CPU initializes this field for transmitted packets. The DMA overwrites this field upon packet reception. |

**Table 16-23. Buffer Descriptor Word 6 (BD6) Bit Field Descriptions**

| Bits  | Field Name               | Description   |
|-------|--------------------------|---|
| 31-22 | Reserved                 | Reserved  |
| 21-0  | Original buffer 0 length | The buffer length field indicates the original size of the buffer in bytes. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer size as allocated by the CPU at initialization. Since the buffer length in Word 3 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer size information. |

**Table 16-24. Buffer Descriptor Word 7 (BD7) Bit Field Descriptions**

| Bits | Field Name                | Description  |
|------|---------------------------|--|
| 31-0 | Original buffer 0 pointer | The buffer pointer is the byte-aligned memory address of the buffer associated with the buffer descriptor. This value is not overwritten during reception. This value is read by the Rx DMA to determine the actual buffer location as allocated by the CPU at initialization. Since the buffer pointer in Word 4 is overwritten by the Rx port during reception, this field is necessary to permanently store the buffer pointer information. |

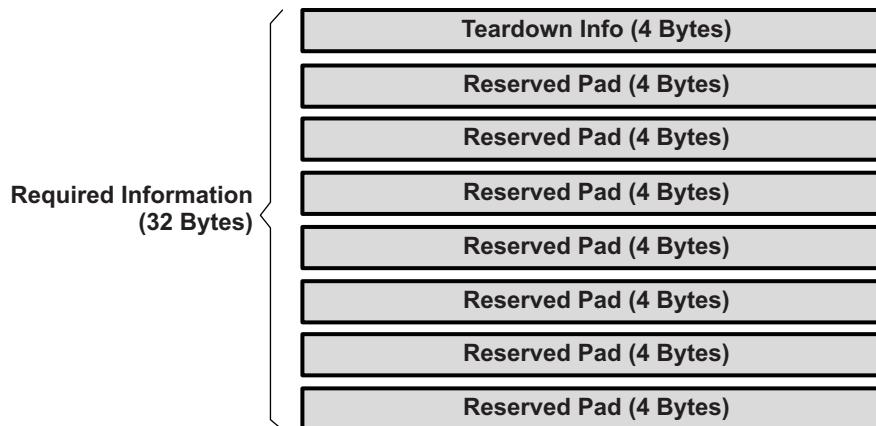
### 16.3.9.2.3 Teardown Descriptor

The Teardown Descriptor is not like the packet or buffer descriptors since it is not used to describe either a packet or a buffer. The teardown descriptor is always 32 bytes long and is comprised of 4 bytes of actual teardown information and 28 bytes of pad. The teardown descriptor layout and associated bit field descriptions are shown in the Figure below and Tables that follows. Since the 5 LSBs of the descriptor pointers are used in CPPI 4.1 for the purpose of indicating the length of the descriptor, the minimum size of a descriptor is 32 bytes.

The teardown descriptor is used to describe a channel halt and teardown event. Channel teardown ensures that when a connection is no longer needed that the hardware can be reliably halted and any remaining packets which had not yet been transmitted can be reclaimed by the host without the possibility of losing buffer or descriptor references (which results in a memory leak).

The teardown descriptor contains the following information:

- Indicator which identifies the descriptor as a teardown packet descriptor
- DMA controller number where teardown
- Channel number within DMA where teardown occurred
- Indicator of whether this teardown was for the Tx or Rx channel

**Figure 16-15. Teardown Descriptor Layout****Table 16-25. Teardown Descriptor Word 0 Bit Field Descriptions**

| Bits  | Field Name      | Description   |
|-------|-----------------|---|
| 31-27 | Descriptor type | The teardown descriptor type is 19 decimal (13h)                |
| 26-17 | Reserved        | Reserved  |
| 16    | TX_RX           | Indicates whether teardown is a TX (0) or RX (1).               |
| 15-10 | DMA number      | Indicates the DMA number for this teardown.                     |
| 9-6   | Reserved        | Reserved  |
| 5-0   | Channel number  | Indicates the channel number within the DMA that was torn down. |

**Table 16-26. Teardown Descriptor Words 1 to 7 Bit Field Descriptions**

| Bits | Field Name | Description |
|------|------------|-------------|
| 31-0 | Reserved   | Reserved    |

Teardown operation of an endpoint requires three operations. The teardown register in the CPPI DMA must be written, the corresponding endpoint bit in TEARDOWN of the USB module must be set, and the FlushFIFO bit in the Mentor USB controller Tx/RxCSR register must be set.

The following is the Transmit teardown procedure highlighting the steps required to be followed:

1. Set the TX\_TEARDOWN bit in the CPPI DMA TX channel n global configuration register (TXGCRn).
2. Set the appropriate TX\_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Write Tx Endpoint Number to teardown to TEARDOWN[TX\_TDOWN] field.
3. Check if the teardown descriptor has been received on the teardown queue: The completion queue (see Queue Assignment table) is usually used as the Teardown Completion queue when the Teardown descriptor has been received, the descriptor address will be loaded onto CTRLD[Completion Queue #] register:
  - (a) If not, go to step 2
  - (b) If so, go to step 4
4. Set the appropriate TX\_TDOWN bit in the USBOTG controller's USB teardown register (TEARDOWN). Set the bit corresponding to the Channel Number within TEARDOWN[TX\_TDOWN] field.
5. Flush the TX FIFO in the Mentor OTG core: Set PERI\_TXCSR[FLUSHFIFO] for the corresponding Endpoint.
6. Re-enable the Tx DMA channel.
  - (a) Clear TXGCRn[TX\_TEARDOWN and TX\_ENABLE] bit.
  - (b) Set TXGCRn[TX\_ENABLE] bit.

### 16.3.9.3 Queue Manager

The queue manager (QM) is a hardware module that is responsible for accelerating management of the queues, i.e. queues are maintained within a queue manager module. Packets are added to a queue by writing the 32-bit descriptor address to a particular memory mapped location in the queue manager module. Packets are de-queued by reading the same location for that particular queue. A single queue manager exists within the device and handles all available 156 queues within the USB subsystem.

Descriptors are queued onto a logical queue (pushed) by writing the descriptor pointer to the Queue N Register D. When the Queue N Register D is written, this kicks off the queue manager causing it to add the descriptor to the tail of queue.

The QM keeps track of the order of the descriptors pushed within a queue or the linking status of the descriptors. To accomplish this linking status, QM will first resolve the 32-bit descriptor pointer into a 16-bit index which is used for linking a queue pointer purposes. Once the address to index computation is performed, i.e., the physical index information is determined, the QM will link that descriptor onto the descriptor chain that is maintained for that logical queue by writing the linking information out to a linking RAM which is external to the queue manager. More on linking RAM discussion would follow in latter sections. The QM will also update the queue tail pointers appropriately. Since logical queues within the queue manager are maintained using linked lists, queues cannot become full and no check for fullness is required before queuing a packet descriptor.

#### 16.3.9.3.1 Queue Types

Several types of queues exist (a total of 156 queues) within the CPPI 4.1 DMA. Regardless of the type of queue, queues are used to hold pointers to Packet or Buffer Descriptors while they are being passed between the Host and / or any of the ports in the system. All queues are maintained within the single Queue Manager module.

The following types of Queues exist:

- Free descriptor queue (unassigned to specific endpoint but assigned to specific endpoint type; receive endpoints – can be used as a receive submit queue)
- Transmit submit queue
- Transmit completion (return) queue
- Receive completion (return) queue
- Teardown queue

Dedicated queues exist where one or more queues are assigned for the use of a single endpoint and non-dedicated queues exist where no specific queue assignment to an endpoint is required (but pertains to receive endpoints only). Transmit endpoints are not allowed to use free descriptor queues.

Dedicated queues exist for each specific endpoint use and non-dedicated queues exist that can be used by any/all receive endpoints. Three queues are reserved for each endpoint/port for transmit actions with two of the three queues being transmit submit queues while the remaining queue is used as a completion/return queue. For receive actions, the only dedicated queues are completion/return queues; one queue is assigned/reserved for each receive endpoint. 32 free descriptor queues that do not have dedicated endpoint-queue assignment exist and these queues are used to service any receive endpoint as receive submit queues.

[Table 16-27](#) displays queue-endpoint assignments.

**Table 16-27. Queue-Endpoint Assignments**

| Queue Start Number | Queue Count | Queue-Endpoint Association                                    |
|--------------------|-------------|---|
| 0                  | 32          | Free Descriptor Queues/Rx Submit Queues (USB0/1 Rx Endpoints) |
| 32                 | 2           | USB0 Tx Endpoint 1  |
| 34                 | 2           | USB0 Tx Endpoint 2  |
| 36                 | 2           | USB0 Tx Endpoint 3  |
| 38                 | 2           | USB0 Tx Endpoint 4  |

**Table 16-27. Queue-Endpoint Assignments (continued)**

| Queue Start Number | Queue Count | Queue-Endpoint Association           |
|--------------------|-------------|--------------------------------------|
| 40                 | 2           | USB0 Tx Endpoint 5                   |
| 42                 | 2           | USB0 Tx Endpoint 6                   |
| 44                 | 2           | USB0 Tx Endpoint 7                   |
| 46                 | 2           | USB0 Tx Endpoint 8                   |
| 48                 | 2           | USB0 Tx Endpoint 9                   |
| 50                 | 2           | USB0 Tx Endpoint 10                  |
| 52                 | 2           | USB0 Tx Endpoint 11                  |
| 54                 | 2           | USB0 Tx Endpoint 12                  |
| 56                 | 2           | USB0 Tx Endpoint 13                  |
| 58                 | 2           | USB0 Tx Endpoint 14                  |
| 60                 | 2           | USB0 Tx Endpoint 15                  |
| 62                 | 2           | USB1 Tx Endpoint 1                   |
| 64                 | 2           | USB1 Tx Endpoint 2                   |
| 66                 | 2           | USB1 Tx Endpoint 3                   |
| 68                 | 2           | USB1 Tx Endpoint 4                   |
| 70                 | 2           | USB1 Tx Endpoint 5                   |
| 72                 | 2           | USB1 Tx Endpoint 6                   |
| 74                 | 2           | USB1 Tx Endpoint 7                   |
| 76                 | 2           | USB1 Tx Endpoint 8                   |
| 78                 | 2           | USB1 Tx Endpoint 9                   |
| 80                 | 2           | USB1 Tx Endpoint 10                  |
| 82                 | 2           | USB1 Tx Endpoint 11                  |
| 84                 | 2           | USB1 Tx Endpoint 12                  |
| 86                 | 2           | USB1 Tx Endpoint 13                  |
| 88                 | 2           | USB1 Tx Endpoint 14                  |
| 90                 | 2           | USB1 Tx Endpoint 15                  |
| 92                 | 1           | Reserved                             |
| 93                 | 1           | USB0 Tx Endpoint 1 completion queue  |
| 94                 | 1           | USB0 Tx Endpoint 2 completion queue  |
| 95                 | 1           | USB0 Tx Endpoint 3 completion queue  |
| 96                 | 1           | USB0 Tx Endpoint 4 completion queue  |
| 97                 | 1           | USB0 Tx Endpoint 5 completion queue  |
| 98                 | 1           | USB0 Tx Endpoint 6 completion queue  |
| 99                 | 1           | USB0 Tx Endpoint 7 completion queue  |
| 100                | 1           | USB0 Tx Endpoint 8 completion queue  |
| 101                | 1           | USB0 Tx Endpoint 9 completion queue  |
| 102                | 1           | USB0 Tx Endpoint 10 completion queue |
| 103                | 1           | USB0 Tx Endpoint 11 completion queue |
| 104                | 1           | USB0 Tx Endpoint 12 completion queue |
| 104                | 1           | USB0 Tx Endpoint 13 completion queue |
| 106                | 1           | USB0 Tx Endpoint 14 completion queue |
| 107                | 1           | USB0 Tx Endpoint 15 completion queue |
| 108                | 1           | Reserved                             |
| 109                | 1           | USB0 Rx Endpoint 1 completion queue  |
| 110                | 1           | USB0 Rx Endpoint 2 completion queue  |
| 111                | 1           | USB0 Rx Endpoint 3 completion queue  |
| 112                | 1           | USB0 Rx Endpoint 4 completion queue  |

**Table 16-27. Queue-Endpoint Assignments (continued)**

| Queue Start Number | Queue Count | Queue-Endpoint Association           |
|--------------------|-------------|--------------------------------------|
| 113                | 1           | USB0 Rx Endpoint 5 completion queue  |
| 114                | 1           | USB0 Rx Endpoint 6 completion queue  |
| 115                | 1           | USB0 Rx Endpoint 7 completion queue  |
| 116                | 1           | USB0 Rx Endpoint 8 completion queue  |
| 117                | 1           | USB0 Rx Endpoint 9 completion queue  |
| 118                | 1           | USB0 Rx Endpoint 10 completion queue |
| 119                | 1           | USB0 Rx Endpoint 11 completion queue |
| 120                | 1           | USB0 Rx Endpoint 12 completion queue |
| 121                | 1           | USB0 Rx Endpoint 13 completion queue |
| 122                | 1           | USB0 Rx Endpoint 14 completion queue |
| 123                | 1           | USB0 Rx Endpoint 15 completion queue |
| 124                | 1           | Reserved                             |
| 125                | 1           | USB1 Tx Endpoint 1 completion queue  |
| 126                | 1           | USB1 Tx Endpoint 2 completion queue  |
| 127                | 1           | USB1 Tx Endpoint 3 completion queue  |
| 128                | 1           | USB1 Tx Endpoint 4 completion queue  |
| 129                | 1           | USB1 Tx Endpoint 5 completion queue  |
| 130                | 1           | USB1 Tx Endpoint 6 completion queue  |
| 131                | 1           | USB1 Tx Endpoint 7 completion queue  |
| 132                | 1           | USB1 Tx Endpoint 8 completion queue  |
| 133                | 1           | USB1 Tx Endpoint 9 completion queue  |
| 134                | 1           | USB1 Tx Endpoint 10 completion queue |
| 135                | 1           | USB1 Tx Endpoint 11 completion queue |
| 136                | 1           | USB1 Tx Endpoint 12 completion queue |
| 137                | 1           | USB1 Tx Endpoint 13 completion queue |
| 138                | 1           | USB1 Tx Endpoint 14 completion queue |
| 139                | 1           | USB1 Tx Endpoint 15 completion queue |
| 140                | 1           | Reserved                             |
| 141                | 1           | USB1 Rx Endpoint 1 completion queue  |
| 142                | 1           | USB1 Rx Endpoint 2 completion queue  |
| 143                | 1           | USB1 Rx Endpoint 3 completion queue  |
| 144                | 1           | USB1 Rx Endpoint 4 completion queue  |
| 145                | 1           | USB1 Rx Endpoint 5 completion queue  |
| 146                | 1           | USB1 Rx Endpoint 6 completion queue  |
| 147                | 1           | USB1 Rx Endpoint 7 completion queue  |
| 148                | 1           | USB1 Rx Endpoint 8 completion queue  |
| 149                | 1           | USB1 Rx Endpoint 9 completion queue  |
| 150                | 1           | USB1 Rx Endpoint 10 completion queue |
| 151                | 1           | USB1 Rx Endpoint 11 completion queue |
| 152                | 1           | USB1 Rx Endpoint 12 completion queue |
| 153                | 1           | USB1 Rx Endpoint 13 completion queue |
| 154                | 1           | USB1 Rx Endpoint 14 completion queue |
| 155                | 1           | USB1 Rx Endpoint 15 completion queue |

#### 16.3.9.3.2 Free Descriptor Queue (Receive Submit Queue)

Receive endpoints/channels use queues, referred to as free descriptor queues, or receive submit queues, to forward completed receive packets to the host. The entries on the free descriptor queues have pre-attached empty buffers whose size and location are described in the original buffer information fields in the descriptor. The host is required to allocate both the descriptor and buffer and pre-link them prior to adding (submitting) a descriptor to one of the available free descriptor queue. The first 32 queues (queue 0 up to queue 31) are reserved for all 30 USB0/1 (endpoints 1 to 15 of each USB module) receive endpoints/channels to handle incoming packets to the device.

#### 16.3.9.3.3 Transmit Submit Queue

Transmit ports use packet queues, referred to as transmit submit queues, to store the packets that are waiting to be transmitted. Each transmit endpoint has dedicated queues (2 queues per port) that are reserved exclusively for a use by a single endpoint. Multiple queues per port/channel are allocated to facilitate quality of service (QoS) for applications that require QoS. The first 30 queues, queue 32 up to queue 61, are allocated for USB0 transmit endpoints 1 to 15 with two queues per endpoint. queues 62 up to queue 91, are allocated for transmit USB1 endpoints 1 to 15.

#### 16.3.9.3.4 Transmit Completion (Return) Queue

Transmit ports use packet queues, referred to as "transmit completion queues, to return packet descriptors to the host after packets have been transmitted. A single queue is reserved to be used by a single transmit endpoint. Application s/w needs to insure that the right set of queues is used to return the Descriptors after the completion of a packet transmission based on the endpoint number used. For USB0 transmit endpoints 1 to 15, queues 93–107 are reserved and assigned for use as a completion queue, respectively. Similarly for USB1 transmit endpoints 1 to 15, queues 125–139 are used as completion queues, respectively.

Transmit Completion Queues are also used to return packet Descriptors when performing a Transmit channel teardown operation.

#### 16.3.9.3.5 Receive Completion (Return) Queue

Receive ports use packet queues, referred to as receive completion queues, to return packet descriptors to the port after packets have been received. A single queue is reserved to be used by a single receive endpoint. Application s/w needs to insure that the right set of queues is used to return the descriptors after the completion of a packet reception based on the endpoint number used. For USB0 receive endpoints 1 to 15 queues 122–136 are reserved/assigned for use as a receive completion queue respectively. Similarly for USB1 receive endpoints 1 to 15, queues 137–151 is used as completion queue respectively.

Receive channel teardown is not necessary for receive transactions and no channel teardown information nor resource is available

#### 16.3.9.3.6 Diverting Queue Packets from one Queue to Another

The host can move the entire contents of one queue to another queue by writing the source queue number and the destination queue number to the Queue Diversion register. When diverting packets, the descriptors are pushed onto the tail of the destination queue.

#### 16.3.9.4 Memory Regions and Linking RAM

In addition to allocating buffer for raw data and memory for Descriptors, the host is responsible for allocating additional memory for exclusive use of the CPPI DMA Queue Manager to be used as a scratch PAD RAM. The queue manager uses this memory to manage states of descriptors submitted within the submit queues. In other words, this memory needs not to be managed by the user software and firmware responsibility lies only for allocation/reserving a chunk of memory for the use of the queue manager. The allocated memory can be a single block of memory that is contiguous or two blocks of memory that are not contiguous. These two blocks of memory are referred to as a Linking RAM Regions and should not be confused with memory regions that are used to store descriptors. That is, the use of the term *region* should be used in the context of its use.

To accomplish the linking of submitted descriptors, the queue manager will first resolve the 32-bit descriptor pointer into a 16-bit index which is used for linking and queue pointer purposes. Once the physical index information is determined, the queue manager will link that descriptor onto the descriptor chain that is maintained for that logical queue by writing the linking information out to a linking RAM which is external to the queue manager. The queue manager will also update the queue tail pointer appropriately. Since logical queues within the queue manager are maintained using linked lists, queues cannot become full and no check for fullness is required before queuing a packet descriptor.

The actual physical size of the Linking RAM region(s) to be allocated depends on the total number of descriptors defined within all memory regions. A minimum of four bytes of memory needs to be allocated for each Descriptor defined within all 16 memory regions.

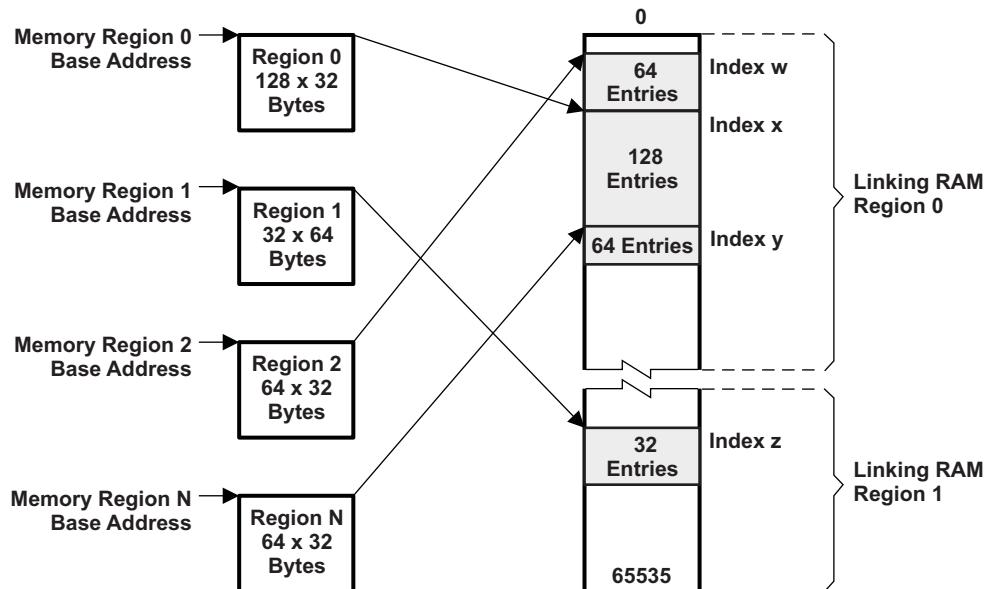
The queue manager has the capability of managing up to 16 memory regions. These memory regions are used to store descriptors of variable sizes. The total number of descriptors that can be managed by the queue manager should not exceed 64K. Each memory region has descriptors of one configurable size, that is, descriptors with different sizes cannot be housed within a single memory region. These 64K descriptors are referenced internally in the queue manager by a 16-bit quantity index.

The information about the Linking RAM regions and the size that are allocated is communicated to the CPPI DMA via three registers dedicated for this purpose. Two of the three registers are used to store the 32-bit aligned start addresses of the Linking RAM regions. The remaining one register is used to store the size of the first Linking RAM. The linking RAM size value stored here is the number of descriptors that is to be managed by the queue manager within that region not the physical size of the buffer, which is four times the number of descriptors.

Note that application is not required to use both linking RAM regions, if the size of the Linking RAM for the first region is large enough to accommodate all descriptors defined. No linking RAM size register for linking RAM region 2 exists. The size of the second linking RAM, when used, is indirectly computed from the total number of descriptors defined less the number of descriptors managed by the first linking RAM.

Figure 16-16 displays the relationship between several memory regions and linking RAM.

**Figure 16-16. Relationship Between Memory Regions and Linking RAM**



### 16.3.9.5 Zero Length Packets

A special case is the handling of null packets with the CPPI 4.1 compliant DMA controller. Upon receiving a zero length USB packet, the XFER DMA will send a data block to the DMA controller with byte count of zero and the zero byte packet bit of INFO Word 2 set. The DMA controller will then perform normal End of Packet termination of the packet, without transferring data.

If a zero-length USB packet is received, the XDMA will send the CDMA a data block with a byte count of 0 and this bit set. The CDMA will then perform normal EOP termination of the packet without transferring data. For transmit, if a packet has this bit set, the XDMA will ignore the CPPI packet size and send a zero-length packet to the USB controller.

### 16.3.9.6 CPPI DMA Scheduler

The CPPI DMA scheduler is responsible for controlling the rate and order between the different Tx and Rx threads/transfers that are provided in the CPPI DMA controller. The scheduler table RAM exists within the scheduler.

The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations to be time division multiplexed between channels in order to share the underlying DMA hardware. A DMA scheduler is used to control the ordering and rate at which this multiplexing occurs.

#### 16.3.9.6.1 CPPI DMA Scheduler Operation

Once the scheduler is enabled it will begin processing the entries in the table. When appropriate the scheduler will pass credits to the DMA controller to perform a Tx or Rx operation. The operation of the DMA controller is as follows:

1. After the DMA scheduler is enabled it begins with the table index set to 0.
2. The scheduler reads the entry pointed to by the index and checks to see if the channel in question is currently in a state where a DMA operation can be accepted.
  - (a) The DMA channel must be enabled AND
  - (b) The CPPI FIFO that the channel talks to has free space on TX (FIFO full signal is not asserted) or a valid block on Rx (FIFO empty signal is not asserted)
3. If the DMA channel is capable of processing a credit to transfer a block, the DMA scheduler will issue that credit via the DMA scheduling interface which is a point to point connection between the DMA Scheduler and the DMA Controller.
  - (a) The DMA controller may not be ready to accept the credit immediately and is provided a `sched_ready` signal which is used to stall the scheduler until it can accept the credit. The DMA controller only asserts the `sched_ready` signal when it is in the IDLE state.
  - (b) Once a credit has been accepted (indicated by `sched_req` and `sched_ready` both asserted), the scheduler will increment the index to the next entry and will start again at step 2.
4. If the channel in question is not currently capable of processing a credit, the scheduler will increment the index in the scheduler table to the next entry and will start at step 2.
5. Note that when the scheduler attempts to increment its index, to the value programmed in the table size register, the index will be reset to 0.

##### 16.3.9.6.1.1 CPPI DMA Scheduler Initialization

Before the scheduler can be used, the host is required to initialize and enable the block. This initialization is performed as follows:

1. The Host initializes entries within an internal memory array in the scheduler. This array contains up to 256 entries (4 entries per table word n where n=0-63) and each entry consists of a DMA channel number and a bit indicating if this is a Tx or Rx opportunity. These entries represent both the order and frequency that various Tx and Rx channels will be processed. A table size of 256 entries allows channel bandwidth to be allocated with a maximum precision of 1/256th of the total DMA bandwidth. The more entries that are present for a given channel, the bigger the slice of the bandwidth that channel will be given. Larger tables can be accommodated to allow for more precision. This array can only be written by the Host, it cannot be read.
2. If the application does not need to use the entire 256 entries, firmware can initialize the portion of the 256 entries and indicate the size of the entries used by writing onto an internal register in the scheduler which sets the actual size of the array (it can be less than 256 entries).
3. The host writes an internal register bit to enable the scheduler. The scheduler is not required to be disabled in order to change the scheduler array contents.

### **16.3.9.6.1.2 CPPI DMA Scheduler Programming Examples**

Consider a three endpoints use on a system with the following configurations: EP1-Tx, EP2-Rx, and EP2-Tx. Two assumptions are considered:

**Case 1:** Assume that you would like to service each enabled endpoints (EP1-Tx, EP2-Rx, and EP2-Tx) with equal priority.

The scheduler handles the rate at which an endpoint is serviced by the number of credits programmed (entries) for that particular endpoint within the scheduler Table Words. The scheduler has up to 256 credits that it can grant and for this example the user can configure the number of entries/credits to be anywhere from 3 to 256 with a set of 3 entries.

However, the optimum and direct programming for this scenario would be programming only the first three entries of the scheduler via scheduler Table WORD[0]. Since this case expects the Scheduler to use only the first three entries, you communicate that by programming DMA\_SCHED\_CTRL.LAST\_ENTRY with 2 (that is, 3 -1). The Enabled Endpoint numbers and the data transfer direction is then communicated by programming the first three entries of WORD[0] (ENTRY0\_CHANNEL = 1: ENTRY0\_RXTX = 0; ENTRY1\_CHANNEL = 2: ENTRY1\_RXTX = 1; ENTRY2\_CHANNEL = 2: ENTRY2\_RXTX = 0). With this programming, the Scheduler will only service the first three entries in a round-robin fashion, checking each credited endpoint for transfer one after the other, and servicing the endpoint that has data to transfer.

**Case 2:** Enabled endpoint EP1-Tx is serviced at twice the rate as the other enabled endpoints (EP2-Rx and EP2-Tx).

The number of entries/credit that has to be awarded to EP1-Tx has to be twice as much of the others. Four entries/credits would suffice to satisfy our requirement with two credits for EP1-Tx, one credit for EP2-Rx, and one credit for EP2-Tx. This requirement is satisfied by allocating any two of the four entries to EP1-Tx endpoint. Again for this example, scheduler Table WORD[0] would suffice since it can handle the first 4 entries. Even though several scenarios exist to programming the order of service for this case, one scenario would be to allow servicing EP1-Tx to back-to-back followed by the other enabled endpoints. Program DMA\_SCHED\_CTRL.LAST\_ENTRY with 3 (that is, 4 -1). Program WORD[0] (ENTRY0\_CHANNEL = 1: ENTRY0\_RXTX = 0; ENTRY1\_CHANNEL = 1: ENTRY1\_RXTX = 0; ENTRY2\_CHANNEL = 2: ENTRY2\_RXTX = 1; ENTRY3\_CHANNEL = 2: ENTRY3\_RXTX = 0).

### **16.3.9.7 CPPI DMA State Registers**

The port must store and maintain state information for each transmit and receive port/channel. The state information is referred to as the Tx DMA State and Rx DMA State.

#### **16.3.9.7.1 Transmit DMA State Registers**

The Tx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures and transmit packets. Each transmit channel has two queues. Each queue has a one head descriptor pointer and one completion pointer. There are thirty Tx DMA State registers; one for each port/channel.

The following information is stored in the Tx DMA State:

- Tx Queue Head Descriptor Pointer(s)
- Tx Completion Pointer(s)
- Protocol specific control/status (port scratchpad)

#### **16.3.9.7.2 Receive DMA State Registers**

The Rx DMA State is a combination of control fields and protocol specific port scratchpad space used to manipulate data structures in order to receive packets. Each receive channel has only one queue. Each channel queue has one head descriptor pointer and one completion pointer. There are thirty Rx DMA State registers; one for each port/channel.

The following information is stored in the Rx DMA State:

- Rx Queue Head Descriptor Pointer
- Rx Queue Completion Pointer

- Rx Buffer Offset

### 16.3.9.8 CPPI DMA Protocols Supported

Four different type of DMA transfers are supported by the CPPI 4.1 DMA; Transparent, RNDIS, Generic RNDIS, and Linux CDC. The following sections will outline the details on these DMA transfer types.

#### 16.3.9.8.1 Transparent DMA Transfer

Transparent Mode DMA operation is the default DMA mode where DMA interrupt is generated whenever a DMA packet is transferred. In the transparent mode, DMA packet size cannot be greater than USB MaxPktSize for the endpoint. This transfer type is ideal for transfer (not packet) sizes that are less than a max packet size.

##### 16.3.9.8.1.1 Transparent DMA Transfer Setup

The following will configure all 30 ports/channels for Transparent DMA Transfer type.

Make sure that RNDIS Mode is disabled globally. This allows the application to configure the CPPI DMA protocol in use to be configured per endpoint need. To disable RNDIS operation, clear RNDIS bit in the USB Control Registers corresponding to the USB Module (default setting), i.e., CTRL0[RNDIS]=0 and CTRL1[RNDIS]=0.

Configure the USB0/1 Tx/Rx DMA Mode Registers (USB0/1 TX(RX)MODE0/1) for the Endpoint field in use is programmed for Transparent Mode, i.e., TXMODE0/1[TXn\_MODE] = 00b and RXMODE0/1[RXn\_MODE] = 00b.

#### 16.3.9.8.2 RNDIS DMA Transfer

RNDIS mode DMA is used for large transfers (i.e., total data size to be transferred is greater than USB MaxPktSize where the MzxPktSize is a multiple of 64 bytes) that requires multiple USB packets. This is accomplished by breaking the larger packet into smaller packets, where each packet size being USB MaxPktSize except the last packet where its size is less than USB MaxPktSize, including zero bytes. This implies that multiple USB packets of MaxPktSize will be received and transferred together as a single large DMA transfer and the DMA interrupt is generated only at the end of the complete reception of DMA transfer. The protocol defines the end of the complete transfer by receiving a short USB packet (smaller than USB MaxPktSize as mentioned in USB specification 2.0). If the DMA packet size is an exact multiple of USB MaxPktSize, the DMA controller waits for a zero byte packet at the end of complete transfer to signify the completion of the transfer.

**NOTE:** RNDIS Mode DMA is supported only when USB MaxPktSize is an integral multiple of 64 bytes.

##### 16.3.9.8.2.1 RNDIS DMA Transfer Setup

If all endpoints in use are desired to operate in RNDIS mode, it is only suffix to configure RNDIS DMA operation in RNDIS mode at the global level and application can ignore individual endpoint DMA mode configuration. This is achieved by programming CTRLR0/1[RNDIS] with a '1.'

However if DMA mode is required to be configured at the endpoint level, it is required to disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS]=0), since global configuration over-rides endpoint configuration.

To configure RNDIS DMA mode use, configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register, i.e., TXMODE0/1[TXn\_MODE] = 01b and RXMODE0/1[RXn\_MODE] = 01b

#### 16.3.9.8.3 Generic RNDIS DMA Transfer

Generic RNDIS DMA transfer mode is identical to the normal RNDIS mode in nearly all respects, except for the exception case where the last packet of the transfer can either be a short packet or the MaxPktSize. When the last packet size is equal to the MaxPktSize then no additional zero-byte packet is sent when using Generic RNDIS transfer. Generic RNDIS transfer makes use of a USB0/1 GENERIC RNDIS EPn Size register (there exists a register for each endpoint) that must be programmed with the

transfer size (in bytes) of the transfer for the USB Module (USB0 or USB1) prior to posting a transfer transaction. If the transfer size is an integer multiple of USB MaxPktSize then no additional zero-byte packet is sent when using Generic RNDIS transfer. However, if a short packet has been sent prior to programmed size count, the transfer would end in a similar fashion an RDIS transfer would behave. For example, if the USB MaxPktSize (Tx/RxMaxP) is programmed with a value of 64, the Generic RNDIS EP Size register for that endpoint must be programmed with a value that is an integer multiple of 64 (for example, 64, 128, 192, 256, etc.) for it behave differently than RNDIS transfer. In other words, when using Generic RNDIS mode and the DMA is tasked to transfer data transfer size that is less or equal the size value programmed within the USB0/1 GENERIC RNDIS EPn Size register.

This means that Generic RNDIS mode will perform data transfer in the same manner as RNDIS mode, closing the CPPI packet if a USB packet with a size less than the USB MaxPktSize size value is received. Otherwise, the packet will be closed when the value in the Generic RNDIS EP Size register is reached.

Using USB0/1 GENERIC RNDIS EPn Size register, a packet of up to 64K bytes (65536 bytes) can be transferred. This is to allow the host software to program the USB module to transfer data that is an exact multiple of the USB MaxPktSize (Tx/RxMaxP programmed value) without having to send an additional short packet to terminate.

**NOTE:** As in RNDIS mode, the USB max packet size (Tx/RxMaxp programmed value) of any Generic RNDIS mode enabled endpoints must be a multiple of 64 bytes. Generic RNDIS acceleration should not be enabled for endpoints where the max packet size is not a multiple of 64 bytes. Only transparent mode should be used for such endpoints.

#### 16.3.9.8.3.1 Generic RNDIS DMA Transfer Setup

Disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS]=0), since global configuration over-rides endpoint configuration.

Configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register, i.e., TXMODE0/1[TXn\_MODE] = 11b and RXMODE0/1[RXn\_MODE] = 11b.

#### 16.3.9.8.4 Linux CDC DMA Transfer

Linux CDC DMA transfer mode acts in the same manner as RNDIS packets, except for the case where the last data matches the max USB packet size requiring additional zero-byte packet transfer in RNDIS mode and this is not the case for Linux CDC. If the last data packet of a transfer is a short packet where the data size is greater than zero and less the USB MaxPktSize, then the behavior of the Linux CDC DMA transfer type is identical with the RNDIS DMA transfer type. The only exception is when the short packet length terminating the transfer is a Null Packet. In this case, instead of transferring the Null Packet, it will transfer a data packet of size 1 byte with the data value of 00h.

In transmit operation, if an endpoint is configured or CDC Linux mode, upon receiving a Null Packet from the CPPI DMA, the XFER DMA will then generate a packet containing 1 byte of data, whose value is 00h, indicating the end of the transfer. During receive operation, the XFER DMA will recognize the one byte zero packet as a termination of the data transfer, and sends a block of data with the EOP indicator set and a byte count of one to the CPPI DMA controller. The CPPI DMA realizing the end of the transfer termination will not update/increase the packet size count of the Host Packet Descriptor.

#### 16.3.9.8.4.1 Linux CDC DMA Transfer Setup

Disable the use of RNDIS at the global level, this is achieved by clearing RNDIS bit field (CTRLR0/1[RNDIS]=0), since global configuration over-rides endpoint configuration.

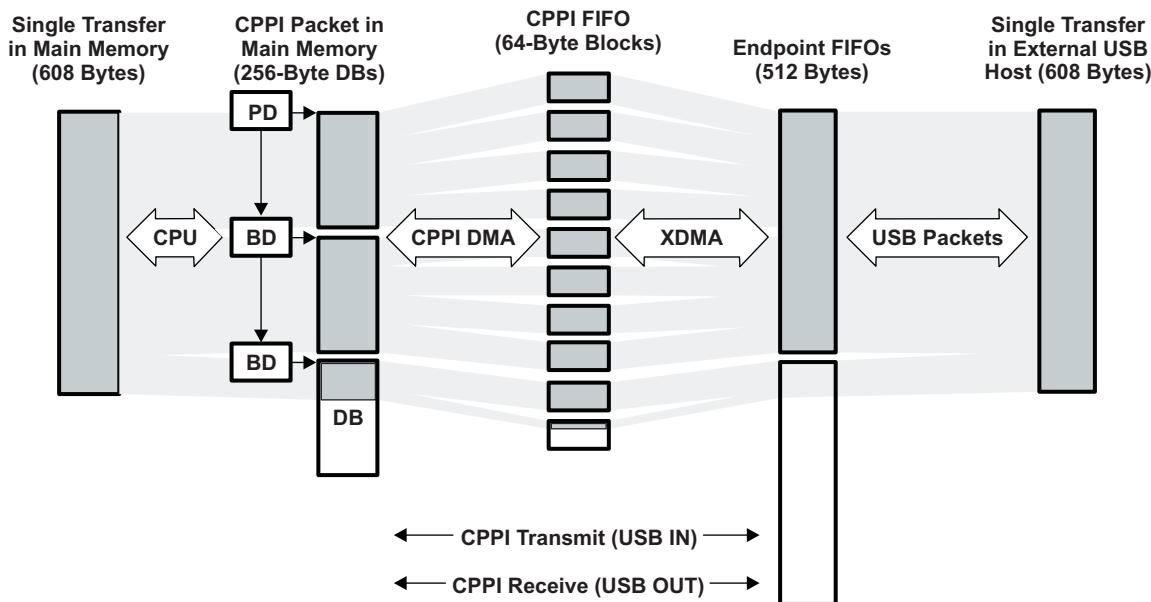
Configure the field that correspond to the USB module endpoint using the corresponding USB0/1 TX(RX) Mode Register, i.e., TXMODE0/1[TXn\_MODE] = 10b and RXMODE0/1[RXn\_MODE] = 10b.

#### 16.3.9.9 USB Data Flow Using DMA

The necessary steps required to perform a USB data transfer using the CPPI 4.1 DMA is expressed using an example for both transmit and receive cases. Assume USB0 is ready to perform a USB data transfer of size 608 bytes (see [Figure 16-17](#)).

The 608 bytes data to be transferred is described using three descriptors. Since each data buffer defined within a single descriptor has a size of 256 bytes, a 608 bytes data buffer would require three descriptors.

**Figure 16-17. High-level Transmit and Receive Data Transfer Example**



#### Example assumptions:

- The CPPI data buffers are 256 bytes in length.
- USB0 module is to be used to perform this transfer. Note that the steps required is similar for USB1 use.
- The USB0 endpoint 1 Tx and Rx endpoint 1 size are programmed to handle max USB packet size of 512 bytes.
- A single transfer length is 608 bytes.
- The SOP offset is 0.

The above translates to the following multi-descriptor setup:

#### Transmit Case

Transmit setup is as follows:

- One packet descriptor with packet length (this is NOT data buffer length, the term packet used here is to mean a transfer length not USB packet) field of 608 bytes and a Data Buffer of size 256 Bytes linked to the 1st host buffer descriptor.
- Two buffer descriptors with first buffer descriptor (this is the one linked to the packet descriptor) defining the second data buffer size of 256 Bytes which in turn is linked to the next (second) buffer descriptor.
- Second buffer fescriptor with a fata buffer size of 96 bytes (can be greater, the packet descriptor contain the size of the packet) linked to no other descriptor (NULL).

#### Receive Case

For this example since each data buffer size is 256 bytes, we would require a minimum of three descriptors that would define data buffer size of 608 bytes. The receive setup is as follows:

- Two buffer descriptors with 256 bytes of data buffer size
- One buffer descriptor with 96 bytes (can be greater) of data buffer size

Within the rest of this section, the following nomenclature is used:

**BD** — Buffer Descriptor or Host Buffer Descriptor

**DB** — Data Buffer Size of 256 Bytes

**PBD** — Pointer to Host Buffer Descriptor

**PD** — Host Packet Descriptor

**PPD** — Pointer to Host Packet Descriptor

**TXSQ** — Transmit Queue or Transmit Submit Queue (for USB0 EP1, use Queues 32 or 33, for USB1 EP1 use Queues 62 or 63)

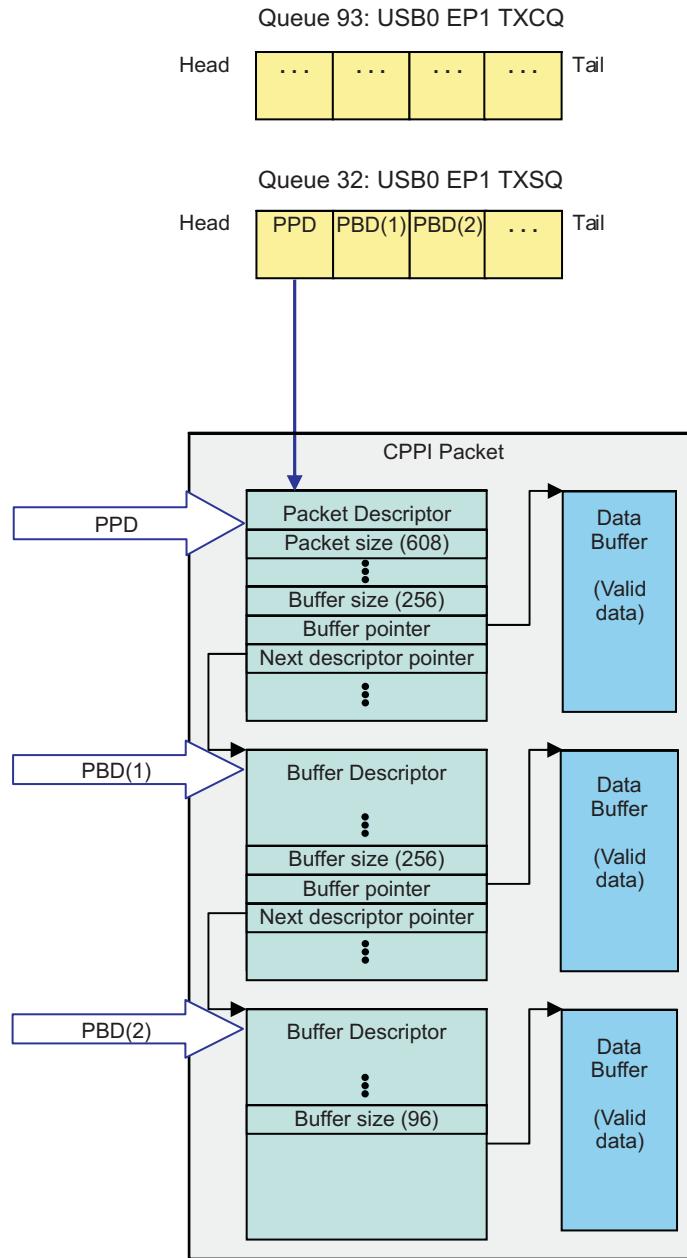
**TXCQ** — Transmit Completion Queue or Transmit Return Queue (for USB0 Tx EP1, use Queue 93, and for USB1 Tx EP1 use Queue 125)

**RXCQ** — Receive Completion Queue or Receive Return Queue (for USB0 Rx EP1, use Queue 109, for USB1 Rx EP1 use Queue 141)

**RXSQ** — Receive Free/Buffer Descriptor Queue or Receive Submit Queue. (For USB0 Rx EP1 Queue 0 is used and for USB1 Rx EP1 Queue 16 should be used)

#### 16.3.9.9.1 *Transmit USB Data Flow Using DMA*

The transmit descriptors and queue status configuration prior to the transfer taking place is shown in [Figure 16-18](#).

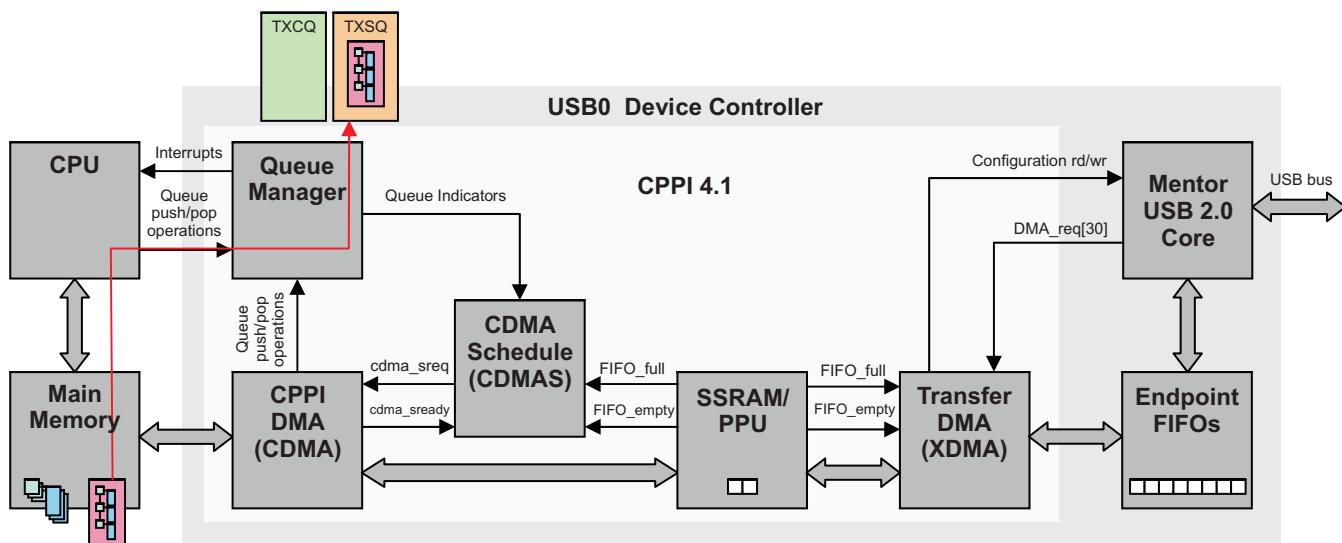
**Figure 16-18. Transmit Descriptors and Queue Status Configuration**


#### 16.3.9.9.1.1 *Transmit Initialization (Step 1)*

The CPU performs the following steps for transmit initialization:

1. Initializes Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. Creates PD, BDs, and DBs in main memory and link as indicated in [Figure 16-19](#).
3. Initializes and configures the Queue Manager, Channel Setup, DMA Scheduler, and Mentor USB 2.0 Core.
4. Adds (pushes) the PPD and the two PBDs to the TXSQ by writing the Packet Descriptor address to the TXSQ CTRL D Register.

[Figure 16-19](#) captures the BD/DB pair in main memory and later submitted within the TXSQ.

**Figure 16-19. Transmit USB Data Flow Example (Initialization)**


#### 16.3.9.9.1.2 CDMA and XDMA Transfer Packet Data Into Endpoint FIFO (Step 2)

The steps for CDMA and XDMA to transfer packet data into endpoint FIFO are as follows:

1. The Queue Manager informs the CDMAS that the TXSQ is not empty.
2. CDMAS checks that the CPPI FIFO FIFO\_full is not asserted, then issues a credit to the CDMA.
3. CDMA reads the Packet Descriptor pointer (PPD) and descriptor size hint from the queue manager
4. For each 64-byte block of data in the packet data payload (note that packet refers here to CPPI packet which is not the same as USB packet and it means to refer to data transfer size):
  - (a) The CDMA transfers a max burst of 64-byte block (OCP burst) from the data to be transferred in main memory to the CPPI FIFO
  - (b) The XDMA sees FIFO\_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
  - (c) The CDMA performs the above two steps ('a' and 'b') three more times since the data size of the HDP is 256 bytes.
5. The CDMA reads the first buffer descriptor pointer (PBD).
6. For each 64-byte block of data in the packet data payload:
  - (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
  - (b) The XDMA sees FIFO\_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
  - (c) The CDMA performs the above two steps ('a' and 'b') two more times since data size of the HBD is 256 bytes.
7. The CDMA reads the second Buffer Descriptor pointer (PBD)
8. For each 64-byte block of data in the packet data payload:
  - (a) The CDMA transfers a max burst of 64-byte block from the data to be transferred in main memory to the CPPI FIFO.
  - (b) The XDMA sees FIFO\_empty not asserted and transfers 64-byte block from CPPI FIFO to Endpoint FIFO.
  - (c) The CDMA transfers the last remaining 32-byte from the data to be transferred in main memory to the CPPI FIFO.
  - (d) The XDMA sees FIFO\_empty not asserted and transfers 32-byte block from CPPI FIFO to Endpoint FIFO.

**16.3.9.9.1.3 USB 2.0 Core Transmits USB Packets for Tx (Step 3)**

1. Once the XDMA has transferred enough 64-byte blocks of burst of data from the CPPI FIFO to fill the Endpoint FIFO (accumulated 512 bytes), it signals the Mentor USB 2.0 Core that a TX packet is ready (sets the endpoint's TxPktRdy bit).
2. The USB 2.0 Core will transmit the packet from the Endpoint FIFO out on the USB BUS when it receives a corresponding IN request from the attached USB Host.
3. After the USB packet is transferred, the USB 2.0 Core issues a TX DMA\_req to the XDMA.
4. This process is repeated until (the above steps '1', '2', and '3') the entire packet has been transmitted. The XDMA will also generate the required termination packet depending on the termination mode configured for the endpoint.

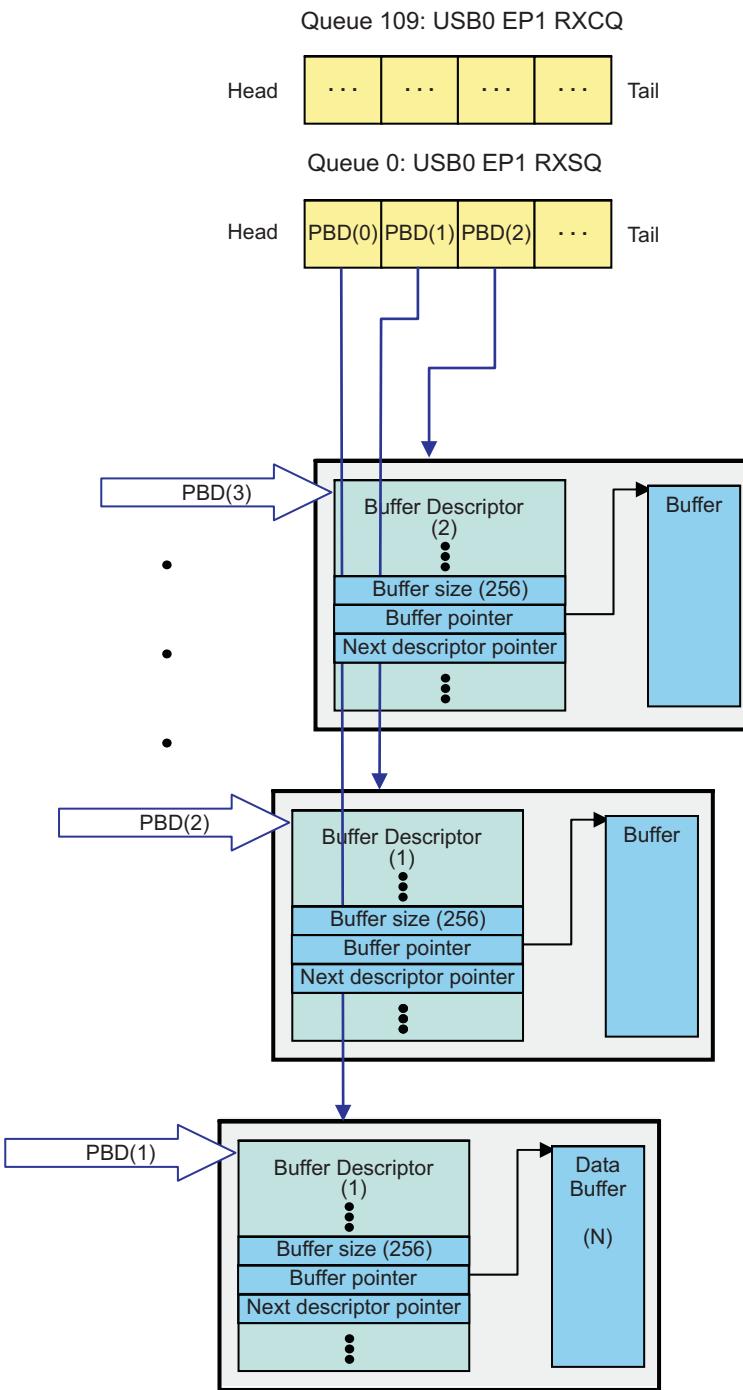
**16.3.9.9.1.4 Return Packet to Completion Queue and Interrupt CPU for Tx (Step 4)**

1. After all data for the packet has been transmitted (as specified by the packet size field), the CDMA will write the pointer of the Packet Descriptor only to the TX Completion Queue specified in the return queue manager / queue number fields of the packet descriptor.
2. The Queue Manager then indicates the status of the TXSQ (empty) to the CDMAS and the TXCQ to the CPU via an interrupt.

**16.3.9.9.2 Receive USB Data Flow Using DMA**

The receive buffer descriptors and queue status configuration prior to the transfer taking place is shown in [Figure 16-20](#).

**Figure 16-20. Receive Buffer Descriptors and Queue Status Configuration**



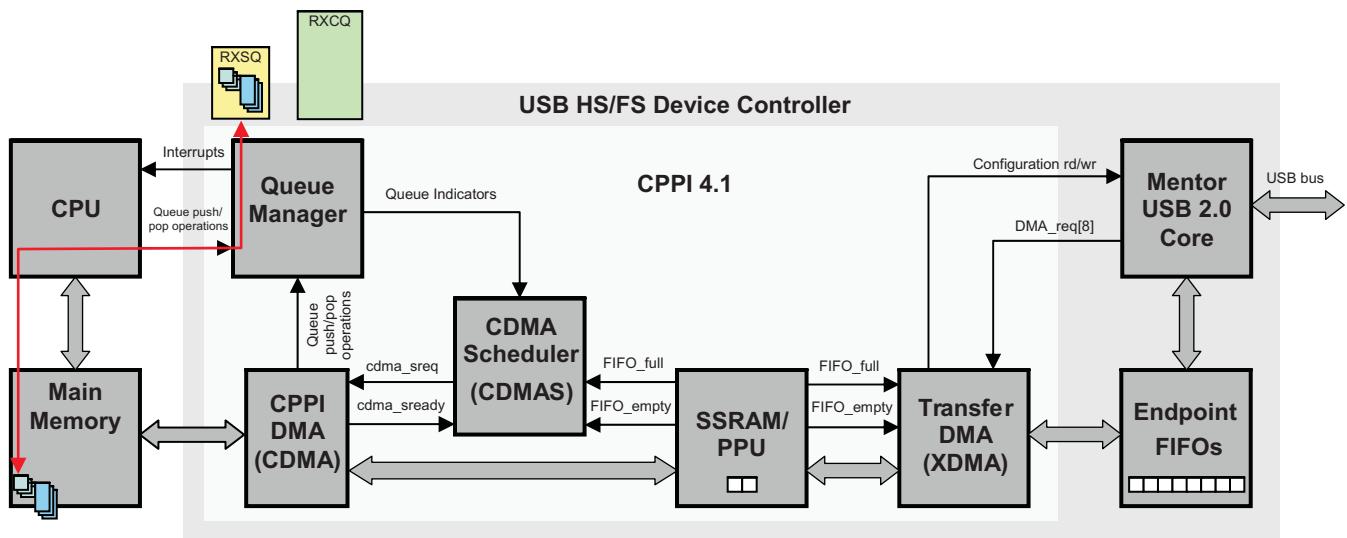
#### 16.3.9.9.2.1 Receive Initialization (Step 1)

1. The CPU initializes Queue Manager with the Memory Region 0 base address and Memory Region 0 size, Link RAM0 Base address, Link RAM0 data size, and Link RAM1 Base address.
2. The CPU creates BDs, and DBs in main memory and link them creating a BD and DB pairs.
3. CPU then initializes the RXCQ queue and configures the Queue Manager, Channel Setup, DMA Scheduler, and USB 2.0 Core.
4. It then adds (pushes) the address of the addresses/pointers of three Buffer Descriptors (PBDs) into the

RXSQ by writing onto CTRL D register.

Figure 16-21 displays receive operation Initialization.

**Figure 16-21. Receive USB Data Flow Example (Initialization)**



#### 16.3.9.9.2.2 USB 2.0 Core Receives a Packet, XDMA Starts Data Transfer for Receive (Step 2)

1. The USB 2.0 Core receives a USB packet from the USB Host and stores it in the Endpoint FIFO.
2. It then asserts a DMA\_req to the XDMA informing it that data is available in the Endpoint FIFO.
3. The XDMA verifies the corresponding CPPI FIFO is not full via the FIFO\_full signal, then starts transferring 64-byte data blocks (burst) from the Endpoint FIFO into the CPPI FIFO.

#### 16.3.9.9.2.3 CDMA Transfers Data from SSRAM / PPU to Main Memory for Receive (Step 3)

1. The CDMAS sees FIFO\_empty de-asserted (there is RX data in the FIFO) and issues a transaction credit to the CDMA.
2. The CDMA begins packet reception by fetching the first PBD from the Queue Manager using the Free Descriptor / Buffer Queue 0 (Rx Submit Queue) index that was initialized in the RX port DMA state for that channel.
3. The CDMA will then begin writing the 64-byte block of packet data into this DB.
4. The CDMA will continue filling the buffer with additional 64-byte blocks of data from the CPPI FIFO and will fetch additional PBD as needed using the Free Descriptor / Buffer Queue 1, 2, and 3 indexes for the 2nd, 3rd, and remaining buffers in the packet. After each buffer is filled, the CDMA writes the buffer descriptor to main memory.

#### 16.3.9.9.2.4 CDMA Completes the Packet Transfer for Receive (Step 4)

1. After the entire packet has been received, the CDMA writes the packet descriptor to main memory.
2. The CDMA then writes the packet descriptor to the RXCQ specified in the Queue Manager / Queue Number fields in the RX Global Configuration Register.
3. The Queue Manager then indicates the status of the RXCQ to the CPU via an interrupt.
4. The CPU can then process the received packet by popping the received packet information from the RXCQ and accessing the packet's data from main memory.

### 16.3.10 USB 2.0 Test Modes

The USB2.0 controller supports the four USB 2.0 test modes (Test\_SE0\_NAK, Test\_J, Test\_K, and Test\_Packet) defined for high-speed functions. It also supports an additional “FIFO access” test mode that can be used to test the operation of the CPU interface, the DMA controller and the RAM block.

The test modes are entered by writing to the TESTMODE register. A test mode is usually requested by the host sending a SET\_FEATURE request to Endpoint 0. When the software receives the request, it should wait until the Endpoint 0 transfer has completed (when it receives the Endpoint 0 interrupt indicating the status phase has completed) then write to the TESTMODE register.

**Note:** These test modes have no purpose in normal operation.

#### 16.3.10.1 TEST\_SE0\_NAK

To enter the Test\_SE0\_NAK test mode, the software should set the TEST\_SE0\_NAK bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it responds to any valid IN token with a NAK.

#### 16.3.10.2 TEST\_J

To enter the Test\_J test mode, the software should set the TEST\_J bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous J on the bus.

#### 16.3.10.3 TEST\_K

To enter the Test\_K test mode, the software should set the TEST\_K bit in the TESTMODE register to 1. The USB controller will then go into a mode in which it transmits a continuous K on the bus.

#### 16.3.10.4 TEST\_PACKET

To execute the Test\_Packet, the software should:

1. Start a session (if the core is being used in Host mode).
2. Write the standard test packet (shown below) to the Endpoint 0 FIFO.
3. Write 8h to the TESTMODE register (TEST\_PACKET = 1) to enter Test\_Packet test mode.
4. Set the TxPktRdy bit in the HOST/PERI\_CSR0 register (bit 1).

The 53 byte test packet to load is as follows (all bytes in hex). The test packet only has to be loaded once; the USB controller will keep re-sending the test packet without any further intervention from the software.

Table 16-28 displays the 53 Bytes test packet content.

**Table 16-28. 53 Bytes Test Packet Content**

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0  | AA |
| AA | EE |
| EE | FE | FF | FF | FF | FF | FF | FF |
| FF | FF | FF | FF | FF | 7F | BF | DF |
| EF | F7 | FB | FD | FC | 7E | BF | DF |
| EF | F7 | FB | FD | 7E |    |    |    |

This data sequence is defined in Universal Serial Bus Specification Revision 2.0, Section 7.1.20. The USB controller will add the DATAA0 PID to the head of the data sequence and the CRC to the end.

### 16.3.10.5 FIFO\_ACCESS

The FIFO Access test mode allows you to test the operation of CPU Interface, the DMA controller (if configured), and the RAM block by loading a packet of up to 64 bytes into the Endpoint 0 FIFO and then reading it back out again. Endpoint 0 is used because it is a bidirectional endpoint that uses the same area of RAM for its Tx and Rx FIFOs.

**NOTE:** The core does not need to be connected to the USB bus to run this test. If it is connected, then no session should be in progress when the test is run.

The test procedure is as follows:

1. Load a packet of up to 64 bytes into the Endpoint 0 Tx FIFO.
2. Set HOST/PERI\_CSR0[TXPKTRDY].
3. Write 40h to the TESTMODE register (FIFO\_ACCESS = 1).
4. Unload the packet from the Endpoint Rx FIFO, again.
5. Set HOST/PERI\_CSR0[SERVICEDRXPKTRDY].

Writing 40h to the TESTMODE register causes the following sequence of events:

The Endpoint 0 CPU pointer (that records the number of bytes to be transmitted) is copied to the Endpoint 0 USB pointer (that records the number of bytes received).

1. The Endpoint 0 CPU pointer is reset.
2. HOST/PERI\_CSR0[TXPKTRDY] is cleared.
3. HOST/PERI\_CSR0[RXPKTRDY] is set.
4. An Endpoint 0 interrupt is generated (if enabled).

The effect of these steps is to make the Endpoint 0 controller act as if the packet loaded into the Tx FIFO has flushed and the same packet received over the USB. The data that was loaded in the Tx FIFO can now be read out of the Rx FIFO.

### 16.3.10.6 FORCE\_HOST

The Force Host test mode enables you to instruct the core to operate in Host mode, regardless of whether it is actually connected to any peripheral; that is, the state of the CID input and the LINESTATE and HOSTDISCON signals are ignored. (While in this mode, the state of the HOSTDISCON signal can be read from the BDEVICE bit in the device control register (DEVCTL)).

This mode, which is selected by writing 80h to the TESTMODE register (FORCE\_HOST = 1), allows implementation of the USB Test\_Force\_Enable (USB 2.0 Specification Section 7.1.20). It can also be used for debugging PHY problems in hardware.

While the FORCE\_HOST bit remains set, the core enters the Host mode when the SESSION bit in DEVCTL is set to 1 and remains in the Host mode until the SESSION bit is cleared to 0 even if a connected device is disconnected during the session. The operating speed while in this mode is determined by the FORCE\_HS and FORCE\_FS bits in the TESTMODE register.

## 16.4 Supported Use Cases

The USB Subsystem supports two independent USB Modules where each USB module can operate as a host or peripheral. When operating as a host, it is capable of interfacing to a single target/peripheral directly or to multiple targets via hub at low-, full-, or high-speed. When operating as a peripheral, it is capable of interfacing to a host at a full- or high-speed.

## 16.5 USB Registers

### 16.5.1 USBSS Registers

Table 16-29 lists the memory-mapped registers for the USBSS. All register offset addresses not listed in Table 16-29 should be considered as reserved locations and the register contents should not be modified.

**Table 16-29. USBSS REGISTERS**

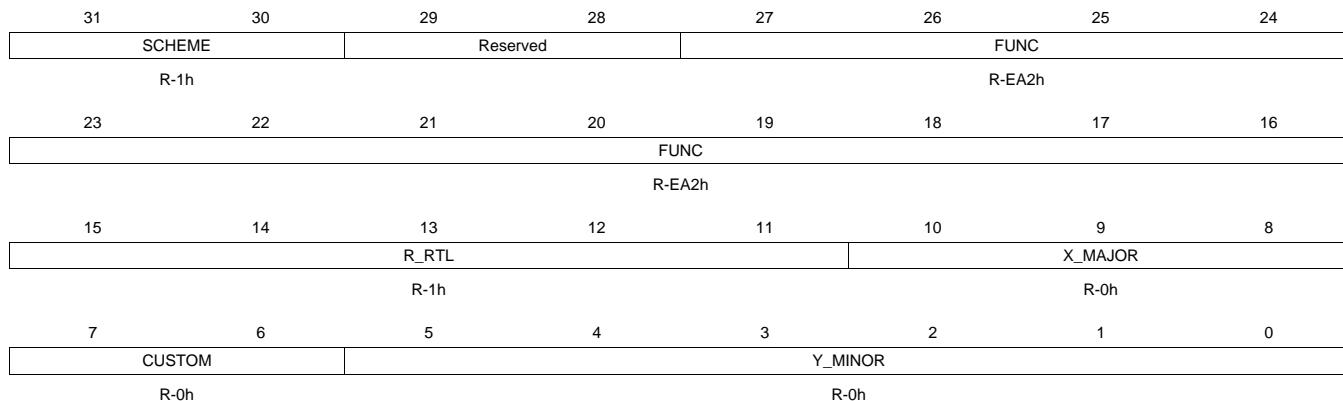
| Offset | Acronym           | Register Name | Section                           |
|--------|-------------------|---------------|-----------------------------------|
| 0h     | REVREG            |               | <a href="#">Section 16.5.1.1</a>  |
| 10h    | SYSCONFIG         |               | <a href="#">Section 16.5.1.2</a>  |
| 24h    | IRQSTATRAW        |               | <a href="#">Section 16.5.1.3</a>  |
| 28h    | IRQSTAT           |               | <a href="#">Section 16.5.1.4</a>  |
| 2Ch    | IRQENABLER        |               | <a href="#">Section 16.5.1.5</a>  |
| 30h    | IRQCLEARR         |               | <a href="#">Section 16.5.1.6</a>  |
| 100h   | IRQDMATHOLDTX00   |               | <a href="#">Section 16.5.1.7</a>  |
| 104h   | IRQDMATHOLDTX01   |               | <a href="#">Section 16.5.1.8</a>  |
| 108h   | IRQDMATHOLDTX02   |               | <a href="#">Section 16.5.1.9</a>  |
| 10Ch   | IRQDMATHOLDTX03   |               | <a href="#">Section 16.5.1.10</a> |
| 110h   | IRQDMATHOLDRX00   |               | <a href="#">Section 16.5.1.11</a> |
| 114h   | IRQDMATHOLDRX01   |               | <a href="#">Section 16.5.1.12</a> |
| 118h   | IRQDMATHOLDRX02   |               | <a href="#">Section 16.5.1.13</a> |
| 11Ch   | IRQDMATHOLDRX03   |               | <a href="#">Section 16.5.1.14</a> |
| 120h   | IRQDMATHOLDTX10   |               | <a href="#">Section 16.5.1.15</a> |
| 124h   | IRQDMATHOLDTX11   |               | <a href="#">Section 16.5.1.16</a> |
| 128h   | IRQDMATHOLDTX12   |               | <a href="#">Section 16.5.1.17</a> |
| 12Ch   | IRQDMATHOLDTX13   |               | <a href="#">Section 16.5.1.18</a> |
| 130h   | IRQDMATHOLDRX10   |               | <a href="#">Section 16.5.1.19</a> |
| 134h   | IRQDMATHOLDRX11   |               | <a href="#">Section 16.5.1.20</a> |
| 138h   | IRQDMATHOLDRX12   |               | <a href="#">Section 16.5.1.21</a> |
| 13Ch   | IRQDMAENABLE0     |               | <a href="#">Section 16.5.1.22</a> |
| 140h   | IRQDMAENABLE1     |               | <a href="#">Section 16.5.1.23</a> |
| 200h   | IRQFRAMETHOLDTX00 |               | <a href="#">Section 16.5.1.25</a> |
| 204h   | IRQFRAMETHOLDTX01 |               | <a href="#">Section 16.5.1.26</a> |
| 208h   | IRQFRAMETHOLDTX02 |               | <a href="#">Section 16.5.1.27</a> |
| 20Ch   | IRQFRAMETHOLDTX03 |               | <a href="#">Section 16.5.1.28</a> |
| 210h   | IRQFRAMETHOLDRX00 |               | <a href="#">Section 16.5.1.29</a> |
| 214h   | IRQFRAMETHOLDRX01 |               | <a href="#">Section 16.5.1.30</a> |
| 218h   | IRQFRAMETHOLDRX02 |               | <a href="#">Section 16.5.1.31</a> |
| 21Ch   | IRQFRAMETHOLDRX03 |               | <a href="#">Section 16.5.1.32</a> |
| 220h   | IRQFRAMETHOLDTX10 |               | <a href="#">Section 16.5.1.33</a> |
| 224h   | IRQFRAMETHOLDTX11 |               | <a href="#">Section 16.5.1.34</a> |
| 228h   | IRQFRAMETHOLDTX12 |               | <a href="#">Section 16.5.1.35</a> |
| 22Ch   | IRQFRAMETHOLDTX13 |               | <a href="#">Section 16.5.1.36</a> |
| 230h   | IRQFRAMETHOLDRX10 |               | <a href="#">Section 16.5.1.37</a> |
| 234h   | IRQFRAMETHOLDRX11 |               | <a href="#">Section 16.5.1.38</a> |
| 238h   | IRQFRAMETHOLDRX12 |               | <a href="#">Section 16.5.1.39</a> |
| 23Ch   | IRQFRAMETHOLDRX13 |               | <a href="#">Section 16.5.1.40</a> |
| 240h   | IRQFRAMEENABLE0   |               | <a href="#">Section 16.5.1.41</a> |
| 244h   | IRQFRAMEENABLE1   |               | <a href="#">Section 16.5.1.42</a> |



### 16.5.1.1 REVREG Register (offset = 0h) [reset = 4EA20800h]

REVREG is shown in [Figure 16-22](#) and described in [Table 16-30](#).

**Figure 16-22. REVREG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

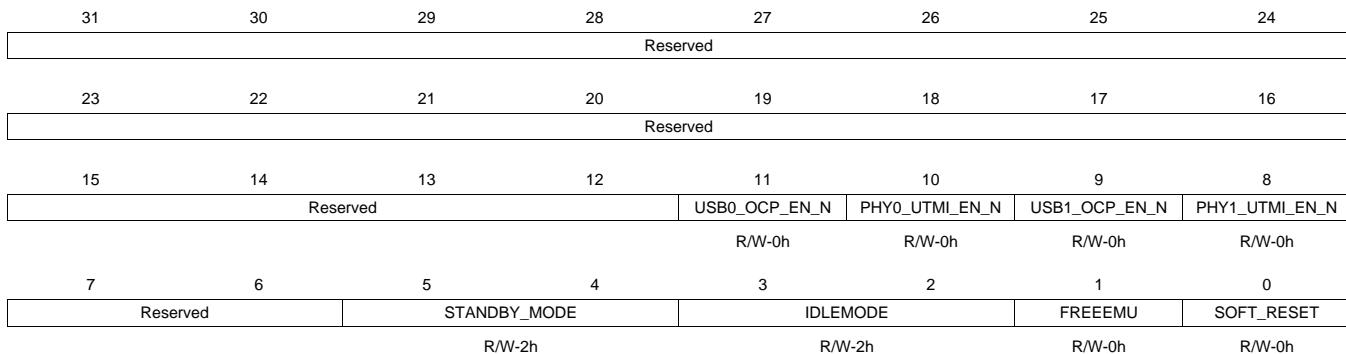
**Table 16-30. REVREG Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description   |
|-------|---------|------|-------|---|
| 31-30 | SCHEME  | R    | 1h    | Used to distinguish between legacy interface scheme and current.<br>0 = Legacy<br>1 = Current |
| 27-16 | FUNC    | R    | EA2h  | Function indicates a software compatible module family.                                       |
| 15-11 | R RTL   | R    | 1h    | RTL revision.<br>Will vary depending on release.  |
| 10-8  | X_MAJOR | R    | 0h    | Major revision.   |
| 7-6   | CUSTOM  | R    | 0h    | Custom revision   |
| 5-0   | Y_MINOR | R    | 0h    | Minor revision.<br>USBSS Revision Register  |

### 16.5.1.2 SYSCONFIG Register (offset = 10h) [reset = 28h]

SYSCONFIG is shown in [Figure 16-23](#) and described in [Table 16-31](#).

**Figure 16-23. SYSCONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-31. SYSCONFIG Register Field Descriptions**

| Bit | Field          | Type | Reset | Description   |
|-----|----------------|------|-------|---|
| 11  | USB0_OCP_EN_N  | R/W  | 0h    | Active low clock enable for usb0_ocp_clk<br>0 = enable<br>1 = disable   |
| 10  | PHY0_UTMI_EN_N | R/W  | 0h    | Active low clock enable for phy0_utmi_clk<br>0 = enable<br>1 = disable  |
| 9   | USB1_OCP_EN_N  | R/W  | 0h    | Active low clock enable for usb1_ocp_clk<br>0 = enable<br>1 = disable   |
| 8   | PHY1_UTMI_EN_N | R/W  | 0h    | Active low clock enable for phy1_utmi_clk<br>0 = enable<br>1 = disable  |
| 5-4 | STANDBY_MODE   | R/W  | 2h    | Configuration of the local initiator state management mode.<br>0 = force-standby mode.<br>1 = no-standby mode.<br>2 = smart-standby mode.<br>3 = Reserved.                  |
| 3-2 | IDLEMODE       | R/W  | 2h    | Configuration of the local target state management mode.<br>0 = force-idle mode.<br>1 = no-idle mode.<br>2 = smart-idle mode.<br>3 = Reserved.                              |
| 1   | FREEEMU        | R/W  | 0h    | Sensitivity to emulation (debug) suspend input signal.<br>0 = sensitive to emulation suspend<br>1 = NOT sensitive to emulation suspend                                      |
| 0   | SOFT_RESET     | R/W  | 0h    | Software reset of USBSS, USB0, and USB1 modules<br>Write 0 = No action.<br>Write 1 = Initiate software reset.<br>Read 0 = Reset done, no action.<br>Read 1 = Reset ongoing. |

### 16.5.1.3 IRQSTATRAW Register (offset = 24h) [reset = 0h]

IRQSTATRAW is shown in [Figure 16-24](#) and described in [Table 16-32](#).

**Figure 16-24. IRQSTATRAW Register**

|          |    |    |    |              |                   |                   |              |
|----------|----|----|----|--------------|-------------------|-------------------|--------------|
| 31       | 30 | 29 | 28 | 27           | 26                | 25                | 24           |
| Reserved |    |    |    |              |                   |                   |              |
| 23       | 22 | 21 | 20 | 19           | 18                | 17                | 16           |
| Reserved |    |    |    |              |                   |                   |              |
| 15       | 14 | 13 | 12 | 11           | 10                | 9                 | 8            |
| Reserved |    |    |    | RX_PKT_CMP_1 | TX_PKT_CMP_1      | RX_PKT_CMP_0      | TX_PKT_CMP_0 |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            | R/W-0h       |
| 7        | 6  | 5  | 4  | 3            | 2                 | 1                 | 0            |
| Reserved |    |    |    | PD_CMP_FLAG  | RX_MOP_STARVATION | RX_SOP_STARVATION |              |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            |              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-32. IRQSTATRAW Register Field Descriptions**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 11  | RX_PKT_CMP_1      | R/W  | 0h    | Interrupt status for USB1 Rx CPPI DMA packet completion status  |
| 10  | TX_PKT_CMP_1      | R/W  | 0h    | Interrupt status for USB1 Tx CPPI DMA packet completion status  |
| 9   | RX_PKT_CMP_0      | R/W  | 0h    | Interrupt status for USB0 Rx CPPI DMA packet completion status  |
| 8   | TX_PKT_CMP_0      | R/W  | 0h    | Interrupt status for USB0 Tx CPPI DMA packet completion status  |
| 2   | PD_CMP_FLAG       | R/W  | 0h    | Interrupt status when the packet is completed, the differ bits is set, and the Packet Descriptor is pushed into the queue manager |
| 1   | RX_MOP_STARVATION | R/W  | 0h    | Interrupt status when queue manager cannot allocate an Rx buffer in the middle of a packet.                                       |
| 0   | RX_SOP_STARVATION | R/W  | 0h    | Interrupt status when queue manager cannot allocate an Rx buffer at the start of a packet.<br>USBSS IRQ_STATUS_RAW Register       |

#### 16.5.1.4 IRQSTAT Register (offset = 28h) [reset = 0h]

IRQSTAT is shown in [Figure 16-25](#) and described in [Table 16-33](#).

**Figure 16-25. IRQSTAT Register**

|          |    |    |    |              |                   |                   |              |
|----------|----|----|----|--------------|-------------------|-------------------|--------------|
| 31       | 30 | 29 | 28 | 27           | 26                | 25                | 24           |
| Reserved |    |    |    |              |                   |                   |              |
| 23       | 22 | 21 | 20 | 19           | 18                | 17                | 16           |
| Reserved |    |    |    |              |                   |                   |              |
| 15       | 14 | 13 | 12 | 11           | 10                | 9                 | 8            |
| Reserved |    |    |    | RX_PKT_CMP_1 | TX_PKT_CMP_1      | RX_PKT_CMP_0      | TX_PKT_CMP_0 |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            | R/W-0h       |
| 7        | 6  | 5  | 4  | 3            | 2                 | 1                 | 0            |
| Reserved |    |    |    | PD_CMP_FLAG  | RX_MOP_STARVATION | RX_SOP_STARVATION |              |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            |              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-33. IRQSTAT Register Field Descriptions**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 11  | RX_PKT_CMP_1      | R/W  | 0h    | Interrupt status for USB1 Rx CPPI DMA packet completion status  |
| 10  | TX_PKT_CMP_1      | R/W  | 0h    | Interrupt status for USB1 Tx CPPI DMA packet completion status  |
| 9   | RX_PKT_CMP_0      | R/W  | 0h    | Interrupt status for USB0 Rx CPPI DMA packet completion status  |
| 8   | TX_PKT_CMP_0      | R/W  | 0h    | Interrupt status for USB0 Tx CPPI DMA packet completion status  |
| 2   | PD_CMP_FLAG       | R/W  | 0h    | Interrupt status when the packet is completed, the differ bits is set, and the Packet Descriptor is pushed into the queue manager |
| 1   | RX_MOP_STARVATION | R/W  | 0h    | Interrupt status when queue manager cannot allocate an Rx buffer in the middle of a packet.                                       |
| 0   | RX_SOP_STARVATION | R/W  | 0h    | Interrupt status when queue manager cannot allocate an Rx buffer at the start of a packet.<br>USBSS IRQ_STATUS Register           |

### 16.5.1.5 IRQENABLE Register (offset = 2Ch) [reset = 0h]

IRQENABLE is shown in [Figure 16-26](#) and described in [Table 16-34](#).

**Figure 16-26. IRQENABLE Register**

|          |    |    |    |              |                   |                   |              |
|----------|----|----|----|--------------|-------------------|-------------------|--------------|
| 31       | 30 | 29 | 28 | 27           | 26                | 25                | 24           |
| Reserved |    |    |    |              |                   |                   |              |
| 23       | 22 | 21 | 20 | 19           | 18                | 17                | 16           |
| Reserved |    |    |    |              |                   |                   |              |
| 15       | 14 | 13 | 12 | 11           | 10                | 9                 | 8            |
| Reserved |    |    |    | RX_PKT_CMP_1 | TX_PKT_CMP_1      | RX_PKT_CMP_0      | TX_PKT_CMP_0 |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            | R/W-0h       |
| 7        | 6  | 5  | 4  | 3            | 2                 | 1                 | 0            |
| Reserved |    |    |    | PD_CMP_FLAG  | RX_MOP_STARVATION | RX_SOP_STARVATION |              |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            |              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-34. IRQENABLE Register Field Descriptions**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 11  | RX_PKT_CMP_1      | R/W  | 0h    | Interrupt enable for USB1 Rx CPPI DMA packet completion status  |
| 10  | TX_PKT_CMP_1      | R/W  | 0h    | Interrupt enable for USB1 Tx CPPI DMA packet completion status  |
| 9   | RX_PKT_CMP_0      | R/W  | 0h    | Interrupt enable for USB0 Rx CPPI DMA packet completion status  |
| 8   | TX_PKT_CMP_0      | R/W  | 0h    | Interrupt enable for USB0 Tx CPPI DMA packet completion status  |
| 2   | PD_CMP_FLAG       | R/W  | 0h    | Interrupt enable when the packet is completed, the differ bits is set, and the Packet Descriptor is pushed into the queue manager |
| 1   | RX_MOP_STARVATION | R/W  | 0h    | Interrupt enable when queue manager cannot allocate an Rx buffer in the middle of a packet.                                       |
| 0   | RX_SOP_STARVATION | R/W  | 0h    | Interrupt enable when queue manager cannot allocate an Rx buffer at the start of a packet.<br>USBSS IRQ_ENABLE_SET Register       |

### 16.5.1.6 IRQCLEARR Register (offset = 30h) [reset = 0h]

IRQCLEARR is shown in [Figure 16-27](#) and described in [Table 16-35](#).

**Figure 16-27. IRQCLEARR Register**

|          |    |    |    |              |                   |                   |              |
|----------|----|----|----|--------------|-------------------|-------------------|--------------|
| 31       | 30 | 29 | 28 | 27           | 26                | 25                | 24           |
| Reserved |    |    |    |              |                   |                   |              |
| 23       | 22 | 21 | 20 | 19           | 18                | 17                | 16           |
| Reserved |    |    |    |              |                   |                   |              |
| 15       | 14 | 13 | 12 | 11           | 10                | 9                 | 8            |
| Reserved |    |    |    | RX_PKT_CMP_1 | TX_PKT_CMP_1      | RX_PKT_CMP_0      | TX_PKT_CMP_0 |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            | R/W-0h       |
| 7        | 6  | 5  | 4  | 3            | 2                 | 1                 | 0            |
| Reserved |    |    |    | PD_CMP_FLAG  | RX_MOP_STARVATION | RX_SOP_STARVATION |              |
| R/W-0h   |    |    |    | R/W-0h       | R/W-0h            | R/W-0h            |              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-35. IRQCLEARR Register Field Descriptions**

| Bit | Field             | Type | Reset | Description   |
|-----|-------------------|------|-------|---|
| 11  | RX_PKT_CMP_1      | R/W  | 0h    | Interrupt enable for USB1 Rx CPPI DMA packet completion status  |
| 10  | TX_PKT_CMP_1      | R/W  | 0h    | Interrupt enable for USB1 Tx CPPI DMA packet completion status  |
| 9   | RX_PKT_CMP_0      | R/W  | 0h    | Interrupt enable for USB0 Rx CPPI DMA packet completion status  |
| 8   | TX_PKT_CMP_0      | R/W  | 0h    | Interrupt enable for USB0 Tx CPPI DMA packet completion status  |
| 2   | PD_CMP_FLAG       | R/W  | 0h    | Interrupt enable when the packet is completed, the differ bits is set, and the Packet Descriptor is pushed into the queue manager |
| 1   | RX_MOP_STARVATION | R/W  | 0h    | Interrupt enable when queue manager cannot allocate an Rx buffer in the middle of a packet.                                       |
| 0   | RX_SOP_STARVATION | R/W  | 0h    | Interrupt enable when queue manager cannot allocate an Rx buffer at the start of a packet.<br>USBSS IRQ_ENABLE_CLR Register       |

**16.5.1.7 IRQDMATHOLDTX00 Register (offset = 100h) [reset = 0h]**

 IRQDMATHOLDTX00 is shown in [Figure 16-28](#) and described in [Table 16-36](#).

**Figure 16-28. IRQDMATHOLDTX00 Register**

|                 |    |    |    |    |    |    |    |
|-----------------|----|----|----|----|----|----|----|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_TX0_3 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX0_2 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX0_1 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved        |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-36. IRQDMATHOLDTX00 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_TX0_3 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 3. |
| 23-16 | DMA_THRES_TX0_2 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 2. |
| 15-8  | DMA_THRES_TX0_1 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 1. |

**16.5.1.8 IRQDMATHOLDTX01 Register (offset = 104h) [reset = 0h]**

 IRQDMATHOLDTX01 is shown in [Figure 16-29](#) and described in [Table 16-37](#).

**Figure 16-29. IRQDMATHOLDTX01 Register**

|                 |    |    |    |    |    |    |    |
|-----------------|----|----|----|----|----|----|----|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_TX0_7 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX0_6 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX0_5 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX0_4 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-37. IRQDMATHOLDTX01 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_TX0_7 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 7.   |
| 23-16 | DMA_THRES_TX0_6 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 6.   |
| 15-8  | DMA_THRES_TX0_5 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 5.   |
| 7-0   | DMA_THRES_TX0_4 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 4.<br>USBSS IRQ_DMA_THRESHOLD_TX0_1 Register |

### 16.5.1.9 IRQDMATHOLDTX02 Register (offset = 108h) [reset = 0h]

IRQDMATHOLDTX02 is shown in [Figure 16-30](#) and described in [Table 16-38](#).

**Figure 16-30. IRQDMATHOLDTX02 Register**

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_TX0_11 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX0_10 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX0_9  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX0_8  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-38. IRQDMATHOLDTX02 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_TX0_11 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 11.   |
| 23-16 | DMA_THRES_TX0_10 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 10.   |
| 15-8  | DMA_THRES_TX0_9  | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 9.  |
| 7-0   | DMA_THRES_TX0_8  | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 8.<br>USBSS IRQ_DMA_THRSHOLD_TX0_2 Register |

**16.5.1.10 IRQDMATHOLDTX03 Register (offset = 10Ch) [reset = 0h]**

 IRQDMATHOLDTX03 is shown in [Figure 16-31](#) and described in [Table 16-39](#).

**Figure 16-31. IRQDMATHOLDTX03 Register**

| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|------------------|----|----|----|----|----|----|----|
| DMA_THRES_TX0_15 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX0_14 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX0_13 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX0_12 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-39. IRQDMATHOLDTX03 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_TX0_15 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 15.   |
| 23-16 | DMA_THRES_TX0_14 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 14.   |
| 15-8  | DMA_THRES_TX0_13 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 13.   |
| 7-0   | DMA_THRES_TX0_12 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB0 Endpoint 12.<br>USBSS IRQ_DMA_THRESHOLD_TX0_3 Register |

**16.5.1.11 IRQDMATHOLDRX00 Register (offset = 110h) [reset = 0h]**

 IRQDMATHOLDRX00 is shown in [Figure 16-32](#) and described in [Table 16-40](#).

**Figure 16-32. IRQDMATHOLDRX00 Register**

|                 |    |    |    |    |    |    |    |
|-----------------|----|----|----|----|----|----|----|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX0_3 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX0_2 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX0_1 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved        |    |    |    |    |    |    |    |

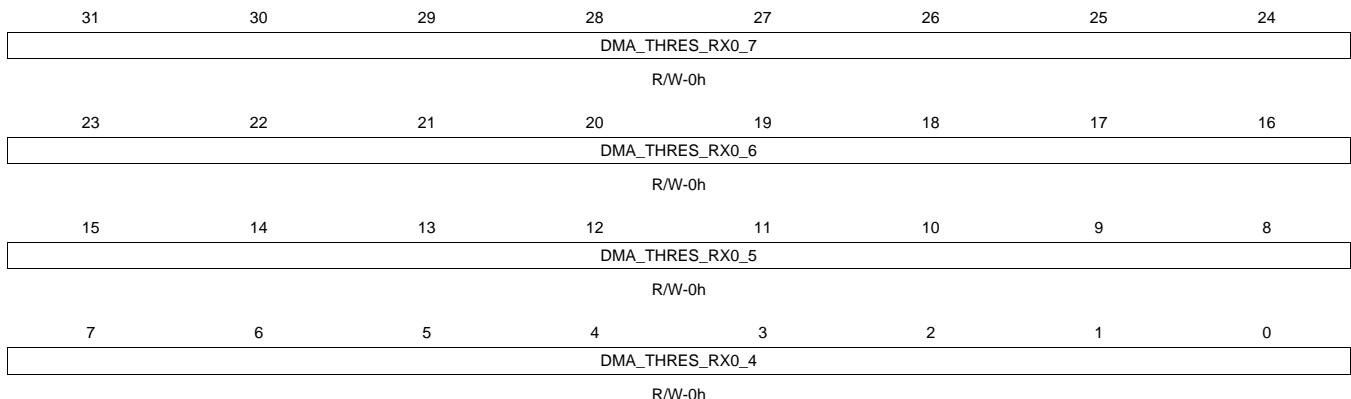
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-40. IRQDMATHOLDRX00 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_RX0_3 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 3. |
| 23-16 | DMA_THRES_RX0_2 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 2. |
| 15-8  | DMA_THRES_RX0_1 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 1. |

**16.5.1.12 IRQDMATHOLDRX01 Register (offset = 114h) [reset = 0h]**

 IRQDMATHOLDRX01 is shown in [Figure 16-33](#) and described in [Table 16-41](#).

**Figure 16-33. IRQDMATHOLDRX01 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-41. IRQDMATHOLDRX01 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_RX0_7 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 7.   |
| 23-16 | DMA_THRES_RX0_6 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 6.   |
| 15-8  | DMA_THRES_RX0_5 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 5.   |
| 7-0   | DMA_THRES_RX0_4 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 4.<br>USBSS IRQ_DMA_THRESHOLD_RX0_1 Register |

**16.5.1.13 IRQDMATHOLDRX02 Register (offset = 118h) [reset = 0h]**

 IRQDMATHOLDRX02 is shown in [Figure 16-34](#) and described in [Table 16-42](#).

**Figure 16-34. IRQDMATHOLDRX02 Register**

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX0_11 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX0_10 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX0_9  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_RX0_8  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-42. IRQDMATHOLDRX02 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | DMA_THRES_RX0_11 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 11.  |
| 23-16 | DMA_THRES_RX0_10 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 10.  |
| 15-8  | DMA_THRES_RX0_9  | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 9.   |
| 7-0   | DMA_THRES_RX0_8  | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 8.<br>USBSS IRQ_DMA_THRESHOLD_RX0_2 Register |

**16.5.1.14 IRQDMATHOLDRX03 Register (offset = 11Ch) [reset = 0h]**

 IRQDMATHOLDRX03 is shown in [Figure 16-35](#) and described in [Table 16-43](#).

**Figure 16-35. IRQDMATHOLDRX03 Register**

| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|------------------|----|----|----|----|----|----|----|
| DMA_THRES_RX0_15 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX0_14 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX0_13 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_RX0_12 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-43. IRQDMATHOLDRX03 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_RX0_15 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 15.   |
| 23-16 | DMA_THRES_RX0_14 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 14.   |
| 15-8  | DMA_THRES_RX0_13 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 13.   |
| 7-0   | DMA_THRES_RX0_12 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB0 Endpoint 12.<br>USBSS IRQ_DMA_THRESHOLD_RX0_3 Register |

**16.5.1.15 IRQDMATHOLDTX10 Register (offset = 120h) [reset = 0h]**

 IRQDMATHOLDTX10 is shown in [Figure 16-36](#) and described in [Table 16-44](#).

**Figure 16-36. IRQDMATHOLDTX10 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-----------------|----|----|----|----|----|----|----|
| DMA_THRES_TX1_3 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| DMA_THRES_TX1_2 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| DMA_THRES_TX1_1 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| Reserved        |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-44. IRQDMATHOLDTX10 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_TX1_3 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 3. |
| 23-16 | DMA_THRES_TX1_2 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 2. |
| 15-8  | DMA_THRES_TX1_1 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 1. |

**16.5.1.16 IRQDMATHOLDTX11 Register (offset = 124h) [reset = 0h]**

 IRQDMATHOLDTX11 is shown in [Figure 16-37](#) and described in [Table 16-45](#).

**Figure 16-37. IRQDMATHOLDTX11 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-----------------|----|----|----|----|----|----|----|
| DMA_THRES_TX1_7 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX1_6 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX1_5 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX1_4 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-45. IRQDMATHOLDTX11 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_TX1_7 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 7.   |
| 23-16 | DMA_THRES_TX1_6 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 6.   |
| 15-8  | DMA_THRES_TX1_5 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 5.   |
| 7-0   | DMA_THRES_TX1_4 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 4.<br>USBSS IRQ_DMA_THRESHOLD_TX1_1 Register |

### 16.5.1.17 IRQDMATHOLDTX12 Register (offset = 128h) [reset = 0h]

IRQDMATHOLDTX12 is shown in [Figure 16-38](#) and described in [Table 16-46](#).

**Figure 16-38. IRQDMATHOLDTX12 Register**

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_TX1_11 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX1_10 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX1_9  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX1_8  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-46. IRQDMATHOLDTX12 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_TX1_11 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 11.   |
| 23-16 | DMA_THRES_TX1_10 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 10.   |
| 15-8  | DMA_THRES_TX1_9  | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 9.  |
| 7-0   | DMA_THRES_TX1_8  | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 8.<br>USBSS IRQ_DMA_THRSHOLD_TX1_2 Register |

**16.5.1.18 IRQDMATHOLDTX13 Register (offset = 12Ch) [reset = 0h]**

 IRQDMATHOLDTX13 is shown in [Figure 16-39](#) and described in [Table 16-47](#).

**Figure 16-39. IRQDMATHOLDTX13 Register**

| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|------------------|----|----|----|----|----|----|----|
| DMA_THRES_TX1_15 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_TX1_14 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_TX1_13 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_TX1_12 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-47. IRQDMATHOLDTX13 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_TX1_15 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 15.   |
| 23-16 | DMA_THRES_TX1_14 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 14.   |
| 15-8  | DMA_THRES_TX1_13 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 13.   |
| 7-0   | DMA_THRES_TX1_12 | R/W  | 0h    | DMA threshold value for tx_pkt_cmp_0 for USB1 Endpoint 12.<br>USBSS IRQ_DMA_THRESHOLD_TX1_3 Register |

**16.5.1.19 IRQDMATHOLDRX10 Register (offset = 130h) [reset = 0h]**

 IRQDMATHOLDRX10 is shown in [Figure 16-40](#) and described in [Table 16-48](#).

**Figure 16-40. IRQDMATHOLDRX10 Register**

|                 |    |    |    |    |    |    |    |
|-----------------|----|----|----|----|----|----|----|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX1_3 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX1_2 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX1_1 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved        |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-48. IRQDMATHOLDRX10 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_RX1_3 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 3. |
| 23-16 | DMA_THRES_RX1_2 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 2. |
| 15-8  | DMA_THRES_RX1_1 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 1. |

**16.5.1.20 IRQDMATHOLDRX11 Register (offset = 134h) [reset = 0h]**

 IRQDMATHOLDRX11 is shown in [Figure 16-41](#) and described in [Table 16-49](#).

**Figure 16-41. IRQDMATHOLDRX11 Register**

|                 |    |    |    |    |    |    |    |
|-----------------|----|----|----|----|----|----|----|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX1_7 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 23              | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX1_6 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 15              | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX1_5 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |
| 7               | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_RX1_4 |    |    |    |    |    |    |    |
| R/W-0h          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-49. IRQDMATHOLDRX11 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31-24 | DMA_THRES_RX1_7 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 7.   |
| 23-16 | DMA_THRES_RX1_6 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 6.   |
| 15-8  | DMA_THRES_RX1_5 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 5.   |
| 7-0   | DMA_THRES_RX1_4 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 4.<br>USBSS IRQ_DMA_THRESHOLD_RX1_1 Register |

**16.5.1.21 IRQDMATHOLDRX12 Register (offset = 138h) [reset = 0h]**

 IRQDMATHOLDRX12 is shown in [Figure 16-42](#) and described in [Table 16-50](#).

**Figure 16-42. IRQDMATHOLDRX12 Register**

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX1_11 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX1_10 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX1_9  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_RX1_8  |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-50. IRQDMATHOLDRX12 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31-24 | DMA_THRES_RX1_11 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 11.  |
| 23-16 | DMA_THRES_RX1_10 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 10.  |
| 15-8  | DMA_THRES_RX1_9  | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 9.   |
| 7-0   | DMA_THRES_RX1_8  | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 8.<br>USBSS IRQ_DMA_THRESHOLD_RX1_2 Register |

**16.5.1.22 IRQDMATHOLDRX13 Register (offset = 13Ch) [reset = 0h]**

 IRQDMATHOLDRX13 is shown in [Figure 16-43](#) and described in [Table 16-51](#).

**Figure 16-43. IRQDMATHOLDRX13 Register**

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| DMA_THRES_RX1_15 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 23               | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA_THRES_RX1_14 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 15               | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| DMA_THRES_RX1_13 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |
| 7                | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| DMA_THRES_RX1_12 |    |    |    |    |    |    |    |
| R/W-0h           |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-51. IRQDMATHOLDRX13 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | DMA_THRES_RX1_15 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 15.   |
| 23-16 | DMA_THRES_RX1_14 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 14.   |
| 15-8  | DMA_THRES_RX1_13 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 13.   |
| 7-0   | DMA_THRES_RX1_12 | R/W  | 0h    | DMA threshold value for rx_pkt_cmp_0 for USB1 Endpoint 12.<br>USBSS IRQ_DMA_THRESHOLD_RX1_3 Register |

**16.5.1.23 IRQDMAENABLE0 Register (offset = 140h) [reset = 0h]**

 IRQDMAENABLE0 is shown in [Figure 16-44](#) and described in [Table 16-52](#).

**Figure 16-44. IRQDMAENABLE0 Register**

|               |          |    |    |              |          |    |              |
|---------------|----------|----|----|--------------|----------|----|--------------|
| 31            | 30       | 29 | 28 | 27           | 26       | 25 | 24           |
| DMA_EN_RX0_15 | Reserved |    |    |              |          |    |              |
| R/W-0h        |          |    |    |              |          |    |              |
| 23            | 22       | 21 | 20 | 19           | 18       | 17 | 16           |
| Reserved      |          |    |    |              |          |    | DMA_EN_RX0_1 |
| R/W-0h        |          |    |    |              |          |    |              |
| 15            | 14       | 13 | 12 | 11           | 10       | 9  | 8            |
| DMA_EN_TX0_15 | Reserved |    |    |              |          |    |              |
| R/W-0h        |          |    |    |              |          |    |              |
| 7             | 6        | 5  | 4  | 3            | 2        | 1  | 0            |
| Reserved      |          |    |    | DMA_EN_TX0_2 | Reserved |    | DMA_EN_TX0_1 |
| R/W-0h        |          |    |    |              |          |    |              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-52. IRQDMAENABLE0 Register Field Descriptions**

| Bit | Field         | Type | Reset | Description   |
|-----|---------------|------|-------|---|
| 31  | DMA_EN_RX0_15 | R/W  | 0h    | DMA threshold enable value for rx_pkt_cmp_0 for USB0 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 17  | DMA_EN_RX0_1  | R/W  | 0h    | DMA threshold enable value for rx_pkt_cmp_0 for USB0 Endpoint 1.  |
| 15  | DMA_EN_TX0_15 | R/W  | 0h    | DMA threshold enable value for tx_pkt_cmp_0 for USB0 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 2   | DMA_EN_TX0_2  | R/W  | 0h    | DMA threshold enable value for tx_pkt_cmp_0 for USB0 Endpoint 2.  |
| 0   | DMA_EN_TX0_1  | R/W  | 0h    | DMA threshold enable value for tx_pkt_cmp_0 for USB0 Endpoint 1.  |

**16.5.1.24 IRQDMAENABLE1 Register (offset = 144h) [reset = 0h]**

 IRQDMAENABLE1 is shown in [Figure 16-45](#) and described in [Table 16-53](#).

**Figure 16-45. IRQDMAENABLE1 Register**

|               |    |    |    |          |    |    |    |
|---------------|----|----|----|----------|----|----|----|
| 31            | 30 | 29 | 28 | 27       | 26 | 25 | 24 |
| DMA_EN_RX1_15 |    |    |    | Reserved |    |    |    |
| R/W-0h        |    |    |    |          |    |    |    |
| 23            | 22 | 21 | 20 | 19       | 18 | 17 | 16 |
| Reserved      |    |    |    |          |    |    |    |
| R/W-0h        |    |    |    |          |    |    |    |
| 15            | 14 | 13 | 12 | 11       | 10 | 9  | 8  |
| DMA_EN_TX1_15 |    |    |    | Reserved |    |    |    |
| R/W-0h        |    |    |    |          |    |    |    |
| 7             | 6  | 5  | 4  | 3        | 2  | 1  | 0  |
| Reserved      |    |    |    |          |    |    |    |
| R/W-0h        |    |    |    |          |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-53. IRQDMAENABLE1 Register Field Descriptions**

| Bit | Field         | Type | Reset | Description   |
|-----|---------------|------|-------|---|
| 31  | DMA_EN_RX1_15 | R/W  | 0h    | DMA threshold enable value for rx_pkt_cmp_1 for USB1 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 17  | DMA_EN_RX1_1  | R/W  | 0h    | DMA threshold enable value for rx_pkt_cmp_1 for USB1 Endpoint 1.  |
| 15  | DMA_EN_TX1_15 | R/W  | 0h    | DMA threshold enable value for tx_pkt_cmp_1 for USB1 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 1   | DMA_EN_TX1_1  | R/W  | 0h    | DMA threshold enable value for tx_pkt_cmp_1 for USB1 Endpoint 1.  |

**16.5.1.25 IRQFRAMETHOLDTX00 Register (offset = 200h) [reset = 0h]**

IRQFRAMETHOLDTX00 is shown in [Figure 16-46](#) and described in [Table 16-54](#).

**Figure 16-46. IRQFRAMETHOLDTX00 Register**

|                   |    |    |    |    |    |    |    |
|-------------------|----|----|----|----|----|----|----|
| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_TX1_3 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_2 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_1 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-54. IRQFRAMETHOLDTX00 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_3 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 3. |
| 23-16 | FRAME_THRES_TX1_2 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 2. |
| 15-8  | FRAME_THRES_TX1_1 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 1. |

**16.5.1.26 IRQFRAMETHOLDTX01 Register (offset = 204h) [reset = 0h]**

 IRQFRAMETHOLDTX01 is shown in [Figure 16-47](#) and described in [Table 16-55](#).

**Figure 16-47. IRQFRAMETHOLDTX01 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_TX1_7 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_6 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_5 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_4 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-55. IRQFRAMETHOLDTX01 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_7 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 7.   |
| 23-16 | FRAME_THRES_TX1_6 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 6.   |
| 15-8  | FRAME_THRES_TX1_5 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 5.   |
| 7-0   | FRAME_THRES_TX1_4 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 4.<br>USBSS IRQ_FRAME_THRESHOLD_TX0_1 Register |

**16.5.1.27 IRQFRAMETHOLDTX02 Register (offset = 208h) [reset = 0h]**

 IRQFRAMETHOLDTX02 is shown in [Figure 16-48](#) and described in [Table 16-56](#).

**Figure 16-48. IRQFRAMETHOLDTX02 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_TX1_11 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_10 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_9  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_8  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-56. IRQFRAMETHOLDTX02 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_11 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 11.  |
| 23-16 | FRAME_THRES_TX1_10 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 10.  |
| 15-8  | FRAME_THRES_TX1_9  | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 9.   |
| 7-0   | FRAME_THRES_TX1_8  | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 8.<br>USBSS IRQ_FRAME_THRESHOLD_TX0_2 Register |

**16.5.1.28 IRQFRAMETHOLDTX03 Register (offset = 20Ch) [reset = 0h]**

IRQFRAMETHOLDTX03 is shown in [Figure 16-49](#) and described in [Table 16-57](#).

**Figure 16-49. IRQFRAMETHOLDTX03 Register**

| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------------------|----|----|----|----|----|----|----|
| FRAME_THRES_TX1_15 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_14 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_13 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_12 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-57. IRQFRAMETHOLDTX03 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-24 | FRAME_THRES_TX1_15 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 15.   |
| 23-16 | FRAME_THRES_TX1_14 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 14.   |
| 15-8  | FRAME_THRES_TX1_13 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 13.   |
| 7-0   | FRAME_THRES_TX1_12 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB0 Endpoint 12.<br>USBSS IRQ_FRAME_THRESHOLD_TX0_3 Register |

**16.5.1.29 IRQFRAMETHOLDRX00 Register (offset = 210h) [reset = 0h]**

 IRQFRAMETHOLDRX00 is shown in [Figure 16-50](#) and described in [Table 16-58](#).

**Figure 16-50. IRQFRAMETHOLDRX00 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_RX1_3 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_2 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_1 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-58. IRQFRAMETHOLDRX00 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_3 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 3. |
| 23-16 | FRAME_THRES_RX1_2 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 2. |
| 15-8  | FRAME_THRES_RX1_1 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 1. |

**16.5.1.30 IRQFRAMETHOLDRX01 Register (offset = 214h) [reset = 0h]**

 IRQFRAMETHOLDRX01 is shown in [Figure 16-51](#) and described in [Table 16-59](#).

**Figure 16-51. IRQFRAMETHOLDRX01 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_RX1_7 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_6 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_5 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_4 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-59. IRQFRAMETHOLDRX01 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_7 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 7.   |
| 23-16 | FRAME_THRES_RX1_6 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 6.   |
| 15-8  | FRAME_THRES_RX1_5 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 5.   |
| 7-0   | FRAME_THRES_RX1_4 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 4.<br>USBSS IRQ_FRAME_THRESHOLD_RX0_1 Register |

**16.5.1.31 IRQFRAMETHOLDRX02 Register (offset = 218h) [reset = 0h]**

 IRQFRAMETHOLDRX02 is shown in [Figure 16-52](#) and described in [Table 16-60](#).

**Figure 16-52. IRQFRAMETHOLDRX02 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_RX1_11 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_10 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_9  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_8  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-60. IRQFRAMETHOLDRX02 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_11 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 11.  |
| 23-16 | FRAME_THRES_RX1_10 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 10.  |
| 15-8  | FRAME_THRES_RX1_9  | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 9.   |
| 7-0   | FRAME_THRES_RX1_8  | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 8.<br>USBSS IRQ_FRAME_THRESHOLD_RX0_2 Register |

**16.5.1.32 IRQFRAMETHOLDRX03 Register (offset = 21Ch) [reset = 0h]**

IRQFRAMETHOLDRX03 is shown in [Figure 16-53](#) and described in [Table 16-61](#).

**Figure 16-53. IRQFRAMETHOLDRX03 Register**

| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|--------------------|----|----|----|----|----|----|----|
| FRAME_THRES_RX1_15 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_14 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_13 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_12 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-61. IRQFRAMETHOLDRX03 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-24 | FRAME_THRES_RX1_15 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 15.   |
| 23-16 | FRAME_THRES_RX1_14 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 14.   |
| 15-8  | FRAME_THRES_RX1_13 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 13.   |
| 7-0   | FRAME_THRES_RX1_12 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB0 Endpoint 12.<br>USBSS IRQ_FRAME_THRESHOLD_RX0_3 Register |

**16.5.1.33 IRQFRAMETHOLDTX10 Register (offset = 220h) [reset = 0h]**

 IRQFRAMETHOLDTX10 is shown in [Figure 16-54](#) and described in [Table 16-62](#).

**Figure 16-54. IRQFRAMETHOLDTX10 Register**

|                   |    |    |    |    |    |    |    |
|-------------------|----|----|----|----|----|----|----|
| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_TX1_3 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_2 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_1 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-62. IRQFRAMETHOLDTX10 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_3 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 3. |
| 23-16 | FRAME_THRES_TX1_2 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 2. |
| 15-8  | FRAME_THRES_TX1_1 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 1. |

**16.5.1.34 IRQFRAMETHOLDTX11 Register (offset = 224h) [reset = 0h]**

 IRQFRAMETHOLDTX11 is shown in [Figure 16-55](#) and described in [Table 16-63](#).

**Figure 16-55. IRQFRAMETHOLDTX11 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_TX1_7 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_6 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_5 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_4 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-63. IRQFRAMETHOLDTX11 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_7 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 7.   |
| 23-16 | FRAME_THRES_TX1_6 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 6.   |
| 15-8  | FRAME_THRES_TX1_5 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 5.   |
| 7-0   | FRAME_THRES_TX1_4 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 4.<br>USBSS IRQ_FRAME_THRESHOLD_TX1_1 Register |

**16.5.1.35 IRQFRAMETHOLDTX12 Register (offset = 228h) [reset = 0h]**

 IRQFRAMETHOLDTX12 is shown in [Figure 16-56](#) and described in [Table 16-64](#).

**Figure 16-56. IRQFRAMETHOLDTX12 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_TX1_11 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_10 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_9  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_8  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-64. IRQFRAMETHOLDTX12 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-24 | FRAME_THRES_TX1_11 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 11.  |
| 23-16 | FRAME_THRES_TX1_10 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 10.  |
| 15-8  | FRAME_THRES_TX1_9  | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 9.   |
| 7-0   | FRAME_THRES_TX1_8  | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 8.<br>USBSS IRQ_FRAME_THRESHOLD_TX1_2 Register |

**16.5.1.36 IRQFRAMETHOLDTX13 Register (offset = 22Ch) [reset = 0h]**

IRQFRAMETHOLDTX13 is shown in [Figure 16-57](#) and described in [Table 16-65](#).

**Figure 16-57. IRQFRAMETHOLDTX13 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_TX1_15 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_TX1_14 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_TX1_13 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_TX1_12 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-65. IRQFRAMETHOLDTX13 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-24 | FRAME_THRES_TX1_15 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 15.   |
| 23-16 | FRAME_THRES_TX1_14 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 14.   |
| 15-8  | FRAME_THRES_TX1_13 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 13.   |
| 7-0   | FRAME_THRES_TX1_12 | R/W  | 0h    | FRAME threshold value for tx_pkt_cmp_0 for USB1 Endpoint 12.<br>USBSS IRQ_FRAME_THRESHOLD_TX1_3 Register |

**16.5.1.37 IRQFRAMETHOLDRX10 Register (offset = 230h) [reset = 0h]**

 IRQFRAMETHOLDRX10 is shown in [Figure 16-58](#) and described in [Table 16-66](#).

**Figure 16-58. IRQFRAMETHOLDRX10 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_RX1_3 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_2 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_1 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Reserved          |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-66. IRQFRAMETHOLDRX10 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_3 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 3. |
| 23-16 | FRAME_THRES_RX1_2 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 2. |
| 15-8  | FRAME_THRES_RX1_1 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 1. |

**16.5.1.38 IRQFRAMETHOLDRX11 Register (offset = 234h) [reset = 0h]**

 IRQFRAMETHOLDRX11 is shown in [Figure 16-59](#) and described in [Table 16-67](#).

**Figure 16-59. IRQFRAMETHOLDRX11 Register**

| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|-------------------|----|----|----|----|----|----|----|
| FRAME_THRES_RX1_7 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 23                | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_6 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 15                | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_5 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |
| 7                 | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_4 |    |    |    |    |    |    |    |
| R/W-0h            |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-67. IRQFRAMETHOLDRX11 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_7 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 7.   |
| 23-16 | FRAME_THRES_RX1_6 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 6.   |
| 15-8  | FRAME_THRES_RX1_5 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 5.   |
| 7-0   | FRAME_THRES_RX1_4 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 4.<br>USBSS IRQ_FRAME_THRESHOLD_RX1_1 Register |

**16.5.1.39 IRQFRAMETHOLDRX12 Register (offset = 238h) [reset = 0h]**

 IRQFRAMETHOLDRX12 is shown in [Figure 16-60](#) and described in [Table 16-68](#).

**Figure 16-60. IRQFRAMETHOLDRX12 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_RX1_11 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_10 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_9  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_8  |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-68. IRQFRAMETHOLDRX12 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31-24 | FRAME_THRES_RX1_11 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 11.  |
| 23-16 | FRAME_THRES_RX1_10 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 10.  |
| 15-8  | FRAME_THRES_RX1_9  | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 9.   |
| 7-0   | FRAME_THRES_RX1_8  | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 8.<br>USBSS IRQ_FRAME_THRESHOLD_RX1_2 Register |

**16.5.1.40 IRQFRAMETHOLDRX13 Register (offset = 23Ch) [reset = 0h]**

IRQFRAMETHOLDRX13 is shown in [Figure 16-61](#) and described in [Table 16-69](#).

**Figure 16-61. IRQFRAMETHOLDRX13 Register**

|                    |    |    |    |    |    |    |    |
|--------------------|----|----|----|----|----|----|----|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| FRAME_THRES_RX1_15 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 23                 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FRAME_THRES_RX1_14 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 15                 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| FRAME_THRES_RX1_13 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |
| 7                  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FRAME_THRES_RX1_12 |    |    |    |    |    |    |    |
| R/W-0h             |    |    |    |    |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-69. IRQFRAMETHOLDRX13 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-24 | FRAME_THRES_RX1_15 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 15.   |
| 23-16 | FRAME_THRES_RX1_14 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 14.   |
| 15-8  | FRAME_THRES_RX1_13 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 13.   |
| 7-0   | FRAME_THRES_RX1_12 | R/W  | 0h    | FRAME threshold value for rx_pkt_cmp_0 for USB1 Endpoint 12.<br>USBSS IRQ_FRAME_THRESHOLD_RX1_3 Register |

**16.5.1.41 IRQFRAMEENABLE0 Register (offset = 240h) [reset = 0h]**

 IRQFRAMEENABLE0 is shown in [Figure 16-62](#) and described in [Table 16-70](#).

**Figure 16-62. IRQFRAMEENABLE0 Register**

| 31              | 30 | 29       | 28 | 27             | 26 | 25       | 24 |
|-----------------|----|----------|----|----------------|----|----------|----|
| FRAME_EN_RX0_15 |    |          |    | Reserved       |    |          |    |
| R/W-0h          |    |          |    |                |    |          |    |
| 23              | 22 | 21       | 20 | 19             | 18 | 17       | 16 |
|                 |    | Reserved |    | FRAME_EN_RX0_1 |    | Reserved |    |
| R/W-0h          |    |          |    |                |    |          |    |
| 15              | 14 | 13       | 12 | 11             | 10 | 9        | 8  |
| FRAME_EN_TX0_15 |    |          |    | Reserved       |    |          |    |
| R/W-0h          |    |          |    |                |    |          |    |
| 7               | 6  | 5        | 4  | 3              | 2  | 1        | 0  |
|                 |    | Reserved |    | FRAME_EN_TX0_1 |    | Reserved |    |
| R/W-0h          |    |          |    |                |    |          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-70. IRQFRAMEENABLE0 Register Field Descriptions**

| Bit | Field           | Type | Reset | Description   |
|-----|-----------------|------|-------|---|
| 31  | FRAME_EN_RX0_15 | R/W  | 0h    | FRAME threshold enable value for rx_pkt_cmp_0 for USB0 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 17  | FRAME_EN_RX0_1  | R/W  | 0h    | FRAME threshold enable value for rx_pkt_cmp_0 for USB0 Endpoint 1.  |
| 15  | FRAME_EN_TX0_15 | R/W  | 0h    | FRAME threshold enable value for tx_pkt_cmp_0 for USB0 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 1   | FRAME_EN_TX0_1  | R/W  | 0h    | FRAME threshold enable value for tx_pkt_cmp_0 for USB0 Endpoint 1.  |

### 16.5.1.42 IRQFRAMEENABLE1 Register (offset = 244h) [reset = 0h]

IRQFRAMEENABLE1 is shown in [Figure 16-63](#) and described in [Table 16-71](#).

**Figure 16-63. IRQFRAMEENABLE1 Register**

| 31              | 30 | 29 | 28 | 27       | 26             | 25       | 24 |
|-----------------|----|----|----|----------|----------------|----------|----|
| FRAME_EN_RX1_15 |    |    |    | Reserved |                |          |    |
| R/W-0h          |    |    |    |          |                |          |    |
| 23              | 22 | 21 | 20 | 19       | 18             | 17       | 16 |
| Reserved        |    |    |    |          | FRAME_EN_RX1_1 | Reserved |    |
| R/W-0h          |    |    |    |          |                |          |    |
| 15              | 14 | 13 | 12 | 11       | 10             | 9        | 8  |
| FRAME_EN_TX1_15 |    |    |    | Reserved |                |          |    |
| R/W-0h          |    |    |    |          |                |          |    |
| 7               | 6  | 5  | 4  | 3        | 2              | 1        | 0  |
| Reserved        |    |    |    |          | FRAME_EN_TX1_1 | Reserved |    |
| R/W-0h          |    |    |    |          |                |          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-71. IRQFRAMEENABLE1 Register Field Descriptions**

| Bit | Field           | Type | Reset | Description   |
|-----|-----------------|------|-------|---|
| 31  | FRAME_EN_RX1_15 | R/W  | 0h    | FRAME threshold enable value for rx_pkt_cmp_1 for USB1 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 17  | FRAME_EN_RX1_1  | R/W  | 0h    | FRAME threshold enable value for rx_pkt_cmp_1 for USB1 Endpoint 1.  |
| 15  | FRAME_EN_TX1_15 | R/W  | 0h    | FRAME threshold enable value for tx_pkt_cmp_1 for USB1 Endpoint 15.<br>...<br>...<br>...<br>...<br>...<br>... |
| 1   | FRAME_EN_TX1_1  | R/W  | 0h    | FRAME threshold enable value for tx_pkt_cmp_1 for USB1 Endpoint 1.  |

### 16.5.2 USB0\_CTRL Registers

[Table 16-72](#) lists the memory-mapped registers for the USB0\_CTRL. All register offset addresses not listed in [Table 16-72](#) should be considered as reserved locations and the register contents should not be modified.

**Table 16-72. USB0\_CTRL REGISTERS**

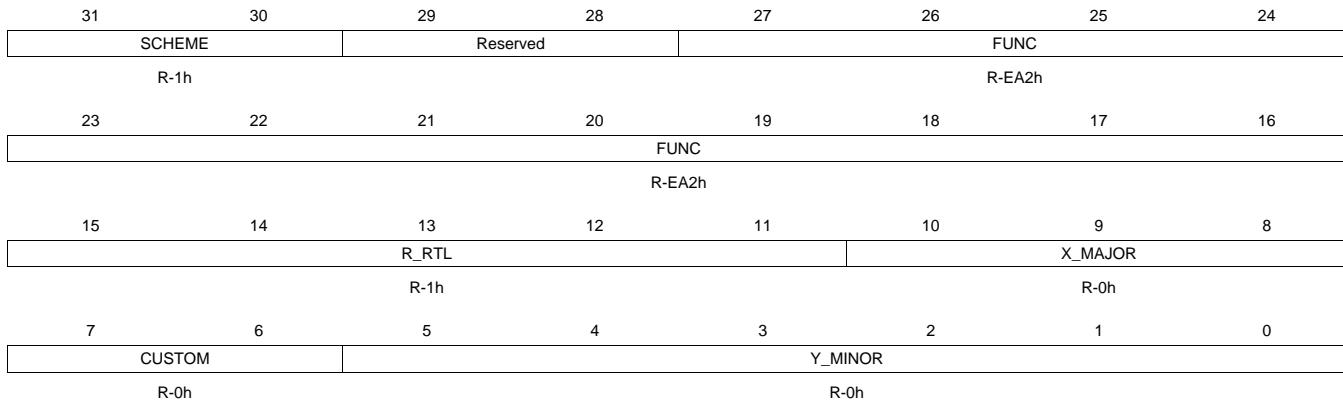
| Offset | Acronym         | Register Name | Section                          |
|--------|-----------------|---------------|----------------------------------|
| 0h     | USB0REV         |               | <a href="#">Section 16.5.2.1</a> |
| 14h    | USB0CTRL        |               | <a href="#">Section 16.5.2.2</a> |
| 18h    | USB0STAT        |               | <a href="#">Section 16.5.2.3</a> |
| 20h    | USB0IRQMSTAT    |               | <a href="#">Section 16.5.2.4</a> |
| 28h    | USB0IRQSTATRAW0 |               | <a href="#">Section 16.5.2.5</a> |
| 2Ch    | USB0IRQSTATRAW1 |               | <a href="#">Section 16.5.2.6</a> |
| 30h    | USB0IRQSTAT0    |               | <a href="#">Section 16.5.2.7</a> |
| 34h    | USB0IRQSTAT1    |               | <a href="#">Section 16.5.2.8</a> |

**Table 16-72. USB0\_CTRL REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b>    | <b>Register Name</b> | <b>Section</b>    |
|---------------|-------------------|----------------------|-------------------|
| 38h           | USB0IRQENABLESET0 |                      | Section 16.5.2.9  |
| 3Ch           | USB0IRQENABLESET1 |                      | Section 16.5.2.10 |
| 40h           | USB0IRQENABLECLR0 |                      | Section 16.5.2.11 |
| 44h           | USB0IRQENABLECLR1 |                      | Section 16.5.2.12 |
| 70h           | USB0TXMODE        |                      | Section 16.5.2.13 |
| 74h           | USB0RXMODE        |                      | Section 16.5.2.14 |
| 80h           | USB0GENRNDISEP1   |                      | Section 16.5.2.15 |
| 84h           | USB0GENRNDISEP2   |                      | Section 16.5.2.16 |
| 88h           | USB0GENRNDISEP3   |                      | Section 16.5.2.17 |
| 8Ch           | USB0GENRNDISEP4   |                      | Section 16.5.2.18 |
| 90h           | USB0GENRNDISEP5   |                      | Section 16.5.2.19 |
| 94h           | USB0GENRNDISEP6   |                      | Section 16.5.2.20 |
| 98h           | USB0GENRNDISEP7   |                      | Section 16.5.2.21 |
| 9Ch           | USB0GENRNDISEP8   |                      | Section 16.5.2.22 |
| A0h           | USB0GENRNDISEP9   |                      | Section 16.5.2.23 |
| A4h           | USB0GENRNDISEP10  |                      | Section 16.5.2.24 |
| A8h           | USB0GENRNDISEP11  |                      | Section 16.5.2.25 |
| ACh           | USB0GENRNDISEP12  |                      | Section 16.5.2.26 |
| B0h           | USB0GENRNDISEP13  |                      | Section 16.5.2.27 |
| B4h           | USB0GENRNDISEP14  |                      | Section 16.5.2.28 |
| B8h           | USB0GENRNDISEP15  |                      | Section 16.5.2.29 |
| D0h           | USB0AUTOREQ       |                      | Section 16.5.2.30 |
| D4h           | USB0SRPFIXTIME    |                      | Section 16.5.2.31 |
| D8h           | USB0_TDOWN        |                      | Section 16.5.2.32 |
| E0h           | USB0UTMI          |                      | Section 16.5.2.33 |
| E4h           | USB0MGCUTMILB     |                      | Section 16.5.2.34 |
| E8h           | USB0MODE          |                      | Section 16.5.2.35 |

**16.5.2.1 USB0REV Register (offset = 0h) [reset = 4EA20800h]**

USB0REV is shown in [Figure 16-64](#) and described in [Table 16-73](#).

**Figure 16-64. USB0REV Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-73. USB0REV Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description   |
|-------|---------|------|-------|---|
| 31-30 | SCHEME  | R    | 1h    | Used to distinguish between legacy interface scheme and current.<br>0 = Legacy<br>1 = Current |
| 27-16 | FUNC    | R    | EA2h  | Function indicates a software compatible module family.                                       |
| 15-11 | R RTL   | R    | 1h    | RTL revision.<br>Will vary depending on release.  |
| 10-8  | X_MAJOR | R    | 0h    | Major revision.   |
| 7-6   | CUSTOM  | R    | 0h    | Custom revision   |
| 5-0   | Y_MINOR | R    | 0h    | Minor revision.<br>USB0 Revision Register   |

### 16.5.2.2 USB0CTRL Register (offset = 14h) [reset = 0h]

USB0CTRL is shown in [Figure 16-65](#) and described in [Table 16-74](#).

**Figure 16-65. USB0CTRL Register**

| 31       | 30       | 29                   | 28     | 27       | 26       | 25      | 24         |
|----------|----------|----------------------|--------|----------|----------|---------|------------|
| DIS_DEB  | DIS_SRTP |                      |        | Reserved |          |         |            |
| R/W-0h   | R/W-0h   |                      |        |          |          |         |            |
| 23       | 22       | 21                   | 20     | 19       | 18       | 17      | 16         |
|          |          |                      |        | Reserved |          |         |            |
| 15       | 14       | 13                   | 12     | 11       | 10       | 9       | 8          |
|          |          |                      |        | Reserved |          |         |            |
| 7        | 6        | 5                    | 4      | 3        | 2        | 1       | 0          |
| Reserved |          | SOFT_RESET_ISOLATION | RNDIS  | UINT     | Reserved | CLKFACK | SOFT_RESET |
|          |          | R/W-0h               | R/W-0h | R/W-0h   |          | R/W-0h  | R/W-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-74. USB0CTRL Register Field Descriptions**

| Bit | Field                | Type | Reset | Description  |
|-----|----------------------|------|-------|--|
| 31  | DIS_DEB              | R/W  | 0h    | Disable the VBUS debouncer circuit fix   |
| 30  | DIS_SRTP             | R/W  | 0h    | Disable the OTG Session Request Protocol (SRP) AVALID circuit fix. When enabled (=0) this allows additional time for the VBUS signal to be measured against the VBUS thresholds. The time is specified in the USB0 SRP Fix Time Register.  |
| 5   | SOFT_RESET_ISOLATION | R/W  | 0h    | Soft reset isolation. When high this bit forces all USB0 signals that connect to the USBSS to known values during a soft reset via bit 0 of this register. This bit should be set high prior to setting bit 0 and cleared after bit 0 is cleared.  |
| 4   | RNDIS                | R/W  | 0h    | Global RNDIS mode enable for all endpoints.  |
| 3   | UINT                 | R/W  | 0h    | USB interrupt enable<br>1 = Legacy<br>0 = Current (recommended setting)<br>If uint is set high, then the mentor controller generic interrupt for USB[9] will be generated (if enabled). This requires S/W to read the mentor controller's registers to determine which interrupt from USB[0] to USB[7] that occurred. If uint is set low, then the usb20otg_f module will automatically read the mentor controller's registers and set the appropriate interrupt from USB[0] to USB[7] (if enabled). The generic interrupt for USB[9] will not be generated. |
| 1   | CLKFACK              | R/W  | 0h    | Clock stop fast ack enable.  |

**Table 16-74. USB0CTRL Register Field Descriptions (continued)**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 0   | SOFT_RESET | R/W  | 0h    | <p>Software reset of USB0.<br/>         Write 0 = no action<br/>         Write 1 = Initiate software reset<br/>         Read 0 = Reset done, no action<br/>         Read 1 = Reset ongoing<br/>         Both the soft_reset and soft_reset_isolation bits should be asserted simultaneously.<br/>         This will cause the following sequence of actions to occur over multiple cycles:<br/>         - All USB0 output signals will go to a known constant value via multiplexers.<br/>         This removes the possibility of timing errors due to the asynchronous resets.<br/>         - All USB0 registers will be reset.<br/>         - The USB0 resets will be de-asserted.<br/>         - The reset isolation multiplexer inputs will be de-selected.<br/>         - Both the soft_reset and soft_reset_isolation bits will be automatically cleared.<br/>         Setting only the soft_reset_isolation bit will cause all USB0 output signals to go to a known constant value via multiplexers.<br/>         This will prevent future access to USB0.<br/>         To clear this condition the USBSS must be reset via a hard or soft reset.</p> |

### 16.5.2.3 USB0STAT Register (offset = 18h) [reset = 0h]

USB0STAT is shown in Figure 16-66 and described in Table 16-75.

**Figure 16-66. USB0STAT Register**

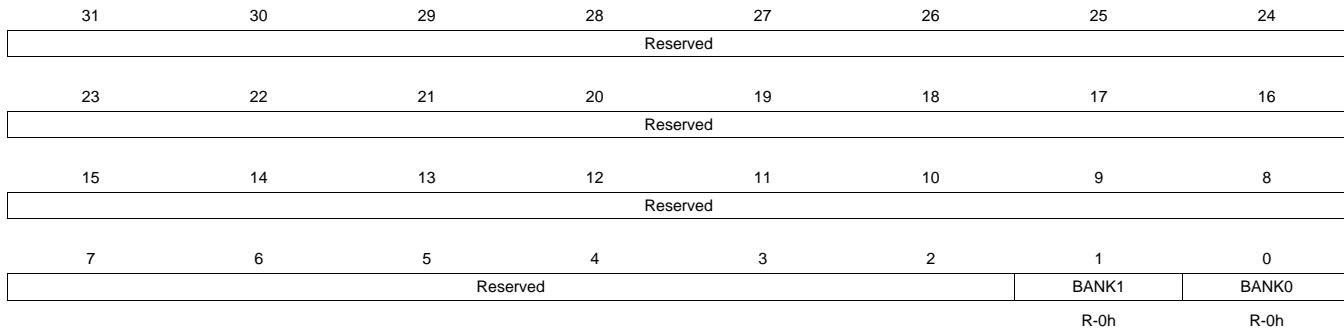
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-75. USB0STAT Register Field Descriptions**

| Bit | Field   | Type | Reset | Description                                |
|-----|---------|------|-------|--|
| 0   | DRVVBUS | R    | 0h    | Current DRVVBUS value USB0 Status Register |

**16.5.2.4 USB0IRQMSTAT Register (offset = 20h) [reset = 0h]**

USB0IRQMSTAT is shown in [Figure 16-67](#) and described in [Table 16-76](#).

**Figure 16-67. USB0IRQMSTAT Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-76. USB0IRQMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | BANK1 | R    | 0h    | 0: No events pending from IRQ_STATUS_1<br>1: At least one event is pending from IRQ_STATUS_1                                 |
| 0   | BANK0 | R    | 0h    | 0: No events pending from IRQ_STATUS_0<br>1: At least one event is pending from IRQ_STATUS_0 USB0 IRQ_MERGED_STATUS Register |

**16.5.2.5 USB0IRQSTATRAW0 Register (offset = 28h) [reset = 0h]**

 USB0IRQSTATRAW0 is shown in [Figure 16-68](#) and described in [Table 16-77](#).

**Figure 16-68. USB0IRQSTATRAW0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  |          |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-77. USB0IRQSTATRAW0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt status for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt status for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt status for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt status for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt status for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt status for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt status for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt status for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt status for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt status for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt status for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt status for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt status for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt status for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt status for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt status for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt status for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt status for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt status for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt status for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt status for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt status for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt status for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt status for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt status for TX endpoint 6  |

**Table 16-77. USB0IRQSTATRAW0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt status for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt status for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt status for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt status for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt status for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt status for TX endpoint 0 USB0 IRQ_STATUS_RAW_0 Register |

**16.5.2.6 USB0IRQSTATRAW1 Register (offset = 2Ch) [reset = 0h]**

 USB0IRQSTATRAW1 is shown in [Figure 16-69](#) and described in [Table 16-78](#).

**Figure 16-69. USB0IRQSTATRAW1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | TX_FIFO_0 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-78. USB0IRQSTATRAW1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 15  |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 14  |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 13  |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 12  |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 11  |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 10  |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 9   |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 8   |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 7   |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 6   |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 5   |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 4   |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 3   |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 2   |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 1   |
| 16  | TX_FIFO_0  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 0   |
| 9   | USB_9      | R/W  | 0h    | Interrupt status for Mentor controller USB_INT generic interrupt                                |
| 8   | USB_8      | R/W  | 0h    | Interrupt status for DRVVBUS level change   |
| 7   | USB_7      | R/W  | 0h    | Interrupt status for VBUS andlt;br/>VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt status for SRP detected   |
| 5   | USB_5      | R/W  | 0h    | Interrupt status for device disconnected (host mode)  |
| 4   | USB_4      | R/W  | 0h    | Interrupt status for device connected (host mode)   |
| 3   | USB_3      | R/W  | 0h    | Interrupt status for SOF started  |
| 2   | USB_2      | R/W  | 0h    | Interrupt status for Reset signaling detected (peripheral mode)<br/>Babble detected (host mode) |

**Table 16-78. USB0IRQSTATRAW1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | USB_1 | R/W  | 0h    | Interrupt status for Resume signaling detected                                 |
| 0   | USB_0 | R/W  | 0h    | Interrupt status for Suspend signaling detected USB0 IRQ_STATUS_RAW_1 Register |

### 16.5.2.7 USB0IRQSTAT0 Register (offset = 30h) [reset = 0h]

USB0IRQSTAT0 is shown in Figure 16-70 and described in Table 16-79.

**Figure 16-70. USB0IRQSTAT0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-79. USB0IRQSTAT0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt status for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt status for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt status for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt status for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt status for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt status for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt status for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt status for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt status for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt status for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt status for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt status for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt status for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt status for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt status for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt status for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt status for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt status for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt status for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt status for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt status for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt status for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt status for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt status for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt status for TX endpoint 6  |

**Table 16-79. USB0IRQSTAT0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt status for TX endpoint 5                            |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt status for TX endpoint 4                            |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt status for TX endpoint 3                            |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt status for TX endpoint 2                            |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt status for TX endpoint 1                            |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt status for TX endpoint 0 USB0 IRQ_STATUS_0 Register |

### 16.5.2.8 USB0IRQSTAT1 Register (offset = 34h) [reset = 0h]

USB0IRQSTAT1 is shown in Figure 16-71 and described in Table 16-80.

**Figure 16-71. USB0IRQSTAT1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | TX_FIFO_0 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-80. USB0IRQSTAT1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 15   |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 14   |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 13   |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 12   |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 11   |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 10   |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 9  |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 8  |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 7  |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 6  |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 5  |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 4  |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 3  |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 2  |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 1  |
| 16  | TX_FIFO_0  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 0  |
| 9   | USB_9      | R/W  | 0h    | Interrupt status for Mentor controller USB_INT generic interrupt                               |
| 8   | USB_8      | R/W  | 0h    | Interrupt status for DRVVBUS level change  |
| 7   | USB_7      | R/W  | 0h    | Interrupt status for VBUS andlt; VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt status for SRP detected  |
| 5   | USB_5      | R/W  | 0h    | Interrupt status for device disconnected (host mode)   |
| 4   | USB_4      | R/W  | 0h    | Interrupt status for device connected (host mode)  |
| 3   | USB_3      | R/W  | 0h    | Interrupt status for SOF started   |
| 2   | USB_2      | R/W  | 0h    | Interrupt status for Reset signaling detected (peripheral mode)<br>Babble detected (host mode) |

**Table 16-80. USB0IRQSTAT1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | USB_1 | R/W  | 0h    | Interrupt status for Resume signaling detected                             |
| 0   | USB_0 | R/W  | 0h    | Interrupt status for Suspend signaling detected USB0 IRQ_STATUS_1 Register |

### 16.5.2.9 USB0IRQENABLESET0 Register (offset = 38h) [reset = 0h]

USB0IRQENABLESET0 is shown in Figure 16-72 and described in Table 16-81.

**Figure 16-72. USB0IRQENABLESET0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-81. USB0IRQENABLESET0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt enable for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt enable for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt enable for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt enable for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt enable for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt enable for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt enable for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt enable for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt enable for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt enable for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt enable for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt enable for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt enable for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt enable for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt enable for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt enable for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt enable for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt enable for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt enable for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt enable for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt enable for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt enable for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt enable for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt enable for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt enable for TX endpoint 6  |

**Table 16-81. USB0IRQENABLESET0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt enable for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt enable for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt enable for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt enable for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt enable for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt enable for TX endpoint 0 USB0 IRQ_ENABLE_SET_0 Register |

### 16.5.2.10 USB0IRQENABLESET1 Register (offset = 3Ch) [reset = 0h]

USB0IRQENABLESET1 is shown in [Figure 16-73](#) and described in [Table 16-82](#).

**Figure 16-73. USB0IRQENABLESET1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | TX_FIFO_0 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-82. USB0IRQENABLESET1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 15   |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 14   |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 13   |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 12   |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 11   |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 10   |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 9  |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 8  |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 7  |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 6  |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 5  |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 4  |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 3  |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 2  |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 1  |
| 16  | TX_FIFO_0  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 0  |
| 9   | USB_9      | R/W  | 0h    | Interrupt enable for Mentor controller USB_INT generic interrupt                               |
| 8   | USB_8      | R/W  | 0h    | Interrupt enable for DRVVBUS level change  |
| 7   | USB_7      | R/W  | 0h    | Interrupt enable for VBUS and/or VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt enable for SRP detected  |
| 5   | USB_5      | R/W  | 0h    | Interrupt enable for device disconnected (host mode)   |
| 4   | USB_4      | R/W  | 0h    | Interrupt enable for device connected (host mode)  |
| 3   | USB_3      | R/W  | 0h    | Interrupt enable for SOF started   |
| 2   | USB_2      | R/W  | 0h    | Interrupt enable for Reset signaling detected (peripheral mode)<br>Babble detected (host mode) |

**Table 16-82. USB0IRQENABLESET1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | USB_1 | R/W  | 0h    | Interrupt enable for Resume signaling detected                                 |
| 0   | USB_0 | R/W  | 0h    | Interrupt enable for Suspend signaling detected USB0 IRQ_ENABLE_SET_1 Register |

**16.5.2.11 USB0IRQENABLECLR0 Register (offset = 40h) [reset = 0h]**

 USB0IRQENABLECLR0 is shown in [Figure 16-74](#) and described in [Table 16-83](#).

**Figure 16-74. USB0IRQENABLECLR0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-83. USB0IRQENABLECLR0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt enable for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt enable for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt enable for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt enable for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt enable for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt enable for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt enable for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt enable for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt enable for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt enable for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt enable for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt enable for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt enable for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt enable for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt enable for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt enable for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt enable for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt enable for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt enable for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt enable for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt enable for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt enable for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt enable for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt enable for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt enable for TX endpoint 6  |

**Table 16-83. USB0IRQENABLECLR0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt enable for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt enable for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt enable for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt enable for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt enable for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt enable for TX endpoint 0 USB0 IRQ_ENABLE_CLR_0 Register |

**16.5.2.12 USB0IRQENABLECLR1 Register (offset = 44h) [reset = 0h]**

 USB0IRQENABLECLR1 is shown in [Figure 16-75](#) and described in [Table 16-84](#).

**Figure 16-75. USB0IRQENABLECLR1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | TX_FIFO_0 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-84. USB0IRQENABLECLR1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 15   |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 14   |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 13   |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 12   |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 11   |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 10   |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 9  |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 8  |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 7  |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 6  |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 5  |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 4  |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 3  |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 2  |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 1  |
| 16  | TX_FIFO_0  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 0  |
| 9   | USB_9      | R/W  | 0h    | Interrupt enable for Mentor controller USB_INT generic interrupt                               |
| 8   | USB_8      | R/W  | 0h    | Interrupt enable for DRVVBUS level change  |
| 7   | USB_7      | R/W  | 0h    | Interrupt enable for VBUS and/or VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt enable for SRP detected  |
| 5   | USB_5      | R/W  | 0h    | Interrupt enable for device disconnected (host mode)   |
| 4   | USB_4      | R/W  | 0h    | Interrupt enable for device connected (host mode)  |
| 3   | USB_3      | R/W  | 0h    | Interrupt enable for SOF started   |
| 2   | USB_2      | R/W  | 0h    | Interrupt enable for Reset signaling detected (peripheral mode)<br>Babble detected (host mode) |

**Table 16-84. USB0IRQENABLECLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | USB_1 | R/W  | 0h    | Interrupt enable for Resume signaling detected                                 |
| 0   | USB_0 | R/W  | 0h    | Interrupt enable for Suspend signaling detected USB0 IRQ_ENABLE_CLR_1 Register |

**16.5.2.13 USB0TXMODE Register (offset = 70h) [reset = 0h]**

 USB0TXMODE is shown in [Figure 16-76](#) and described in [Table 16-85](#).

**Figure 16-76. USB0TXMODE Register**

| 31        | 30 | 29        | 28     | 27        | 26     | 25        | 24     |
|-----------|----|-----------|--------|-----------|--------|-----------|--------|
| Reserved  |    | TX15_MODE |        | TX14_MODE |        | TX13_MODE |        |
|           |    | R/W-0h    |        |           | R/W-0h |           | R/W-0h |
| 23        | 22 | 21        | 20     | 19        | 18     | 17        | 16     |
| TX12_MODE |    | TX11_MODE |        | TX10_MODE |        | TX9_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 15        | 14 | 13        | 12     | 11        | 10     | 9         | 8      |
| TX8_MODE  |    | TX7_MODE  |        | TX6_MODE  |        | TX5_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 7         | 6  | 5         | 4      | 3         | 2      | 1         | 0      |
| TX4_MODE  |    | TX3_MODE  |        | TX2_MODE  |        | TX1_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-85. USB0TXMODE Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 29-28 | TX15_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 14<br>01: RNDIS Mode on TX endpoint 14<br>10: CDC Mode on TX endpoint 14<br>11: Generic RNDIS Mode on TX endpoint 14 |
| 27-26 | TX14_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 13<br>01: RNDIS Mode on TX endpoint 13<br>10: CDC Mode on TX endpoint 13<br>11: Generic RNDIS Mode on TX endpoint 13 |
| 25-24 | TX13_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 12<br>01: RNDIS Mode on TX endpoint 12<br>10: CDC Mode on TX endpoint 12<br>11: Generic RNDIS Mode on TX endpoint 12 |
| 23-22 | TX12_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 11<br>01: RNDIS Mode on TX endpoint 11<br>10: CDC Mode on TX endpoint 11<br>11: Generic RNDIS Mode on TX endpoint 11 |
| 21-20 | TX11_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 10<br>01: RNDIS Mode on TX endpoint 10<br>10: CDC Mode on TX endpoint 10<br>11: Generic RNDIS Mode on TX endpoint 10 |
| 19-18 | TX10_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 9<br>01: RNDIS Mode on TX endpoint 9<br>10: CDC Mode on TX endpoint 9<br>11: Generic RNDIS Mode on TX endpoint 9     |
| 17-16 | TX9_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 8<br>01: RNDIS Mode on TX endpoint 8<br>10: CDC Mode on TX endpoint 8<br>11: Generic RNDIS Mode on TX endpoint 8     |
| 15-14 | TX8_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 7<br>01: RNDIS Mode on TX endpoint 7<br>10: CDC Mode on TX endpoint 7<br>11: Generic RNDIS Mode on TX endpoint 7     |
| 13-12 | TX7_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 6<br>01: RNDIS Mode on TX endpoint 6<br>10: CDC Mode on TX endpoint 6<br>11: Generic RNDIS Mode on TX endpoint 6     |

**Table 16-85. USB0TXMODE Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 11-10 | TX6_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 5<br>01: RNDIS Mode on TX endpoint 5<br>10: CDC Mode on TX endpoint 5<br>11: Generic RNDIS Mode on TX endpoint 5                        |
| 9-8   | TX5_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 4<br>01: RNDIS Mode on TX endpoint 4<br>10: CDC Mode on TX endpoint 4<br>11: Generic RNDIS Mode on TX endpoint 4                        |
| 7-6   | TX4_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 3<br>01: RNDIS Mode on TX endpoint 3<br>10: CDC Mode on TX endpoint 3<br>11: Generic RNDIS Mode on TX endpoint 3                        |
| 5-4   | TX3_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 2<br>01: RNDIS Mode on TX endpoint 2<br>10: CDC Mode on TX endpoint 2<br>11: Generic RNDIS Mode on TX endpoint 2                        |
| 3-2   | TX2_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 1<br>01: RNDIS Mode on TX endpoint 1<br>10: CDC Mode on TX endpoint 1<br>11: Generic RNDIS Mode on TX endpoint 1                        |
| 1-0   | TX1_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 0<br>01: RNDIS Mode on TX endpoint 0<br>10: CDC Mode on TX endpoint 0<br>11: Generic RNDIS Mode on TX endpoint 0 USB0 Tx Mode Registers |

**16.5.2.14 USB0RXMODE Register (offset = 74h) [reset = 0h]**

 USB0RXMODE is shown in [Figure 16-77](#) and described in [Table 16-86](#).

**Figure 16-77. USB0RXMODE Register**

| 31        | 30 | 29        | 28     | 27        | 26     | 25        | 24     |
|-----------|----|-----------|--------|-----------|--------|-----------|--------|
| Reserved  |    | RX15_MODE |        | RX14_MODE |        | RX13_MODE |        |
|           |    | R/W-0h    |        |           | R/W-0h |           | R/W-0h |
| 23        | 22 | 21        | 20     | 19        | 18     | 17        | 16     |
| RX12_MODE |    | RX11_MODE |        | RX10_MODE |        | RX9_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 15        | 14 | 13        | 12     | 11        | 10     | 9         | 8      |
| RX8_MODE  |    | RX7_MODE  |        | RX6_MODE  |        | RX5_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 7         | 6  | 5         | 4      | 3         | 2      | 1         | 0      |
| RX4_MODE  |    | RX3_MODE  |        | RX2_MODE  |        | RX1_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-86. USB0RXMODE Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 29-28 | RX15_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 14<br>01: RNDIS Mode on RX endpoint 14<br>10: CDC Mode on RX endpoint 14<br>11: Generic RNDIS or Infinite Mode on RX endpoint 14 |
| 27-26 | RX14_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 13<br>01: RNDIS Mode on RX endpoint 13<br>10: CDC Mode on RX endpoint 13<br>11: Generic RNDIS or Infinite Mode on RX endpoint 13 |
| 25-24 | RX13_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 12<br>01: RNDIS Mode on RX endpoint 12<br>10: CDC Mode on RX endpoint 12<br>11: Generic RNDIS or Infinite Mode on RX endpoint 12 |
| 23-22 | RX12_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 11<br>01: RNDIS Mode on RX endpoint 11<br>10: CDC Mode on RX endpoint 11<br>11: Generic RNDIS or Infinite Mode on RX endpoint 11 |
| 21-20 | RX11_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 10<br>01: RNDIS Mode on RX endpoint 10<br>10: CDC Mode on RX endpoint 10<br>11: Generic RNDIS or Infinite Mode on RX endpoint 10 |
| 19-18 | RX10_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 9<br>01: RNDIS Mode on RX endpoint 9<br>10: CDC Mode on RX endpoint 9<br>11: Generic RNDIS or Infinite Mode on RX endpoint 9     |
| 17-16 | RX9_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 8<br>01: RNDIS Mode on RX endpoint 8<br>10: CDC Mode on RX endpoint 8<br>11: Generic RNDIS Mode on RX endpoint 8                 |
| 15-14 | RX8_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 7<br>01: RNDIS Mode on RX endpoint 7<br>10: CDC Mode on RX endpoint 7<br>11: Generic RNDIS or Infinite Mode on RX endpoint 7     |
| 13-12 | RX7_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 6<br>01: RNDIS Mode on RX endpoint 6<br>10: CDC Mode on RX endpoint 6<br>11: Generic RNDIS or Infinite Mode on RX endpoint 6     |

**Table 16-86. USB0RXMODE Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 11-10 | RX6_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 5<br>01: RNDIS Mode on RX endpoint 5<br>10: CDC Mode on RX endpoint 5<br>11: Generic RNDIS or Infinite Mode on RX endpoint 5 |
| 9-8   | RX5_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 4<br>01: RNDIS Mode on RX endpoint 4<br>10: CDC Mode on RX endpoint 4<br>11: Generic RNDIS or Infinite Mode on RX endpoint 4 |
| 7-6   | RX4_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 3<br>01: RNDIS Mode on RX endpoint 3<br>10: CDC Mode on RX endpoint 3<br>11: Generic RNDIS or Infinite Mode on RX endpoint 3 |
| 5-4   | RX3_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 2<br>01: RNDIS Mode on RX endpoint 2<br>10: CDC Mode on RX endpoint 2<br>11: Generic RNDIS or Infinite Mode on RX endpoint 2 |
| 3-2   | RX2_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 1<br>01: RNDIS Mode on RX endpoint 1<br>10: CDC Mode on RX endpoint 1<br>11: Generic RNDIS or Infinite Mode on RX endpoint 1 |

**Table 16-86. USB0RXMODE Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description   |
|-----|----------|------|-------|---|
| 1-0 | RX1_MODE | R/W  | 0h    | <p>00: Transparent Mode on RX endpoint 0</p> <p>01: RNDIS Mode on RX endpoint 0</p> <p>10: CDC Mode on RX endpoint 0</p> <p>11: Generic RNDIS or Infinite Mode on RX endpoint 0 Each RX endpoint can be configured into 1 of 5 packet termination modes.</p> <p>Transparent mode (b00) * Supports USB endpoint sizes of o FS : 8, 16, 32, 64, and 1023 o HS : 64, 128, 512, and 1024 * Mentor Controller's RXMAXP/TXMAXP must be a valid USB endpoint size.</p> <p>* With autoreq=3 XDMA will always generate a ReqPkt's at the end of the packet.</p> <p>* With autoreq=1 XDMA will never generatate a ReqPkt at the end of the packet.</p> <p>* Each USB packet is equivalent to a single CPPI DMA packet.</p> <p>* Each received CPPI packet will be no larger than the USB max packet size (1023 bytes for FS, 1024 bytes for HS) and will be both a SOP and EOP.</p> <p>* Transmitted CPPI packets can be no larger than the USB max packet size.</p> <p>* Primarily used for interrupt or isochronous endpoints, as CPU will receive an interrupt for every USB packet (if enabled).</p> <p>RNDIS mode (b01) * Supports USB endpoint sizes of o FS : 64 o HS : 64, 128, 512, and 1024 * Mentor controller's RXMAXP/TXMAXP must be an integer multiple of the USB endpoint size.</p> <p>* With autoreq=3 XDMA will always generate a ReqPkt's at the end of the packet.</p> <p>* With autoreq=1 XDMA will continue the generation of ReqPkt's to the Mentor controller until a short packet is received.</p> <p>* Supports transmission of CPPI packets larger than the USB max packet size.</p> <p>* The end of a CPPI packet is defined by a USB short packet (a USB packet less than the USB max packet size).</p> <p>* In the case where the CPPI packet is an exact multiple of the USB max packet size, a zero length terminating packet is used (XDMA recognizes this terminating packet on reception, and generates it on transmission).</p> <p>* Designed for use in an RNDIS compliant networking type of USB device and bulk endpoints Linux CDC mode (b10) * Same as RNDIS mode, except terminating packet has 1-byte of 0 data.</p> <p>* Designed for use with a Linux OS and USB driver stack and bulk endpoints.</p> <p>Generic RNDIS mode (b11 and Generic RNDIS EP N Size register is greater than 0) * Same as RNDIS mode, except the end of a CPPI packet is determined by a USB short packet or when the value in the GENERIC RNDIS EPn SIZE Register is reached (Rx mode only)(i.e. Tx mode does not use this register).</p> <p>The CPU configures this register with a value that is a multiple of the corresponding endpoint size.</p> <p>* No terminating packet is used.</p> <p>* Designed for general bulk endpoint use where specific packet terminations are not required.</p> <p>* Transmit mode transfers only support FS or HS EP sizes through the XDMA and Mentor controller.</p> <p>The CPPI DMA size can be of any legal size as defined by the CPPI 4.1 specification.</p> <p>Infinite mode (b11 and Generic RNDIS EP N Size register equal to 0) * Same as RNDIS mode, except the end of a CPPI packet is determined by a USB short packet or when the CPPI DMA closes up the packet after a defined number of buffers have been filled.</p> <p>This is defined in Rx Channel N Global Configuration Register bits 28:26 rx_max_buffer_cnt.</p> <p>* The CPPI DMA will ignore SOPs generated by the XDMA.</p> <p>* The XDMA packet is assumed to be infinite.</p> <p>* The XDMA starts the packet with a SOP (ignored by the CPPI DMA).</p> <p>* The XDMA ends the packet when a short packet occurs.</p> <p>* Short packets are defined when RXMAXP (0x14) is not equal to RXCOUNT (0x18).</p> |

**Table 16-86. USB0RXMODE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
|     |       |      |       | <p>RXMAXP and RXCOUNT are both registers within the Mentor controller.</p> <ul style="list-style-type: none"> <li>* The XDMA will continue to request RX packets from the Mentor controller until a short packet is received.</li> <li>* When the CPPI is configured for chaining mode, the Buffer Descriptor sizes must be integer multiples of the RXMAXP size.</li> <li>* Transmit mode is not supported.</li> <li>* USB0 or USB1 Auto Req register should be set to AUTO REQ ALWAYS.</li> </ul> |

**16.5.2.15 USB0GENRNDISEP1 Register (offset = 80h) [reset = 0h]**

USB0GENRNDISEP1 is shown in [Figure 16-78](#) and described in [Table 16-87](#).

**Figure 16-78. USB0GENRNDISEP1 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP1_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-87. USB0GENRNDISEP1 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP1_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.16 USB0GENRNDISEP2 Register (offset = 84h) [reset = 0h]**

USB0GENRNDISEP2 is shown in [Figure 16-79](#) and described in [Table 16-88](#).

**Figure 16-79. USB0GENRNDISEP2 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP2_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-88. USB0GENRNDISEP2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP2_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.17 USB0GENRNDISEP3 Register (offset = 88h) [reset = 0h]**

USB0GENRNDISEP3 is shown in [Figure 16-80](#) and described in [Table 16-89](#).

**Figure 16-80. USB0GENRNDISEP3 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP3_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-89. USB0GENRNDISEP3 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP3_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.18 USB0GENRNDISEP4 Register (offset = 8Ch) [reset = 0h]**

USB0GENRNDISEP4 is shown in [Figure 16-81](#) and described in [Table 16-90](#).

**Figure 16-81. USB0GENRNDISEP4 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16       | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP4_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-90. USB0GENRNDISEP4 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP4_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.19 USB0GENRNDISEP5 Register (offset = 90h) [reset = 0h]**

USB0GENRNDISEP5 is shown in [Figure 16-82](#) and described in [Table 16-91](#).

**Figure 16-82. USB0GENRNDISEP5 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP5_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-91. USB0GENRNDISEP5 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP5_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.20 USB0GENRNDISEP6 Register (offset = 94h) [reset = 0h]**

USB0GENRNDISEP6 is shown in [Figure 16-83](#) and described in [Table 16-92](#).

**Figure 16-83. USB0GENRNDISEP6 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP6_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-92. USB0GENRNDISEP6 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP6_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.21 USB0GENRNDISEP7 Register (offset = 98h) [reset = 0h]**

USB0GENRNDISEP7 is shown in [Figure 16-84](#) and described in [Table 16-93](#).

**Figure 16-84. USB0GENRNDISEP7 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP7_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-93. USB0GENRNDISEP7 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP7_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.22 USB0GENRNDISEP8 Register (offset = 9Ch) [reset = 0h]**

USB0GENRNDISEP8 is shown in [Figure 16-85](#) and described in [Table 16-94](#).

**Figure 16-85. USB0GENRNDISEP8 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP8_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-94. USB0GENRNDISEP8 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP8_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.23 USB0GENRNDISEP9 Register (offset = A0h) [reset = 0h]**

USB0GENRNDISEP9 is shown in [Figure 16-86](#) and described in [Table 16-95](#).

**Figure 16-86. USB0GENRNDISEP9 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP9_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-95. USB0GENRNDISEP9 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP9_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.24 USB0GENRNDISEP10 Register (offset = A4h) [reset = 0h]**

USB0GENRNDISEP10 is shown in [Figure 16-87](#) and described in [Table 16-96](#).

**Figure 16-87. USB0GENRNDISEP10 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP10_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-96. USB0GENRNDISEP10 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP10_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.25 USB0GENRNDISEP11 Register (offset = A8h) [reset = 0h]**

USB0GENRNDISEP11 is shown in [Figure 16-88](#) and described in [Table 16-97](#).

**Figure 16-88. USB0GENRNDISEP11 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP11_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-97. USB0GENRNDISEP11 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP11_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.26 USB0GENRNDISEP12 Register (offset = ACh) [reset = 0h]**

USB0GENRNDISEP12 is shown in [Figure 16-89](#) and described in [Table 16-98](#).

**Figure 16-89. USB0GENRNDISEP12 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16        | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP12_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R/W-0h    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-98. USB0GENRNDISEP12 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP12_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.27 USB0GENRNDISEP13 Register (offset = B0h) [reset = 0h]**

USB0GENRNDISEP13 is shown in [Figure 16-90](#) and described in [Table 16-99](#).

**Figure 16-90. USB0GENRNDISEP13 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP13_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-99. USB0GENRNDISEP13 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP13_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.28 USB0GENRNDISEP14 Register (offset = B4h) [reset = 0h]**

USB0GENRNDISEP14 is shown in [Figure 16-91](#) and described in [Table 16-100](#).

**Figure 16-91. USB0GENRNDISEP14 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP14_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-100. USB0GENRNDISEP14 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP14_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.29 USB0GENRNDISEP15 Register (offset = B8h) [reset = 0h]**

USB0GENRNDISEP15 is shown in [Figure 16-92](#) and described in [Table 16-101](#).

**Figure 16-92. USB0GENRNDISEP15 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP15_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-101. USB0GENRNDISEP15 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP15_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB0 Generic RNDIS EP N Size Register |

**16.5.2.30 USB0AUTOREQ Register (offset = D0h) [reset = 0h]**

 USB0AUTOREQ is shown in [Figure 16-93](#) and described in [Table 16-102](#).

**Figure 16-93. USB0AUTOREQ Register**

| 31           | 30 | 29           | 28     | 27           | 26     | 25           | 24     |
|--------------|----|--------------|--------|--------------|--------|--------------|--------|
| Reserved     |    | RX15_AUTOREQ |        | RX14_AUTOREQ |        | RX13_AUTOREQ |        |
|              |    | R/W-0h       |        |              | R/W-0h |              | R/W-0h |
| 23           | 22 | 21           | 20     | 19           | 18     | 17           | 16     |
| RX12_AUTOREQ |    | RX11_AUTOREQ |        | RX10_AUTOREQ |        | RX9_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |
| 15           | 14 | 13           | 12     | 11           | 10     | 9            | 8      |
| RX8_AUTOREQ  |    | RX7_AUTOREQ  |        | RX6_AUTOREQ  |        | RX5_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |
| 7            | 6  | 5            | 4      | 3            | 2      | 1            | 0      |
| RX4_AUTOREQ  |    | RX3_AUTOREQ  |        | RX2_AUTOREQ  |        | RX1_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-102. USB0AUTOREQ Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 29-28 | RX15_AUTOREQ | R/W  | 0h    | RX endpoint 14 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 27-26 | RX14_AUTOREQ | R/W  | 0h    | RX endpoint 13 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 25-24 | RX13_AUTOREQ | R/W  | 0h    | RX endpoint 12 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 23-22 | RX12_AUTOREQ | R/W  | 0h    | RX endpoint 11 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 21-20 | RX11_AUTOREQ | R/W  | 0h    | RX endpoint 10 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 19-18 | RX10_AUTOREQ | R/W  | 0h    | RX endpoint 9 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |
| 17-16 | RX9_AUTOREQ  | R/W  | 0h    | RX endpoint 8 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |
| 15-14 | RX8_AUTOREQ  | R/W  | 0h    | RX endpoint 7 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |

**Table 16-102. USB0AUTOREQ Register Field Descriptions (continued)**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 13-12 | RX7_AUTOREQ | R/W  | 0h    | RX endpoint 6 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 11-10 | RX6_AUTOREQ | R/W  | 0h    | RX endpoint 5 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 9-8   | RX5_AUTOREQ | R/W  | 0h    | RX endpoint 4 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 7-6   | RX4_AUTOREQ | R/W  | 0h    | RX endpoint 3 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 5-4   | RX3_AUTOREQ | R/W  | 0h    | RX endpoint 2 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 3-2   | RX2_AUTOREQ | R/W  | 0h    | RX endpoint 1 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 1-0   | RX1_AUTOREQ | R/W  | 0h    | RX endpoint 0 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always USB0 Auto Req Registers |

**16.5.2.31 USB0SRPFIXTIME Register (offset = D4h) [reset = 280DE80h]**

USB0SRPFIXTIME is shown in [Figure 16-94](#) and described in [Table 16-103](#).

**Figure 16-94. USB0SRPFIXTIME Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| SRPFIXTIME   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-280DE80h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

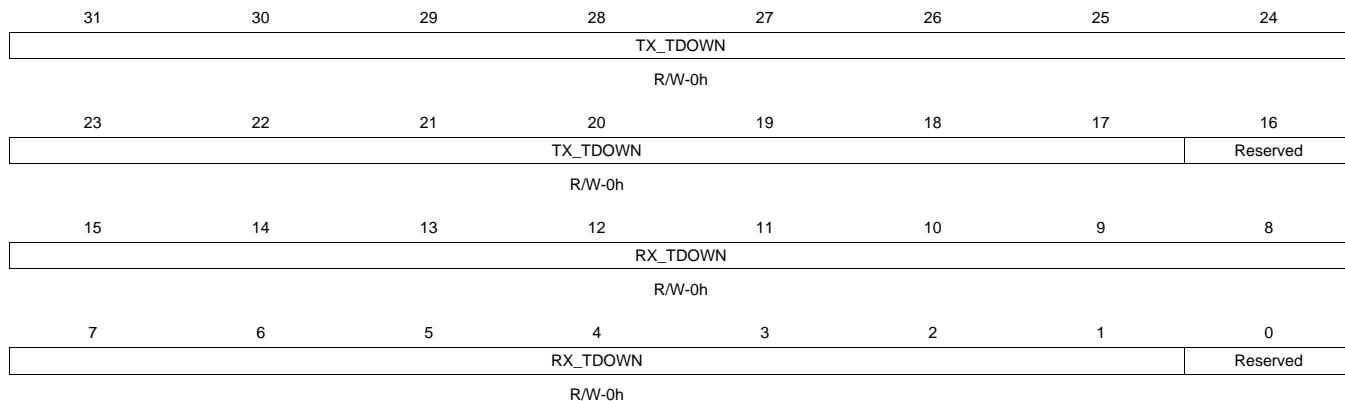
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-103. USB0SRPFIXTIME Register Field Descriptions**

| Bit  | Field      | Type | Reset    | Description  |
|------|------------|------|----------|--|
| 31-0 | SRPFIXTIME | R/W  | 280DE80h | SRP Fix maximum time in 60 MHz cycles.<br>Default is 700 ms.<br>USB0 SRP Fix Time Register |

**16.5.2.32 USB0\_TDOWN Register (offset = D8h) [reset = 0h]**

USB0\_TDOWN is shown in [Figure 16-95](#) and described in [Table 16-104](#).

**Figure 16-95. USB0\_TDOWN Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-104. USB0\_TDOWN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-17 | TX_TDOWN | R/W  | 0h    | Tx Endpoint Teardown.<br>Write '1' to corresponding bit to set.<br>Read as '0'.<br>Bit<br>31 = Endpoint 15 ...<br>Bit<br>17 = Endpoint 1 |
| 15-1  | RX_TDOWN | R/W  | 0h    | RX Endpoint Teardown.<br>Write '1' to corresponding bit to set.<br>Read as '0'.<br>Bit<br>15 = Endpoint 15 ...<br>Bit<br>1 = Endpoint 1  |

**16.5.2.33 USB0UTMI Register (offset = E0h) [reset = 200002h]**

USB0UTMI is shown in [Figure 16-96](#) and described in [Table 16-105](#).

**Figure 16-96. USB0UTMI Register**

|              |               |            |               |               |           |             |          |
|--------------|---------------|------------|---------------|---------------|-----------|-------------|----------|
| 31           | 30            | 29         | 28            | 27            | 26        | 25          | 24       |
| Reserved     |               |            |               |               |           |             |          |
| 23           | 22            | 21         | 20            | 19            | 18        | 17          | 16       |
| TXBITSTUFFEN | TXBITSTUFFENH | OTGDISABLE | VBUSVLDEXTSEL | VBUSVLDEXT    | TXENABLEN | FSXCVROWNER | TXVALIDH |
| R/W-0h       | R/W-0h        | R/W-1h     | R/W-0h        | R/W-0h        | R/W-0h    | R/W-0h      | R/W-0h   |
| 15           | 14            | 13         | 12            | 11            | 10        | 9           | 8        |
| DATAINH      |               |            |               |               |           |             |          |
| R/W-0h       |               |            |               |               |           |             |          |
| 7            | 6             | 5          | 4             | 3             | 2         | 1           | 0        |
| Reserved     |               |            |               | WORDINTERFACE |           | FSDATAEXT   | FSSE0EXT |
|              |               |            |               | R/W-0h        |           | R/W-1h      | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-105. USB0UTMI Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 23   | TXBITSTUFFEN  | R/W  | 0h    | PHY UTMI input for signal txbitstuffen                    |
| 22   | TXBITSTUFFENH | R/W  | 0h    | PHY UTMI input for signal txbitstuffenh                   |
| 21   | OTGDISABLE    | R/W  | 1h    | PHY UTMI input for signal otgdisable                      |
| 20   | VBUSVLDEXTSEL | R/W  | 0h    | PHY UTMI input for signal vbusvldextsel                   |
| 19   | VBUSVLDEXT    | R/W  | 0h    | PHY UTMI input for signal vbusvldext                      |
| 18   | TXENABLEN     | R/W  | 0h    | PHY UTMI input for signal txenablen                       |
| 17   | FSXCVROWNER   | R/W  | 0h    | PHY UTMI input for signal fsxcvrowner                     |
| 16   | TXVALIDH      | R/W  | 0h    | PHY UTMI input for signal txvalidh                        |
| 15-8 | DATAINH       | R/W  | 0h    | PHY UTMI input for signal datainh                         |
| 2    | WORDINTERFACE | R/W  | 0h    | PHY UTMI input for signal wordinterface                   |
| 1    | FSDATAEXT     | R/W  | 1h    | PHY UTMI input for signal fsdataext                       |
| 0    | FSSE0EXT      | R/W  | 0h    | PHY UTMI input for signal fsse0ext USB0 PHY UTMI Register |

**16.5.2.34 USB0MGCUTMILB Register (offset = E4h) [reset = 82h]**

 USB0MGCUTMILB is shown in [Figure 16-97](#) and described in [Table 16-106](#).

**Figure 16-97. USB0MGCUTMILB Register**

| 31        | 30       | 29       | 28       | 27          | 26         | 25         | 24       |
|-----------|----------|----------|----------|-------------|------------|------------|----------|
|           | Reserved |          | SUSPENDM |             | OPMODE     | TXVALID    | XCVRSEL  |
|           |          |          | R-0h     |             | R-0h       | R-0h       | R-0h     |
| 23        | 22       | 21       | 20       | 19          | 18         | 17         | 16       |
| XCVRSEL   | TERMSEL  | DRVVBUS  | CHRGVBUS | DISCHRGVBUS | DPPULLDOWN | DMPULLDOWN | IDPULLUP |
| R-0h      | R-0h     | R-0h     | R-0h     | R-0h        | R-0h       | R-0h       | R-0h     |
| 15        | 14       | 13       | 12       | 11          | 10         | 9          | 8        |
|           | Reserved |          |          | IDDIG       | HOSTDISCON | SESEND     | AVALID   |
|           |          |          |          | R/W-0h      | R/W-0h     | R/W-0h     | R/W-0h   |
| 7         | 6        | 5        | 4        | 3           | 2          | 1          | 0        |
| VBUSVALID | RXERROR  | Reserved |          | LINESTATE   |            | Reserved   |          |
| R/W-1h    | R/W-0h   |          |          | R/W-0h      |            |            |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-106. USB0MGCUTMILB Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description                                  |
|-------|-------------|------|-------|--|
| 28    | SUSPENDM    | R    | 0h    | loopback test observed value for suspendm    |
| 27-26 | OPMODE      | R    | 0h    | loopback test observed value for opmode      |
| 25    | TXVALID     | R    | 0h    | loopback test observed value for txvalid     |
| 24-23 | XCVRSEL     | R    | 0h    | loopback test observed value for xcvsel      |
| 22    | TERMSEL     | R    | 0h    | loopback test observed value for termsel     |
| 21    | DRVVBUS     | R    | 0h    | loopback test observed value for drvvbus     |
| 20    | CHRGVBUS    | R    | 0h    | loopback test observed value for chrgvbus    |
| 19    | DISCHRGVBUS | R    | 0h    | loopback test observed value for dischrgvbus |
| 18    | DPPULLDOWN  | R    | 0h    | loopback test observed value for dppulldown  |
| 17    | DMPULLDOWN  | R    | 0h    | loopback test observed value for dmpulldown  |
| 16    | IDPULLUP    | R    | 0h    | loopback test observed value for idpullup    |
| 11    | IDDIG       | R/W  | 0h    | loopback test value for iddig                |
| 10    | HOSTDISCON  | R/W  | 0h    | loopback test value for hostdiscon           |
| 9     | SESEND      | R/W  | 0h    | loopback test value for sessend              |
| 8     | AVALID      | R/W  | 0h    | loopback test value for avalid               |
| 7     | VBUSVALID   | R/W  | 1h    | loopback test value for vbusvalid            |
| 6     | RXERROR     | R/W  | 0h    | loopback test value for rxerror              |
| 3-2   | LINESTATE   | R/W  | 0h    | loopback test value for linestate            |

### 16.5.2.35 USB0MODE Register (offset = E8h) [reset = 100h]

USB0MODE is shown in [Figure 16-98](#) and described in [Table 16-107](#).

**Figure 16-98. USB0MODE Register**

|           |    |          |    |    |          |       |          |
|-----------|----|----------|----|----|----------|-------|----------|
| 31        | 30 | 29       | 28 | 27 | 26       | 25    | 24       |
| Reserved  |    |          |    |    |          |       |          |
| 23        | 22 | 21       | 20 | 19 | 18       | 17    | 16       |
| Reserved  |    |          |    |    |          |       |          |
| 15        | 14 | 13       | 12 | 11 | 10       | 9     | 8        |
| Reserved  |    |          |    |    |          | IDDIG |          |
| R/W-1h    |    |          |    |    |          |       |          |
| 7         | 6  | 5        | 4  | 3  | 2        | 1     | 0        |
| IDDIG_MUX |    | Reserved |    |    | PHY_TEST |       | LOOPBACK |
| R/W-0h    |    | R/W-0h   |    |    | R/W-0h   |       |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-107. USB0MODE Register Field Descriptions**

| Bit | Field     | Type | Reset | Description  |
|-----|-----------|------|-------|--|
| 8   | IDDIG     | R/W  | 1h    | MGC input value for iddig<br>0=A type<br>1=B type  |
| 7   | IDDIG_MUX | R/W  | 0h    | Multiplexer control for IDDIG signal going to the controller.<br>0 = IDDIG is from PHY0.<br>1 = IDDIG is from bit 8 (IDDIG) of this USB0MODE register. |
| 1   | PHY_TEST  | R/W  | 0h    | PHY test<br>0 = Normal mode<br>1 = PHY test mode   |
| 0   | LOOPBACK  | R/W  | 0h    | Loopback test mode<br>0 = Normal mode<br>1 = Loopback test mode USB0 Mode Register   |

### 16.5.3 USB1\_CTRL Registers

[Table 16-108](#) lists the memory-mapped registers for the USB1\_CTRL. All register offset addresses not listed in [Table 16-108](#) should be considered as reserved locations and the register contents should not be modified.

**Table 16-108. USB1\_CTRL REGISTERS**

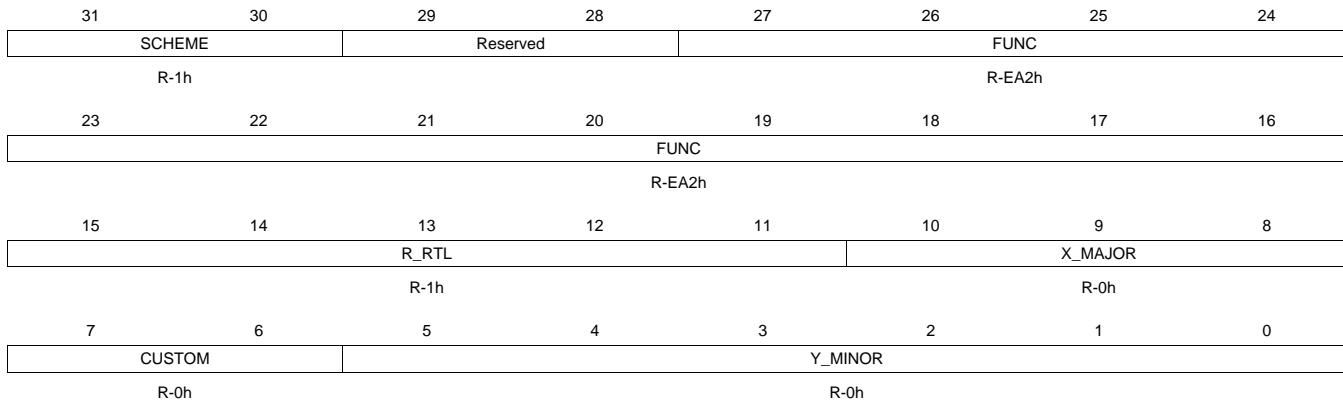
| Offset | Acronym           | Register Name | Section                           |
|--------|-------------------|---------------|-----------------------------------|
| 0h     | USB1REV           |               | <a href="#">Section 16.5.3.1</a>  |
| 14h    | USB1CTRL          |               | <a href="#">Section 16.5.3.2</a>  |
| 18h    | USB1STAT          |               | <a href="#">Section 16.5.3.3</a>  |
| 20h    | USB1IRQMSTAT      |               | <a href="#">Section 16.5.3.4</a>  |
| 28h    | USB1IRQSTATRAW0   |               | <a href="#">Section 16.5.3.5</a>  |
| 2Ch    | USB1IRQSTATRAW1   |               | <a href="#">Section 16.5.3.6</a>  |
| 30h    | USB1IRQSTAT0      |               | <a href="#">Section 16.5.3.7</a>  |
| 34h    | USB1IRQSTAT1      |               | <a href="#">Section 16.5.3.8</a>  |
| 38h    | USB1IRQENABLESET0 |               | <a href="#">Section 16.5.3.9</a>  |
| 3Ch    | USB1IRQENABLESET1 |               | <a href="#">Section 16.5.3.10</a> |
| 40h    | USB1IRQENABLECLR0 |               | <a href="#">Section 16.5.3.11</a> |
| 44h    | USB1IRQENABLECLR1 |               | <a href="#">Section 16.5.3.12</a> |
| 70h    | USB1TXMODE        |               | <a href="#">Section 16.5.3.13</a> |

**Table 16-108. USB1\_CTRL REGISTERS (continued)**

| Offset | Acronym          | Register Name | Section                           |
|--------|------------------|---------------|-----------------------------------|
| 74h    | USB1RXMODE       |               | <a href="#">Section 16.5.3.14</a> |
| 80h    | USB1GENRNDISEP1  |               | <a href="#">Section 16.5.3.15</a> |
| 84h    | USB1GENRNDISEP2  |               | <a href="#">Section 16.5.3.16</a> |
| 88h    | USB1GENRNDISEP3  |               | <a href="#">Section 16.5.3.17</a> |
| 8Ch    | USB1GENRNDISEP4  |               | <a href="#">Section 16.5.3.18</a> |
| 90h    | USB1GENRNDISEP5  |               | <a href="#">Section 16.5.3.19</a> |
| 94h    | USB1GENRNDISEP6  |               | <a href="#">Section 16.5.3.20</a> |
| 98h    | USB1GENRNDISEP7  |               | <a href="#">Section 16.5.3.21</a> |
| 9Ch    | USB1GENRNDISEP8  |               | <a href="#">Section 16.5.3.22</a> |
| A0h    | USB1GENRNDISEP9  |               | <a href="#">Section 16.5.3.23</a> |
| A4h    | USB1GENRNDISEP10 |               | <a href="#">Section 16.5.3.24</a> |
| A8h    | USB1GENRNDISEP11 |               | <a href="#">Section 16.5.3.25</a> |
| ACh    | USB1GENRNDISEP12 |               | <a href="#">Section 16.5.3.26</a> |
| B0h    | USB1GENRNDISEP13 |               | <a href="#">Section 16.5.3.27</a> |
| B4h    | USB1GENRNDISEP14 |               | <a href="#">Section 16.5.3.28</a> |
| B8h    | USB1GENRNDISEP15 |               | <a href="#">Section 16.5.3.29</a> |
| D0h    | USB1AUTOREQ      |               | <a href="#">Section 16.5.3.30</a> |
| D4h    | USB1SRPFIXTIME   |               | <a href="#">Section 16.5.3.31</a> |
| D8h    | USB1TDOWN        |               | <a href="#">Section 16.5.3.32</a> |
| E0h    | USB1UTMI         |               | <a href="#">Section 16.5.3.33</a> |
| E4h    | USB1UTMILB       |               | <a href="#">Section 16.5.3.34</a> |
| E8h    | USB1MODE         |               | <a href="#">Section 16.5.3.35</a> |

**16.5.3.1 USB1REV Register (offset = 0h) [reset = 4EA20800h]**

USB1REV is shown in [Figure 16-99](#) and described in [Table 16-109](#).

**Figure 16-99. USB1REV Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-109. USB1REV Register Field Descriptions**

| Bit   | Field   | Type | Reset | Description   |
|-------|---------|------|-------|---|
| 31-30 | SCHEME  | R    | 1h    | Used to distinguish between legacy interface scheme and current.<br>0 = Legacy<br>1 = Current |
| 27-16 | FUNC    | R    | EA2h  | Function indicates a software compatible module family.                                       |
| 15-11 | R RTL   | R    | 1h    | RTL revision.<br>Will vary depending on release.  |
| 10-8  | X_MAJOR | R    | 0h    | Major revision.   |
| 7-6   | CUSTOM  | R    | 0h    | Custom revision   |
| 5-0   | Y_MINOR | R    | 0h    | Minor revision.<br>USB1 Revision Register   |

### 16.5.3.2 USB1CTRL Register (offset = 14h) [reset = 0h]

USB1CTRL is shown in [Figure 16-100](#) and described in [Table 16-110](#).

**Figure 16-100. USB1CTRL Register**

| 31       | 30      | 29                   | 28     | 27       | 26       | 25      | 24         |
|----------|---------|----------------------|--------|----------|----------|---------|------------|
| DIS_DEB  | DIS_SRП |                      |        | Reserved |          |         |            |
| R/W-0h   | R/W-0h  |                      |        |          |          |         |            |
| 23       | 22      | 21                   | 20     | 19       | 18       | 17      | 16         |
|          |         |                      |        | Reserved |          |         |            |
| 15       | 14      | 13                   | 12     | 11       | 10       | 9       | 8          |
|          |         |                      |        | Reserved |          |         |            |
| 7        | 6       | 5                    | 4      | 3        | 2        | 1       | 0          |
| Reserved |         | SOFT_RESET_ISOLATION | RNDIS  | UINT     | Reserved | CLKFACK | SOFT_RESET |
|          |         |                      | R/W-0h | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-110. USB1CTRL Register Field Descriptions**

| Bit | Field                | Type | Reset | Description  |
|-----|----------------------|------|-------|--|
| 31  | DIS_DEB              | R/W  | 0h    | Disable the VBUS debouncer circuit fix   |
| 30  | DIS_SRП              | R/W  | 0h    | Disable the OTG Session Request Protocol (SRP) AVALID circuit fix. When enabled (=0) this allows additional time for the VBUS signal to be measured against the VBUS thresholds. The time is specified in the USB0 SRP Fix Time Register.  |
| 5   | SOFT_RESET_ISOLATION | R/W  | 0h    | Soft reset isolation. When high this bit forces all USB1 signals that connect to the USBSS to known values during a soft reset via bit 0 of this register. This bit should be set high prior to setting bit 0 and cleared after bit 0 is cleared.  |
| 4   | RNDIS                | R/W  | 0h    | Global RNDIS mode enable for all endpoints.  |
| 3   | UINT                 | R/W  | 0h    | USB interrupt enable<br>1 = Legacy<br>0 = Current (recommended setting)<br>If uint is set high, then the mentor controller generic interrupt for USB[9] will be generated (if enabled). This requires S/W to read the mentor controller's registers to determine which interrupt from USB[0] to USB[7] that occurred. If uint is set low, then the usb20otg_f module will automatically read the mentor controller's registers and set the appropriate interrupt from USB[0] to USB[7] (if enabled). The generic interrupt for USB[9] will not be generated. |
| 1   | CLKFACK              | R/W  | 0h    | Clock stop fast ack enable.  |

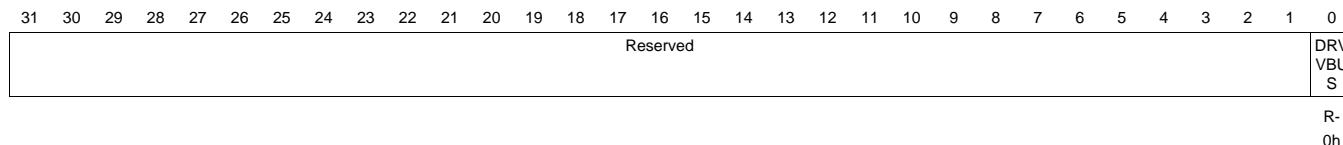
**Table 16-110. USB1CTRL Register Field Descriptions (continued)**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 0   | SOFT_RESET | R/W  | 0h    | <p>Software reset of USB1.<br/>         Write 0 = no action<br/>         Write 1 = Initiate software reset<br/>         Read 0 = Reset done, no action<br/>         Read 1 = Reset ongoing<br/>         Both the soft_reset and soft_reset_isolation bits should be asserted simultaneously.<br/>         This will cause the following sequence of actions to occur over multiple cycles:<br/>         - All USB0 output signals will go to a known constant value via multiplexers.<br/>         This removes the possibility of timing errors due to the asynchronous resets.<br/>         - All USB0 registers will be reset.<br/>         - The USB0 resets will be de-asserted.<br/>         - The reset isolation multiplexer inputs will be de-selected.<br/>         - Both the soft_reset and soft_reset_isolation bits will be automatically cleared.<br/>         Setting only the soft_reset_isolation bit will cause all USB0 output signals to go to a known constant value via multiplexers.<br/>         This will prevent future access to USB0.<br/>         To clear this condition the USBSS must be reset via a hard or soft reset.</p> |

### 16.5.3.3 USB1STAT Register (offset = 18h) [reset = 0h]

USB1STAT is shown in [Figure 16-101](#) and described in [Table 16-111](#).

**Figure 16-101. USB1STAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-111. USB1STAT Register Field Descriptions**

| Bit | Field   | Type | Reset | Description                                |
|-----|---------|------|-------|--|
| 0   | DRVVBUS | R    | 0h    | Current DRVVBUS value USB1 Status Register |

**16.5.3.4 USB1IRQMSTAT Register (offset = 20h) [reset = 0h]**

USB1IRQMSTAT is shown in [Figure 16-102](#) and described in [Table 16-112](#).

**Figure 16-102. USB1IRQMSTAT Register**

|          |    |    |    |    |    |       |       |
|----------|----|----|----|----|----|-------|-------|
| 31       | 30 | 29 | 28 | 27 | 26 | 25    | 24    |
| Reserved |    |    |    |    |    |       |       |
| 23       | 22 | 21 | 20 | 19 | 18 | 17    | 16    |
| Reserved |    |    |    |    |    |       |       |
| 15       | 14 | 13 | 12 | 11 | 10 | 9     | 8     |
| Reserved |    |    |    |    |    |       |       |
| 7        | 6  | 5  | 4  | 3  | 2  | 1     | 0     |
| Reserved |    |    |    |    |    | BANK1 | BANK0 |
|          |    |    |    |    |    | R-0h  | R-0h  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-112. USB1IRQMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 1   | BANK1 | R    | 0h    | 0: No events pending from IRQ_STATUS_1<br>1: At least one event is pending from IRQ_STATUS_1                                 |
| 0   | BANK0 | R    | 0h    | 0: No events pending from IRQ_STATUS_0<br>1: At least one event is pending from IRQ_STATUS_0 USB1 IRQ_MERGED_STATUS Register |

**16.5.3.5 USB1IRQSTATRAW0 Register (offset = 28h) [reset = 0h]**

 USB1IRQSTATRAW0 is shown in [Figure 16-103](#) and described in [Table 16-113](#).

**Figure 16-103. USB1IRQSTATRAW0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-113. USB1IRQSTATRAW0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt status for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt status for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt status for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt status for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt status for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt status for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt status for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt status for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt status for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt status for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt status for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt status for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt status for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt status for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt status for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt status for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt status for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt status for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt status for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt status for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt status for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt status for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt status for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt status for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt status for TX endpoint 6  |

**Table 16-113. USB1IRQSTATRAW0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt status for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt status for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt status for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt status for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt status for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt status for TX endpoint 0 USB1 IRQ_STATUS_RAW_0 Register |

**16.5.3.6 USB1IRQSTATRAW1 Register (offset = 2Ch) [reset = 0h]**

 USB1IRQSTATRAW1 is shown in [Figure 16-104](#) and described in [Table 16-114](#).

**Figure 16-104. USB1IRQSTATRAW1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | Reserved  |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    |           |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-114. USB1IRQSTATRAW1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 15   |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 14   |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 13   |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 12   |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 11   |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 10   |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 9  |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 8  |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 7  |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 6  |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 5  |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 4  |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 3  |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 2  |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 1  |
| 9   | USB_9      | R/W  | 0h    | Interrupt status for Mentor controller USB_INT generic interrupt                               |
| 8   | USB_8      | R/W  | 0h    | Interrupt status for DRVVVBUS level change   |
| 7   | USB_7      | R/W  | 0h    | Interrupt status for VBUS andt VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt status for SRP detected  |
| 5   | USB_5      | R/W  | 0h    | Interrupt status for device disconnected (host mode)   |
| 4   | USB_4      | R/W  | 0h    | Interrupt status for device connected (host mode)  |
| 3   | USB_3      | R/W  | 0h    | Interrupt status for SOF started   |
| 2   | USB_2      | R/W  | 0h    | Interrupt status for Reset signaling detected (peripheral mode)<br>Babble detected (host mode) |
| 1   | USB_1      | R/W  | 0h    | Interrupt status for Resume signaling detected   |

**Table 16-114. USB1IRQSTATRAW1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 0   | USB_0 | R/W  | 0h    | Interrupt status for Suspend signaling detected USB1 IRQ_STATUS_RAW_1 Register |

### 16.5.3.7 USB1IRQSTAT0 Register (offset = 30h) [reset = 0h]

USB1IRQSTAT0 is shown in Figure 16-105 and described in Table 16-115.

**Figure 16-105. USB1IRQSTAT0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-115. USB1IRQSTAT0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt status for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt status for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt status for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt status for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt status for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt status for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt status for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt status for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt status for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt status for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt status for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt status for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt status for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt status for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt status for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt status for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt status for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt status for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt status for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt status for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt status for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt status for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt status for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt status for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt status for TX endpoint 6  |

**Table 16-115. USB1IRQSTAT0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt status for TX endpoint 5                            |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt status for TX endpoint 4                            |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt status for TX endpoint 3                            |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt status for TX endpoint 2                            |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt status for TX endpoint 1                            |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt status for TX endpoint 0 USB1 IRQ_STATUS_0 Register |

### 16.5.3.8 USB1IRQSTAT1 Register (offset = 34h) [reset = 0h]

USB1IRQSTAT1 is shown in Figure 16-106 and described in Table 16-116.

**Figure 16-106. USB1IRQSTAT1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | Reserved  |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    |           |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-116. USB1IRQSTAT1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 15   |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 14   |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 13   |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 12   |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 11   |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt status for TX FIFO endpoint 10   |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 9  |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 8  |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 7  |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 6  |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 5  |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 4  |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 3  |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 2  |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt status for TX FIFO endpoint 1  |
| 9   | USB_9      | R/W  | 0h    | Interrupt status for Mentor controller USB_INT generic interrupt                               |
| 8   | USB_8      | R/W  | 0h    | Interrupt status for DRVVVBUS level change   |
| 7   | USB_7      | R/W  | 0h    | Interrupt status for VBUS andt VBUS valid threshold  |
| 6   | USB_6      | R/W  | 0h    | Interrupt status for SRP detected  |
| 5   | USB_5      | R/W  | 0h    | Interrupt status for device disconnected (host mode)   |
| 4   | USB_4      | R/W  | 0h    | Interrupt status for device connected (host mode)  |
| 3   | USB_3      | R/W  | 0h    | Interrupt status for SOF started   |
| 2   | USB_2      | R/W  | 0h    | Interrupt status for Reset signaling detected (peripheral mode)<br>Babble detected (host mode) |
| 1   | USB_1      | R/W  | 0h    | Interrupt status for Resume signaling detected   |

**Table 16-116. USB1IRQSTAT1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 0   | USB_0 | R/W  | 0h    | Interrupt status for Suspend signaling detected USB1 IRQ_STATUS_1 Register |

### 16.5.3.9 USB1IRQENABLESET0 Register (offset = 38h) [reset = 0h]

USB1IRQENABLESET0 is shown in [Figure 16-107](#) and described in [Table 16-117](#).

**Figure 16-107. USB1IRQENABLESET0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-117. USB1IRQENABLESET0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt enable for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt enable for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt enable for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt enable for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt enable for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt enable for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt enable for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt enable for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt enable for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt enable for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt enable for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt enable for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt enable for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt enable for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt enable for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt enable for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt enable for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt enable for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt enable for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt enable for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt enable for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt enable for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt enable for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt enable for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt enable for TX endpoint 6  |

**Table 16-117. USB1IRQENABLESET0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt enable for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt enable for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt enable for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt enable for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt enable for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt enable for TX endpoint 0 USB1 IRQ_ENABLE_SET_0 Register |

### 16.5.3.10 USB1IRQENABLESET1 Register (offset = 3Ch) [reset = 0h]

USB1IRQENABLESET1 is shown in [Figure 16-108](#) and described in [Table 16-118](#).

**Figure 16-108. USB1IRQENABLESET1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | Reserved  |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    |           |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-118. USB1IRQENABLESET1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 15  |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 14  |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 13  |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 12  |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 11  |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 10  |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 9   |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 8   |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 7   |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 6   |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 5   |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 4   |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 3   |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 2   |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 1   |
| 9   | USB_9      | R/W  | 0h    | Interrupt enable for Mentor controller USB_INT generic interrupt                                |
| 8   | USB_8      | R/W  | 0h    | Interrupt enable for DRVVBUS level change   |
| 7   | USB_7      | R/W  | 0h    | Interrupt enable for VBUS andt VBUS valid threshold   |
| 6   | USB_6      | R/W  | 0h    | Interrupt enable for SRP detected   |
| 5   | USB_5      | R/W  | 0h    | Interrupt enable for device disconnected (host mode)  |
| 4   | USB_4      | R/W  | 0h    | Interrupt enable for device connected (host mode)   |
| 3   | USB_3      | R/W  | 0h    | Interrupt enable for SOF started  |
| 2   | USB_2      | R/W  | 0h    | Interrupt enable for Reset signalling detected (peripheral mode)<br>Babble detected (host mode) |
| 1   | USB_1      | R/W  | 0h    | Interrupt enable for Resume signalling detected   |

**Table 16-118. USB1IRQENABLESET1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 0   | USB_0 | R/W  | 0h    | Interrupt enable for Suspend signaling detected USB1 IRQ_ENABLE_SET_1 Register |

### 16.5.3.11 USB1IRQENABLECLR0 Register (offset = 40h) [reset = 0h]

USB1IRQENABLECLR0 is shown in [Figure 16-109](#) and described in [Table 16-119](#).

**Figure 16-109. USB1IRQENABLECLR0 Register**

| 31       | 30       | 29       | 28       | 27       | 26       | 25      | 24       |
|----------|----------|----------|----------|----------|----------|---------|----------|
| RX_EP_15 | RX_EP_14 | RX_EP_13 | RX_EP_12 | RX_EP_11 | RX_EP_10 | RX_EP_9 | RX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 23       | 22       | 21       | 20       | 19       | 18       | 17      | 16       |
| RX_EP_7  | RX_EP_6  | RX_EP_5  | RX_EP_4  | RX_EP_3  | RX_EP_2  | RX_EP_1 | Reserved |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 15       | 14       | 13       | 12       | 11       | 10       | 9       | 8        |
| TX_EP_15 | TX_EP_14 | TX_EP_13 | TX_EP_12 | TX_EP_11 | TX_EP_10 | TX_EP_9 | TX_EP_8  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |
| 7        | 6        | 5        | 4        | 3        | 2        | 1       | 0        |
| TX_EP_7  | TX_EP_6  | TX_EP_5  | TX_EP_4  | TX_EP_3  | TX_EP_2  | TX_EP_1 | TX_EP_0  |
| R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h   | R/W-0h  | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-119. USB1IRQENABLECLR0 Register Field Descriptions**

| Bit | Field    | Type | Reset | Description                         |
|-----|----------|------|-------|-------------------------------------|
| 31  | RX_EP_15 | R/W  | 0h    | Interrupt enable for RX endpoint 15 |
| 30  | RX_EP_14 | R/W  | 0h    | Interrupt enable for RX endpoint 14 |
| 29  | RX_EP_13 | R/W  | 0h    | Interrupt enable for RX endpoint 13 |
| 28  | RX_EP_12 | R/W  | 0h    | Interrupt enable for RX endpoint 12 |
| 27  | RX_EP_11 | R/W  | 0h    | Interrupt enable for RX endpoint 11 |
| 26  | RX_EP_10 | R/W  | 0h    | Interrupt enable for RX endpoint 10 |
| 25  | RX_EP_9  | R/W  | 0h    | Interrupt enable for RX endpoint 9  |
| 24  | RX_EP_8  | R/W  | 0h    | Interrupt enable for RX endpoint 8  |
| 23  | RX_EP_7  | R/W  | 0h    | Interrupt enable for RX endpoint 7  |
| 22  | RX_EP_6  | R/W  | 0h    | Interrupt enable for RX endpoint 6  |
| 21  | RX_EP_5  | R/W  | 0h    | Interrupt enable for RX endpoint 5  |
| 20  | RX_EP_4  | R/W  | 0h    | Interrupt enable for RX endpoint 4  |
| 19  | RX_EP_3  | R/W  | 0h    | Interrupt enable for RX endpoint 3  |
| 18  | RX_EP_2  | R/W  | 0h    | Interrupt enable for RX endpoint 2  |
| 17  | RX_EP_1  | R/W  | 0h    | Interrupt enable for RX endpoint 1  |
| 15  | TX_EP_15 | R/W  | 0h    | Interrupt enable for TX endpoint 15 |
| 14  | TX_EP_14 | R/W  | 0h    | Interrupt enable for TX endpoint 14 |
| 13  | TX_EP_13 | R/W  | 0h    | Interrupt enable for TX endpoint 13 |
| 12  | TX_EP_12 | R/W  | 0h    | Interrupt enable for TX endpoint 12 |
| 11  | TX_EP_11 | R/W  | 0h    | Interrupt enable for TX endpoint 11 |
| 10  | TX_EP_10 | R/W  | 0h    | Interrupt enable for TX endpoint 10 |
| 9   | TX_EP_9  | R/W  | 0h    | Interrupt enable for TX endpoint 9  |
| 8   | TX_EP_8  | R/W  | 0h    | Interrupt enable for TX endpoint 8  |
| 7   | TX_EP_7  | R/W  | 0h    | Interrupt enable for TX endpoint 7  |
| 6   | TX_EP_6  | R/W  | 0h    | Interrupt enable for TX endpoint 6  |

**Table 16-119. USB1IRQENABLECLR0 Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 5   | TX_EP_5 | R/W  | 0h    | Interrupt enable for TX endpoint 5                                |
| 4   | TX_EP_4 | R/W  | 0h    | Interrupt enable for TX endpoint 4                                |
| 3   | TX_EP_3 | R/W  | 0h    | Interrupt enable for TX endpoint 3                                |
| 2   | TX_EP_2 | R/W  | 0h    | Interrupt enable for TX endpoint 2                                |
| 1   | TX_EP_1 | R/W  | 0h    | Interrupt enable for TX endpoint 1                                |
| 0   | TX_EP_0 | R/W  | 0h    | Interrupt enable for TX endpoint 0 USB1 IRQ_ENABLE_CLR_0 Register |

### 16.5.3.12 USB1IRQENABLECLR1 Register (offset = 44h) [reset = 0h]

USB1IRQENABLECLR1 is shown in [Figure 16-110](#) and described in [Table 16-120](#).

**Figure 16-110. USB1IRQENABLECLR1 Register**

| 31         | 30         | 29         | 28         | 27         | 26         | 25        | 24        |
|------------|------------|------------|------------|------------|------------|-----------|-----------|
| TX_FIFO_15 | TX_FIFO_14 | TX_FIFO_13 | TX_FIFO_12 | TX_FIFO_11 | TX_FIFO_10 | TX_FIFO_9 | TX_FIFO_8 |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |
| 23         | 22         | 21         | 20         | 19         | 18         | 17        | 16        |
| TX_FIFO_7  | TX_FIFO_6  | TX_FIFO_5  | TX_FIFO_4  | TX_FIFO_3  | TX_FIFO_2  | TX_FIFO_1 | Reserved  |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    |           |
| 15         | 14         | 13         | 12         | 11         | 10         | 9         | 8         |
| Reserved   |            |            |            |            |            | USB_9     | USB_8     |
| R/W-0h     |            |            |            |            |            |           |           |
| 7          | 6          | 5          | 4          | 3          | 2          | 1         | 0         |
| USB_7      | USB_6      | USB_5      | USB_4      | USB_3      | USB_2      | USB_1     | USB_0     |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-120. USB1IRQENABLECLR1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description   |
|-----|------------|------|-------|---|
| 31  | TX_FIFO_15 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 15  |
| 30  | TX_FIFO_14 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 14  |
| 29  | TX_FIFO_13 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 13  |
| 28  | TX_FIFO_12 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 12  |
| 27  | TX_FIFO_11 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 11  |
| 26  | TX_FIFO_10 | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 10  |
| 25  | TX_FIFO_9  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 9   |
| 24  | TX_FIFO_8  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 8   |
| 23  | TX_FIFO_7  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 7   |
| 22  | TX_FIFO_6  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 6   |
| 21  | TX_FIFO_5  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 5   |
| 20  | TX_FIFO_4  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 4   |
| 19  | TX_FIFO_3  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 3   |
| 18  | TX_FIFO_2  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 2   |
| 17  | TX_FIFO_1  | R/W  | 0h    | Interrupt enable for TX FIFO endpoint 1   |
| 9   | USB_9      | R/W  | 0h    | Interrupt enable for Mentor controller USB_INT generic interrupt                                |
| 8   | USB_8      | R/W  | 0h    | Interrupt enable for DRVVBUS level change   |
| 7   | USB_7      | R/W  | 0h    | Interrupt enable for VBUS andt VBUS valid threshold   |
| 6   | USB_6      | R/W  | 0h    | Interrupt enable for SRP detected   |
| 5   | USB_5      | R/W  | 0h    | Interrupt enable for device disconnected (host mode)  |
| 4   | USB_4      | R/W  | 0h    | Interrupt enable for device connected (host mode)   |
| 3   | USB_3      | R/W  | 0h    | Interrupt enable for SOF started  |
| 2   | USB_2      | R/W  | 0h    | Interrupt enable for Reset signalling detected (peripheral mode)<br>Babble detected (host mode) |
| 1   | USB_1      | R/W  | 0h    | Interrupt enable for Resume signalling detected   |

**Table 16-120. USB1IRQENABLECLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 0   | USB_0 | R/W  | 0h    | Interrupt enable for Suspend signaling detected USB1 IRQ_ENABLE_CLR_1 Register |

**16.5.3.13 USB1TXMODE Register (offset = 70h) [reset = 0h]**

 USB1TXMODE is shown in [Figure 16-111](#) and described in [Table 16-121](#).

**Figure 16-111. USB1TXMODE Register**

| 31        | 30 | 29        | 28     | 27        | 26     | 25        | 24     |
|-----------|----|-----------|--------|-----------|--------|-----------|--------|
| Reserved  |    | TX15_MODE |        | TX14_MODE |        | TX13_MODE |        |
|           |    | R/W-0h    |        |           | R/W-0h |           | R/W-0h |
| 23        | 22 | 21        | 20     | 19        | 18     | 17        | 16     |
| TX12_MODE |    | TX11_MODE |        | TX10_MODE |        | TX9_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           | R/W-0h |           | R/W-0h |
| 15        | 14 | 13        | 12     | 11        | 10     | 9         | 8      |
| TX8_MODE  |    | TX7_MODE  |        | TX6_MODE  |        | TX5_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           | R/W-0h |           | R/W-0h |
| 7         | 6  | 5         | 4      | 3         | 2      | 1         | 0      |
| TX4_MODE  |    | TX3_MODE  |        | TX2_MODE  |        | TX1_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           | R/W-0h |           | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-121. USB1TXMODE Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 29-28 | TX15_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 14<br>01: RNDIS Mode on TX endpoint 14<br>10: CDC Mode on TX endpoint 14<br>11: Generic RNDIS Mode on TX endpoint 14 |
| 27-26 | TX14_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 13<br>01: RNDIS Mode on TX endpoint 13<br>10: CDC Mode on TX endpoint 13<br>11: Generic RNDIS Mode on TX endpoint 13 |
| 25-24 | TX13_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 12<br>01: RNDIS Mode on TX endpoint 12<br>10: CDC Mode on TX endpoint 12<br>11: Generic RNDIS Mode on TX endpoint 12 |
| 23-22 | TX12_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 11<br>01: RNDIS Mode on TX endpoint 11<br>10: CDC Mode on TX endpoint 11<br>11: Generic RNDIS Mode on TX endpoint 11 |
| 21-20 | TX11_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 10<br>01: RNDIS Mode on TX endpoint 10<br>10: CDC Mode on TX endpoint 10<br>11: Generic RNDIS Mode on TX endpoint 10 |
| 19-18 | TX10_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 9<br>01: RNDIS Mode on TX endpoint 9<br>10: CDC Mode on TX endpoint 9<br>11: Generic RNDIS Mode on TX endpoint 9     |
| 17-16 | TX9_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 8<br>01: RNDIS Mode on TX endpoint 8<br>10: CDC Mode on TX endpoint 8<br>11: Generic RNDIS Mode on TX endpoint 8     |
| 15-14 | TX8_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 7<br>01: RNDIS Mode on TX endpoint 7<br>10: CDC Mode on TX endpoint 7<br>11: Generic RNDIS Mode on TX endpoint 7     |
| 13-12 | TX7_MODE  | R/W  | 0h    | 00: Transparent Mode on TX endpoint 6<br>01: RNDIS Mode on TX endpoint 6<br>10: CDC Mode on TX endpoint 6<br>11: Generic RNDIS Mode on TX endpoint 6     |

**Table 16-121. USB1TXMODE Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 11-10 | TX6_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 5<br>01: RNDIS Mode on TX endpoint 5<br>10: CDC Mode on TX endpoint 5<br>11: Generic RNDIS Mode on TX endpoint 5                        |
| 9-8   | TX5_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 4<br>01: RNDIS Mode on TX endpoint 4<br>10: CDC Mode on TX endpoint 4<br>11: Generic RNDIS Mode on TX endpoint 4                        |
| 7-6   | TX4_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 3<br>01: RNDIS Mode on TX endpoint 3<br>10: CDC Mode on TX endpoint 3<br>11: Generic RNDIS Mode on TX endpoint 3                        |
| 5-4   | TX3_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 2<br>01: RNDIS Mode on TX endpoint 2<br>10: CDC Mode on TX endpoint 2<br>11: Generic RNDIS Mode on TX endpoint 2                        |
| 3-2   | TX2_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 1<br>01: RNDIS Mode on TX endpoint 1<br>10: CDC Mode on TX endpoint 1<br>11: Generic RNDIS Mode on TX endpoint 1                        |
| 1-0   | TX1_MODE | R/W  | 0h    | 00: Transparent Mode on TX endpoint 0<br>01: RNDIS Mode on TX endpoint 0<br>10: CDC Mode on TX endpoint 0<br>11: Generic RNDIS Mode on TX endpoint 0 USB1 Tx Mode Registers |

### 16.5.3.14 USB1RXMODE Register (offset = 74h) [reset = 0h]

USB1RXMODE is shown in [Figure 16-112](#) and described in [Table 16-122](#).

**Figure 16-112. USB1RXMODE Register**

| 31        | 30 | 29        | 28     | 27        | 26     | 25        | 24     |
|-----------|----|-----------|--------|-----------|--------|-----------|--------|
| Reserved  |    | RX15_MODE |        | RX14_MODE |        | RX13_MODE |        |
|           |    | R/W-0h    |        |           | R/W-0h |           | R/W-0h |
| 23        | 22 | 21        | 20     | 19        | 18     | 17        | 16     |
| RX12_MODE |    | RX11_MODE |        | RX10_MODE |        | RX9_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 15        | 14 | 13        | 12     | 11        | 10     | 9         | 8      |
| RX8_MODE  |    | RX7_MODE  |        | RX6_MODE  |        | RX5_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |
| 7         | 6  | 5         | 4      | 3         | 2      | 1         | 0      |
| RX4_MODE  |    | RX3_MODE  |        | RX2_MODE  |        | RX1_MODE  |        |
| R/W-0h    |    |           | R/W-0h |           |        | R/W-0h    |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-122. USB1RXMODE Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 29-28 | RX15_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 14<br>01: RNDIS Mode on RX endpoint 14<br>10: CDC Mode on RX endpoint 14<br>11: Generic RNDIS or Infinite Mode on RX endpoint 14 |
| 27-26 | RX14_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 13<br>01: RNDIS Mode on RX endpoint 13<br>10: CDC Mode on RX endpoint 13<br>11: Generic RNDIS or Infinite Mode on RX endpoint 13 |
| 25-24 | RX13_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 12<br>01: RNDIS Mode on RX endpoint 12<br>10: CDC Mode on RX endpoint 12<br>11: Generic RNDIS or Infinite Mode on RX endpoint 12 |
| 23-22 | RX12_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 11<br>01: RNDIS Mode on RX endpoint 11<br>10: CDC Mode on RX endpoint 11<br>11: Generic RNDIS or Infinite Mode on RX endpoint 11 |
| 21-20 | RX11_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 10<br>01: RNDIS Mode on RX endpoint 10<br>10: CDC Mode on RX endpoint 10<br>11: Generic RNDIS or Infinite Mode on RX endpoint 10 |
| 19-18 | RX10_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 9<br>01: RNDIS Mode on RX endpoint 9<br>10: CDC Mode on RX endpoint 9<br>11: Generic RNDIS or Infinite Mode on RX endpoint 9     |
| 17-16 | RX9_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 8<br>01: RNDIS Mode on RX endpoint 8<br>10: CDC Mode on RX endpoint 8<br>11: Generic RNDIS or Infinite Mode on RX endpoint 8     |
| 15-14 | RX8_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 7<br>01: RNDIS Mode on RX endpoint 7<br>10: CDC Mode on RX endpoint 7<br>11: Generic RNDIS or Infinite Mode on RX endpoint 7     |
| 13-12 | RX7_MODE  | R/W  | 0h    | 00: Transparent Mode on RX endpoint 6<br>01: RNDIS Mode on RX endpoint 6<br>10: CDC Mode on RX endpoint 6<br>11: Generic RNDIS or Infinite Mode on RX endpoint 6     |

**Table 16-122. USB1RXMODE Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 11-10 | RX6_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 5<br>01: RNDIS Mode on RX endpoint 5<br>10: CDC Mode on RX endpoint 5<br>11: Generic RNDIS or Infinite Mode on RX endpoint 5                        |
| 9-8   | RX5_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 4<br>01: RNDIS Mode on RX endpoint 4<br>10: CDC Mode on RX endpoint 4<br>11: Generic RNDIS or Infinite Mode on RX endpoint 4                        |
| 7-6   | RX4_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 3<br>01: RNDIS Mode on RX endpoint 3<br>10: CDC Mode on RX endpoint 3<br>11: Generic RNDIS or Infinite Mode on RX endpoint 3                        |
| 5-4   | RX3_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 2<br>01: RNDIS Mode on RX endpoint 2<br>10: CDC Mode on RX endpoint 2<br>11: Generic RNDIS or Infinite Mode on RX endpoint 2                        |
| 3-2   | RX2_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 1<br>01: RNDIS Mode on RX endpoint 1<br>10: CDC Mode on RX endpoint 1<br>11: Generic RNDIS or Infinite Mode on RX endpoint 1                        |
| 1-0   | RX1_MODE | R/W  | 0h    | 00: Transparent Mode on RX endpoint 0<br>01: RNDIS Mode on RX endpoint 0<br>10: CDC Mode on RX endpoint 0<br>11: Generic RNDIS or Infinite Mode on RX endpoint 0 USB1 Rx Mode Registers |

**16.5.3.15 USB1GENRNDISEP1 Register (offset = 80h) [reset = 0h]**

USB1GENRNDISEP1 is shown in [Figure 16-113](#) and described in [Table 16-123](#).

**Figure 16-113. USB1GENRNDISEP1 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP1_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-123. USB1GENRNDISEP1 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP1_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.16 USB1GENRNDISEP2 Register (offset = 84h) [reset = 0h]**

USB1GENRNDISEP2 is shown in [Figure 16-114](#) and described in [Table 16-124](#).

**Figure 16-114. USB1GENRNDISEP2 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP2_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-124. USB1GENRNDISEP2 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP2_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.17 USB1GENRNDISEP3 Register (offset = 88h) [reset = 0h]**

USB1GENRNDISEP3 is shown in [Figure 16-115](#) and described in [Table 16-125](#).

**Figure 16-115. USB1GENRNDISEP3 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP3_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-125. USB1GENRNDISEP3 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP3_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.18 USB1GENRNDISEP4 Register (offset = 8Ch) [reset = 0h]**

USB1GENRNDISEP4 is shown in [Figure 16-116](#) and described in [Table 16-126](#).

**Figure 16-116. USB1GENRNDISEP4 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP4_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-126. USB1GENRNDISEP4 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP4_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.19 USB1GENRNDISEP5 Register (offset = 90h) [reset = 0h]**

USB1GENRNDISEP5 is shown in [Figure 16-117](#) and described in [Table 16-127](#).

**Figure 16-117. USB1GENRNDISEP5 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP5_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-127. USB1GENRNDISEP5 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP5_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.20 USB1GENRNDISEP6 Register (offset = 94h) [reset = 0h]**

USB1GENRNDISEP6 is shown in [Figure 16-118](#) and described in [Table 16-128](#).

**Figure 16-118. USB1GENRNDISEP6 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP6_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-128. USB1GENRNDISEP6 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP6_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.21 USB1GENRNDISEP7 Register (offset = 98h) [reset = 0h]**

USB1GENRNDISEP7 is shown in [Figure 16-119](#) and described in [Table 16-129](#).

**Figure 16-119. USB1GENRNDISEP7 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP7_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-129. USB1GENRNDISEP7 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP7_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.22 USB1GENRNDISEP8 Register (offset = 9Ch) [reset = 0h]**

USB1GENRNDISEP8 is shown in [Figure 16-120](#) and described in [Table 16-130](#).

**Figure 16-120. USB1GENRNDISEP8 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP8_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-130. USB1GENRNDISEP8 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP8_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.23 USB1GENRNDISEP9 Register (offset = A0h) [reset = 0h]**

USB1GENRNDISEP9 is shown in [Figure 16-121](#) and described in [Table 16-131](#).

**Figure 16-121. USB1GENRNDISEP9 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15       | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP9_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |          |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-131. USB1GENRNDISEP9 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 16-0 | EP9_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.24 USB1GENRNDISEP10 Register (offset = A4h) [reset = 0h]**

USB1GENRNDISEP10 is shown in [Figure 16-122](#) and described in [Table 16-132](#).

**Figure 16-122. USB1GENRNDISEP10 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP10_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-132. USB1GENRNDISEP10 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP10_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.25 USB1GENRNDISEP11 Register (offset = A8h) [reset = 0h]**

USB1GENRNDISEP11 is shown in [Figure 16-123](#) and described in [Table 16-133](#).

**Figure 16-123. USB1GENRNDISEP11 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP11_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-133. USB1GENRNDISEP11 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP11_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.26 USB1GENRNDISEP12 Register (offset = ACh) [reset = 0h]**

USB1GENRNDISEP12 is shown in [Figure 16-124](#) and described in [Table 16-134](#).

**Figure 16-124. USB1GENRNDISEP12 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP12_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-134. USB1GENRNDISEP12 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP12_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.27 USB1GENRNDISEP13 Register (offset = B0h) [reset = 0h]**

USB1GENRNDISEP13 is shown in [Figure 16-125](#) and described in [Table 16-135](#).

**Figure 16-125. USB1GENRNDISEP13 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP13_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-135. USB1GENRNDISEP13 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP13_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.28 USB1GENRNDISEP14 Register (offset = B4h) [reset = 0h]**

USB1GENRNDISEP14 is shown in [Figure 16-126](#) and described in [Table 16-136](#).

**Figure 16-126. USB1GENRNDISEP14 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP14_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-136. USB1GENRNDISEP14 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP14_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.29 USB1GENRNDISEP15 Register (offset = B8h) [reset = 0h]**

USB1GENRNDISEP15 is shown in [Figure 16-127](#) and described in [Table 16-137](#).

**Figure 16-127. USB1GENRNDISEP15 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15        | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EP15_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |           |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-137. USB1GENRNDISEP15 Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 16-0 | EP15_SIZE | R/W  | 0h    | Generic RNDIS packet size.<br>USB1 Generic RNDIS EP N Size Register |

**16.5.3.30 USB1AUTOREQ Register (offset = D0h) [reset = 0h]**

 USB1AUTOREQ is shown in [Figure 16-128](#) and described in [Table 16-138](#).

**Figure 16-128. USB1AUTOREQ Register**

| 31           | 30 | 29           | 28     | 27           | 26     | 25           | 24     |
|--------------|----|--------------|--------|--------------|--------|--------------|--------|
| Reserved     |    | RX15_AUTOREQ |        | RX14_AUTOREQ |        | RX13_AUTOREQ |        |
|              |    | R/W-0h       |        |              | R/W-0h |              | R/W-0h |
| 23           | 22 | 21           | 20     | 19           | 18     | 17           | 16     |
| RX12_AUTOREQ |    | RX11_AUTOREQ |        | RX10_AUTOREQ |        | RX9_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |
| 15           | 14 | 13           | 12     | 11           | 10     | 9            | 8      |
| RX8_AUTOREQ  |    | RX7_AUTOREQ  |        | RX6_AUTOREQ  |        | RX5_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |
| 7            | 6  | 5            | 4      | 3            | 2      | 1            | 0      |
| RX4_AUTOREQ  |    | RX3_AUTOREQ  |        | RX2_AUTOREQ  |        | RX1_AUTOREQ  |        |
| R/W-0h       |    |              | R/W-0h |              |        | R/W-0h       |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-138. USB1AUTOREQ Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 29-28 | RX15_AUTOREQ | R/W  | 0h    | RX endpoint 14 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 27-26 | RX14_AUTOREQ | R/W  | 0h    | RX endpoint 13 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 25-24 | RX13_AUTOREQ | R/W  | 0h    | RX endpoint 12 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 23-22 | RX12_AUTOREQ | R/W  | 0h    | RX endpoint 11 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 21-20 | RX11_AUTOREQ | R/W  | 0h    | RX endpoint 10 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always |
| 19-18 | RX10_AUTOREQ | R/W  | 0h    | RX endpoint 9 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |
| 17-16 | RX9_AUTOREQ  | R/W  | 0h    | RX endpoint 8 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |
| 15-14 | RX8_AUTOREQ  | R/W  | 0h    | RX endpoint 7 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always  |

**Table 16-138. USB1AUTOREQ Register Field Descriptions (continued)**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 13-12 | RX7_AUTOREQ | R/W  | 0h    | RX endpoint 6 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 11-10 | RX6_AUTOREQ | R/W  | 0h    | RX endpoint 5 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 9-8   | RX5_AUTOREQ | R/W  | 0h    | RX endpoint 4 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 7-6   | RX4_AUTOREQ | R/W  | 0h    | RX endpoint 3 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 5-4   | RX3_AUTOREQ | R/W  | 0h    | RX endpoint 2 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 3-2   | RX2_AUTOREQ | R/W  | 0h    | RX endpoint 1 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always                         |
| 1-0   | RX1_AUTOREQ | R/W  | 0h    | RX endpoint 0 Auto Req enable<br>00 = no auto req<br>01 = auto req on all but EOP<br>10 = reserved<br>11 = auto req always USB1 Auto Req Registers |

**16.5.3.31 USB1SRPFIXTIME Register (offset = D4h) [reset = 280DE80h]**

USB1SRPFIXTIME is shown in [Figure 16-129](#) and described in [Table 16-139](#).

**Figure 16-129. USB1SRPFIXTIME Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| SRPFIXTIME   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-280DE80h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

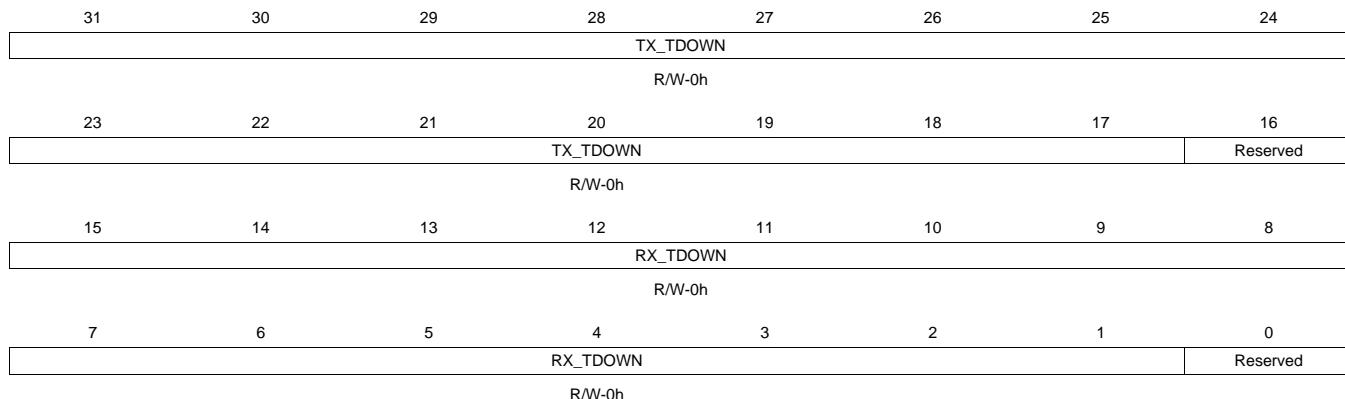
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-139. USB1SRPFIXTIME Register Field Descriptions**

| Bit  | Field      | Type | Reset    | Description  |
|------|------------|------|----------|--|
| 31-0 | SRPFIXTIME | R/W  | 280DE80h | SRP Fix maximum time in 60 MHz cycles.<br>Default is 700 ms.<br>USB1 SRP Fix Time Register |

**16.5.3.32 USB1TDOWN Register (offset = D8h) [reset = 0h]**

USB1TDOWN is shown in [Figure 16-130](#) and described in [Table 16-140](#).

**Figure 16-130. USB1TDOWN Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-140. USB1TDOWN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-17 | TX_TDOWN | R/W  | 0h    | Tx Endpoint Teardown.<br>Write '1' to corresponding bit to set.<br>Read as '0'.<br>Bit<br>31 = Endpoint 15 ...<br>Bit<br>17 = Endpoint 1 |
| 15-1  | RX_TDOWN | R/W  | 0h    | RX Endpoint Teardown.<br>Write '1' to corresponding bit to set.<br>Read as '0'.<br>Bit<br>15 = Endpoint 15 ...<br>Bit<br>1 = Endpoint 1  |

**16.5.3.33 USB1UTMI Register (offset = E0h) [reset = 200002h]**

USB1UTMI is shown in [Figure 16-131](#) and described in [Table 16-141](#).

**Figure 16-131. USB1UTMI Register**

|              |               |            |               |               |           |             |          |
|--------------|---------------|------------|---------------|---------------|-----------|-------------|----------|
| 31           | 30            | 29         | 28            | 27            | 26        | 25          | 24       |
| Reserved     |               |            |               |               |           |             |          |
| 23           | 22            | 21         | 20            | 19            | 18        | 17          | 16       |
| TXBITSTUFFEN | TXBITSTUFFENH | OTGDISABLE | VBUSVLDEXTSEL | VBUSVLDEXT    | TXENABLEN | FSXCVROWNER | TXVALIDH |
| R/W-0h       | R/W-0h        | R/W-1h     | R/W-0h        | R/W-0h        | R/W-0h    | R/W-0h      | R/W-0h   |
| 15           | 14            | 13         | 12            | 11            | 10        | 9           | 8        |
| DATAINH      |               |            |               |               |           |             |          |
| R/W-0h       |               |            |               |               |           |             |          |
| 7            | 6             | 5          | 4             | 3             | 2         | 1           | 0        |
| Reserved     |               |            |               | WORDINTERFACE |           | FSDATAEXT   | FSSE0EXT |
|              |               |            |               | R/W-0h        |           | R/W-1h      | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-141. USB1UTMI Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 23   | TXBITSTUFFEN  | R/W  | 0h    | PHY UTMI input for signal txbitstuffen                    |
| 22   | TXBITSTUFFENH | R/W  | 0h    | PHY UTMI input for signal txbitstuffenh                   |
| 21   | OTGDISABLE    | R/W  | 1h    | PHY UTMI input for signal otgdisable                      |
| 20   | VBUSVLDEXTSEL | R/W  | 0h    | PHY UTMI input for signal vbusvldextsel                   |
| 19   | VBUSVLDEXT    | R/W  | 0h    | PHY UTMI input for signal vbusvldext                      |
| 18   | TXENABLEN     | R/W  | 0h    | PHY UTMI input for signal txenablen                       |
| 17   | FSXCVROWNER   | R/W  | 0h    | PHY UTMI input for signal fsxcvrowner                     |
| 16   | TXVALIDH      | R/W  | 0h    | PHY UTMI input for signal txvalidh                        |
| 15-8 | DATAINH       | R/W  | 0h    | PHY UTMI input for signal datainh                         |
| 2    | WORDINTERFACE | R/W  | 0h    | PHY UTMI input for signal wordinterface                   |
| 1    | FSDATAEXT     | R/W  | 1h    | PHY UTMI input for signal fsdataext                       |
| 0    | FSSE0EXT      | R/W  | 0h    | PHY UTMI input for signal fsse0ext USB1 PHY UTMI Register |

**16.5.3.34 USB1UTMILB Register (offset = E4h) [reset = 82h]**

 USB1UTMILB is shown in [Figure 16-132](#) and described in [Table 16-142](#).

**Figure 16-132. USB1UTMILB Register**

| 31        | 30       | 29       | 28       | 27          | 26         | 25         | 24       |
|-----------|----------|----------|----------|-------------|------------|------------|----------|
|           | Reserved |          | SUSPENDM |             | OPMODE     | TXVALID    | XCVRSEL  |
|           |          |          | R-0h     |             | R-0h       | R-0h       | R-0h     |
| 23        | 22       | 21       | 20       | 19          | 18         | 17         | 16       |
| XCVRSEL   | TERMSEL  | DRVVBUS  | CHRGVBUS | DISCHRGVBUS | DPPULLDOWN | DMPULLDOWN | IDPULLUP |
| R-0h      | R-0h     | R-0h     | R-0h     | R-0h        | R-0h       | R-0h       | R-0h     |
| 15        | 14       | 13       | 12       | 11          | 10         | 9          | 8        |
|           | Reserved |          |          | IDDIG       | HOSTDISCON | SESEND     | AVALID   |
|           |          |          |          | R/W-0h      | R/W-0h     | R/W-0h     | R/W-0h   |
| 7         | 6        | 5        | 4        | 3           | 2          | 1          | 0        |
| VBUSVALID | RXERROR  | Reserved |          | LINESTATE   |            | Reserved   |          |
|           |          |          |          | R/W-0h      |            |            |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-142. USB1UTMILB Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description                                  |
|-------|-------------|------|-------|--|
| 28    | SUSPENDM    | R    | 0h    | loopback test observed value for suspendm    |
| 27-26 | OPMODE      | R    | 0h    | loopback test observed value for opmode      |
| 25    | TXVALID     | R    | 0h    | loopback test observed value for txvalid     |
| 24-23 | XCVRSEL     | R    | 0h    | loopback test observed value for xcvsel      |
| 22    | TERMSEL     | R    | 0h    | loopback test observed value for termsel     |
| 21    | DRVVBUS     | R    | 0h    | loopback test observed value for drvvbus     |
| 20    | CHRGVBUS    | R    | 0h    | loopback test observed value for chrgvbus    |
| 19    | DISCHRGVBUS | R    | 0h    | loopback test observed value for dischrgvbus |
| 18    | DPPULLDOWN  | R    | 0h    | loopback test observed value for dppulldown  |
| 17    | DMPULLDOWN  | R    | 0h    | loopback test observed value for dmpulldown  |
| 16    | IDPULLUP    | R    | 0h    | loopback test observed value for idpullup    |
| 11    | IDDIG       | R/W  | 0h    | loopback test value for iddig                |
| 10    | HOSTDISCON  | R/W  | 0h    | loopback test value for hostdiscon           |
| 9     | SESEND      | R/W  | 0h    | loopback test value for sessend              |
| 8     | AVALID      | R/W  | 0h    | loopback test value for avalid               |
| 7     | VBUSVALID   | R/W  | 1h    | loopback test value for vbusvalid            |
| 6     | RXERROR     | R/W  | 0h    | loopback test value for rxerror              |
| 3-2   | LINESTATE   | R/W  | 0h    | loopback test value for linestate            |

### 16.5.3.35 USB1MODE Register (offset = E8h) [reset = 100h]

USB1MODE is shown in [Figure 16-133](#) and described in [Table 16-143](#).

**Figure 16-133. USB1MODE Register**

|           |    |          |    |    |          |       |          |
|-----------|----|----------|----|----|----------|-------|----------|
| 31        | 30 | 29       | 28 | 27 | 26       | 25    | 24       |
| Reserved  |    |          |    |    |          |       |          |
| 23        | 22 | 21       | 20 | 19 | 18       | 17    | 16       |
| Reserved  |    |          |    |    |          |       |          |
| 15        | 14 | 13       | 12 | 11 | 10       | 9     | 8        |
| Reserved  |    |          |    |    |          | IDDIG |          |
| R/W-1h    |    |          |    |    |          |       |          |
| 7         | 6  | 5        | 4  | 3  | 2        | 1     | 0        |
| IDDIG_MUX |    | Reserved |    |    | PHY_TEST |       | LOOPBACK |
| R/W-0h    |    | R/W-0h   |    |    | R/W-0h   |       |          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-143. USB1MODE Register Field Descriptions**

| Bit | Field     | Type | Reset | Description   |
|-----|-----------|------|-------|---|
| 8   | IDDIG     | R/W  | 1h    | MGC input value for iddig<br>0=A type<br>1=B type   |
| 7   | IDDIG_MUX | R/W  | 0h    | Multiplexer control for IDDIG signal going to the controller.<br>0 = IDDIG is from PHY1.<br>1 = IDDIG is from bit8 (IDDIG) of this USB1MODE register. |
| 1   | PHY_TEST  | R/W  | 0h    | PHY test<br>0 = Normal mode<br>1 = PHY test mode  |
| 0   | LOOPBACK  | R/W  | 0h    | Loopback test mode<br>0 = Normal mode<br>1 = Loopback test mode USB1 Mode Register  |

### 16.5.4 USB2PHY Registers

[Table 16-144](#) lists the memory-mapped registers for the USB2PHY. All register offset addresses not listed in [Table 16-144](#) should be considered as reserved locations and the register contents should not be modified.

**Table 16-144. USB2PHY REGISTERS**

| Offset | Acronym               | Register Name | Section                           |
|--------|-----------------------|---------------|-----------------------------------|
| 0h     | Termination_control   |               | <a href="#">Section 16.5.4.1</a>  |
| 4h     | RX_CALIB              |               | <a href="#">Section 16.5.4.2</a>  |
| 8h     | DLLHS_2               |               | <a href="#">Section 16.5.4.3</a>  |
| Ch     | RX_TEST_2             |               | <a href="#">Section 16.5.4.4</a>  |
| 14h    | CHRG_DET              |               | <a href="#">Section 16.5.4.5</a>  |
| 18h    | PWR_CNTL              |               | <a href="#">Section 16.5.4.6</a>  |
| 1Ch    | UTMI_INTERFACE_CNTL_1 |               | <a href="#">Section 16.5.4.7</a>  |
| 20h    | UTMI_INTERFACE_CNTL_2 |               | <a href="#">Section 16.5.4.8</a>  |
| 24h    | BIST                  |               | <a href="#">Section 16.5.4.9</a>  |
| 28h    | BIST_CRC              |               | <a href="#">Section 16.5.4.10</a> |
| 2Ch    | CDR_BIST2             |               | <a href="#">Section 16.5.4.11</a> |
| 30h    | GPIO                  |               | <a href="#">Section 16.5.4.12</a> |
| 34h    | DLLHS                 |               | <a href="#">Section 16.5.4.13</a> |

**Table 16-144. USB2PHY REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section           |
|--------|-------------------|---------------|-------------------|
| 3Ch    | USB2PHYCM_CONFIG  |               | Section 16.5.4.14 |
| 44h    | AD_INTERFACE_REG1 |               | Section 16.5.4.15 |
| 48h    | AD_INTERFACE_REG2 |               | Section 16.5.4.16 |
| 4Ch    | AD_INTERFACE_REG3 |               | Section 16.5.4.17 |
| 54h    | ANA_CONFIG2       |               | Section 16.5.4.18 |

#### 16.5.4.1 Termination\_control Register (offset = 0h) [reset = 1000800h]

Termination\_control is shown in [Figure 16-134](#) and described in [Table 16-145](#).

contains bits related to control of terminations in USB2PHY

**Figure 16-134. Termination\_control Register**

|                   |                   |                |           |                |                   |                    |    |
|-------------------|-------------------|----------------|-----------|----------------|-------------------|--------------------|----|
| 31                | 30                | 29             | 28        | 27             | 26                | 25                 | 24 |
| Reserved          | ALWAYS_UPDATE     | RTERM_CAL_DONE |           | FS_CODE_SEL    |                   |                    |    |
| R/W-0h            | R/W-0h            | R-0h           |           | R/W-1h         |                   |                    |    |
| 23                | 22                | 21             | 20        | 19             | 18                | 17                 | 16 |
| Reserved          | USE_RTERM_RMX_REG |                | RTERM_RMX |                |                   |                    |    |
| R/W-0h            | R/W-0h            |                | R/W-0h    |                |                   |                    |    |
| 15                | 14                | 13             | 12        | 11             | 10                | 9                  | 8  |
| RTERM_RMX         |                   | HS_CODE_SEL    |           | RTERM_COMP_OUT | RESTART_RTERM_CAL | DISABLE_TEMP_TRACK |    |
| R/W-0h            |                   | R/W-1h         |           | R-0h           | R/W-0h            | R/W-0h             |    |
| 7                 | 6                 | 5              | 4         | 3              | 2                 | 1                  | 0  |
| USE_RTERM_CAL_REG |                   |                | RTERM_CAL |                |                   |                    |    |
| R/W-0h            |                   |                | R/W-0h    |                |                   |                    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-145. Termination\_control Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description  |
|-------|--------------------|------|-------|--|
| 31-30 | Reserved           | R/W  | 0h    |  |
| 29    | ALWAYS_UPDATE      | R/W  | 0h    | When set to 1 , the calibration code is updated immediately after a code computation without waiting for idle periods.   |
| 28    | RTERM_CAL_DONE     | R    | 0h    | Rterm calibration is done.<br>1st time cal is done this bit gets set and gets reset at a restart cal. Read value is valid only if VDDLDO is on.  |
| 27-24 | FS_CODE_SEL        | R/W  | 1h    | FS Code selection control  |
| 23-22 | Reserved           | R/W  | 0h    |  |
| 21    | USE_RTERM_RMX_REG  | R/W  | 0h    | Override termination resistor trim code with RTERM_RMX from this register  |
| 20-14 | RTERM_RMX          | R/W  | 0h    | When read, this field returns the current Termination resistor trim code. Read value is valid only if VDDLDO is on.<br>The value written to this field is used as Termination resistor trim code if bit 21 is set to 1 |
| 13-11 | HS_CODE_SEL        | R/W  | 1h    | HS Code selection control.<br>Each increment increases the termination impedance by ~1.5%. Valid values are 000b 011b.   |
| 10    | RTERM_COMP_OUT     | R    | 0h    | Master loop comparator output.<br>Read value is valid only if VDDLDO is on.  |
| 9     | RESTART_RTERM_CAL  | R/W  | 0h    | Restart the rterm calibration.<br>The calibration restarts on any toggle 0 to 1 or 1 to 0 on this bit.   |
| 8     | DISABLE_TEMP_TRACK | R/W  | 0h    | Disables the temperature tracking function of the termination calibration  |
| 7     | USE_RTERM_CAL_REG  | R/W  | 0h    | when '1' the rterm cal code is overridden by values in RTERM_CAL   |
| 6-0   | RTERM_CAL          | R/W  | 0h    | When read this field returns the current rterm calibration code. Read value is valid only if VDDLDO is on.<br>The value written to this field is used as rterm calibration code if the bit USE_RTERM_CAL_REG is 1.     |

#### 16.5.4.2 RX\_CALIB Register (offset = 4h) [reset = 0h]

RX\_CALIB is shown in [Figure 16-135](#) and described in [Table 16-146](#).

Contains bits related to RX calibration

**Figure 16-135. RX\_CALIB Register**

| 31               | 30              | 29               | 28 | 27               | 26               | 25              | 24 |
|------------------|-----------------|------------------|----|------------------|------------------|-----------------|----|
| RESTART_HSRX_CAL | USE_HS_OFF_REG  |                  |    |                  | HS_OFF_CODE      |                 |    |
| R/W-0h           | R/W-0h          |                  |    |                  | R/W-0h           |                 |    |
| 23               | 22              | 21               | 20 | 19               | 18               | 17              | 16 |
| HSRX_COMP_OUT    | HSRX_CAL_DONE   | USE_SQ_OFF_DAC1  |    |                  | SQ_OFF_CODE_DAC1 |                 |    |
| R-0h             | R-0h            | R/W-0h           |    |                  | R/W-0h           |                 |    |
| 15               | 14              | 13               | 12 | 11               | 10               | 9               | 8  |
| SQ_OFF_CODE_DAC1 | USE_SQ_OFF_DAC2 |                  |    | SQ_OFF_CODE_DAC2 |                  | USE_SQ_OFF_DAC3 |    |
| R/W-0h           | R/W-0h          |                  |    | R/W-0h           |                  | R/W-0h          |    |
| 7                | 6               | 5                | 4  | 3                | 2                | 1               | 0  |
|                  |                 | SQ_OFF_CODE_DAC3 |    | SQ_COMP_OUT      | SQ_CAL_DONE      | RESTART_SQ_CAL  |    |
|                  |                 | R/W-0h           |    | R-0h             | R-0h             | R/W-0h          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-146. RX\_CALIB Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31    | RESTART_HSRX_CAL | R/W  | 0h    | Restart the HSRX calibration state machine when this bit goes from 0 to 1.   |
| 30    | USE_HS_OFF_REG   | R/W  | 0h    | Override HS offset correction with HS_OFF_CODE when set to '1'   |
| 29-24 | HS_OFF_CODE      | R/W  | 0h    | HS offset code, this code is forced when bit 30 is 1 .<br>Code is updated from calibration logic when bit 30 = 0.                                      |
| 23    | HSRX_COMP_OUT    | R    | 0h    | The output of the HSRX comparator.Read value is valid only if VDDLDO is on.  |
| 22    | HSRX_CAL_DONE    | R    | 0h    | Signal that indicates that the HSRX calibration is done.<br>This gets reset at every restart.Read value is valid only if VDDLDO is on.                 |
| 21    | USE_SQ_OFF_DAC1  | R/W  | 0h    | Override Squelch offset DAC1 code when '1'   |
| 20-15 | SQ_OFF_CODE_DAC1 | R/W  | 0h    | When read returns current Sq offset code for DAC1, if VDDLDO is on.<br>When written this is used as Sq offset code for DAC1 when USE_SQ_OFF_DAC1 = '1' |
| 14    | USE_SQ_OFF_DAC2  | R/W  | 0h    | Override Squelch offset DAC2 code when '1'   |
| 13-9  | SQ_OFF_CODE_DAC2 | R/W  | 0h    | When read returns current Sq offset code for DAC2, if VDDLDO is on.<br>When written this is used as Sq offset code for DAC2 when USE_SQ_OFF_DAC2 = '1' |
| 8     | USE_SQ_OFF_DAC3  | R/W  | 0h    | Override Squelch offset DAC3 code when '1'   |
| 7-3   | SQ_OFF_CODE_DAC3 | R/W  | 0h    | When read returns current Sq offset code for DAC3, if VDDLDO is on.<br>When written this is used as Sq offset code for DAC3 when USE_SQ_OFF_DAC3 = '1' |
| 2     | SQ_COMP_OUT      | R    | 0h    | Sq comp output.Read value is valid only if VDDLDO is on.   |

**Table 16-146. RX\_CALIB Register Field Descriptions (continued)**

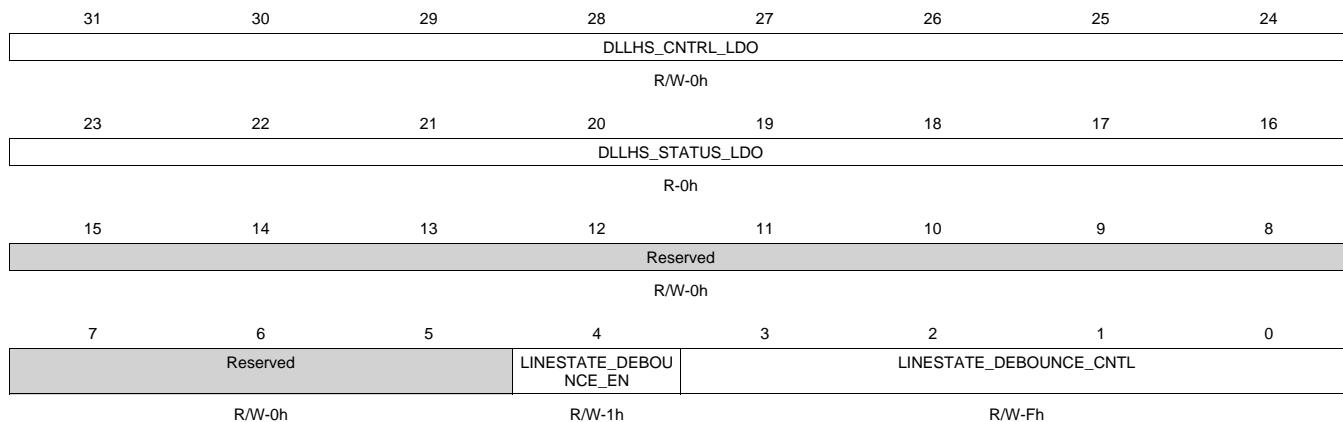
| Bit | Field          | Type | Reset | Description   |
|-----|----------------|------|-------|---|
| 1   | SQ_CAL_DONE    | R    | 0h    | Sq calibration is done when this bit = 1.<br>see RESTART_SQ_CAL for more description.Read value is valid only if VDDLDO is on.                    |
| 0   | RESTART_SQ_CAL | R/W  | 0h    | the squelch calibration continuously goes through restart cycles when this bit is 1 .<br>i.e.<br>restarts waits for done then restarts again etc. |

#### 16.5.4.3 DLLHS\_2 Register (offset = 8h) [reset = 1Fh]

DLLHS\_2 is shown in [Figure 16-136](#) and described in [Table 16-147](#).

the 2nd DLLHS control register. bits 4:0 are unrelated to the DLLHS and are linestate filter settings

**Figure 16-136. DLLHS\_2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-147. DLLHS\_2 Register Field Descriptions**

| Bit   | Field                       | Type | Reset | Description   |
|-------|-----------------------------|------|-------|---|
| 31-24 | DLLHS_CNTRL_LDO             | R/W  | 0h    | See DFT spec for details  |
| 23-16 | DLLHS_STATUS_LDO            | R    | 0h    | See DFT spec for details  |
| 15-5  | Reserved                    | R/W  | 0h    |   |
| 4     | LINESTATE_DEBOUNCE<br>_EN   | R/W  | 1h    | Enables the linestate debounce filter   |
| 3-0   | LINESTATE_DEBOUNCE<br>_CNTL | R/W  | Fh    | Used for control of the linestate debounce filter when going from synchronous to async linestate. |

#### 16.5.4.4 RX\_TEST\_2 Register (offset = Ch) [reset = 0h]

RX\_TEST\_2 is shown in [Figure 16-137](#) and described in [Table 16-148](#).

the 2nd receiver test register

**Figure 16-137. RX\_TEST\_2 Register**

| 31            | 30               | 29                 | 28       | 27            | 26            | 25     | 24 |
|---------------|------------------|--------------------|----------|---------------|---------------|--------|----|
| HSOSREVERSAL  | HSOSBITINVERSION | PHYCLKOUTINVERSION | RXPIDERR | USEINTDATAOUT | INTDATAOUTREG |        |    |
| R/W-0h        | R/W-0h           | R/W-0h             | R-0h     | R/W-0h        |               | R/W-0h |    |
| 23            | 22               | 21                 | 20       | 19            | 18            | 17     | 16 |
| INTDATAOUTREG |                  |                    |          |               |               |        |    |
| R/W-0h        |                  |                    |          |               |               |        |    |
| 15            | 14               | 13                 | 12       | 11            | 10            | 9      | 8  |
| INTDATAOUTREG |                  |                    |          |               | Reserved      |        |    |
| R/W-0h        |                  |                    |          |               |               |        |    |
| 7             | 6                | 5                  | 4        | 3             | 2             | 1      | 0  |
| CDR_TESTOUT   |                  |                    |          |               |               |        |    |
| R-0h          |                  |                    |          |               |               |        |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-148. RX\_TEST\_2 Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description   |
|-------|--------------------|------|-------|---|
| 31    | HSOSREVERSAL       | R/W  | 0h    | Swaps the dataout from HSOS   |
| 30    | HSOSBITINVERSION   | R/W  | 0h    | Inverts the HSOS bits   |
| 29    | PHYCLKOUTINVERSION | R/W  | 0h    | This inverts the phase for the PHYCLKOUT  |
| 28    | RXPIDERR           | R    | 0h    | Flags if the RX data packet has PID error.<br>NOT IMPLEMENTED YET   |
| 27    | USEINTDATAOUT      | R/W  | 0h    | This will bypass the analog and will send data packet to controller<br>incase of receiver (Faking the receive data).<br>data used will be INTDATAOUTREG |
| 26-11 | INTDATAOUTREG      | R/W  | 0h    | This register will be loaded through OCP and this data will be given<br>to the controller if USEINTDATAOUT is set to 1                                  |
| 10-8  | Reserved           | R/W  | 0h    |   |
| 7-0   | CDR_TESTOUT        | R    | 0h    | CDR debug bits.<br>Read value is valid only if VDDLDO is on.<br>see DFT spec for details  |

#### 16.5.4.5 CHRG\_DET Register (offset = 14h) [reset = 0h]

CHRG\_DET is shown in [Figure 16-138](#) and described in [Table 16-149](#).

this is the charger detect register. this register is not used in the dead battery case.

**Figure 16-138. CHRG\_DET Register**

| 31            | 30                | 29          | 28            | 27            | 26              | 25              | 24     |
|---------------|-------------------|-------------|---------------|---------------|-----------------|-----------------|--------|
| Reserved      | USE_CHG_DET_REG   | DIS_CHG_DET | SRC_ON_DM     | SINK_ON_DP    | CHG_DET_EXT_CTL | RESTART_CHG_DET |        |
| R/W-0h        | R/W-0h            | R/W-0h      | R/W-0h        | R/W-0h        | R/W-0h          | R/W-0h          | R/W-0h |
| 23            | 22                | 21          | 20            | 19            | 18              | 17              | 16     |
| CHG_DET_DONE  | CHG_DETECTED      | DATA_DET    | Reserved      | CHG_ISINK_EN  | CHG_VSRC_EN     | COMP_DP         |        |
| R-0h          | R-0h              | R-0h        | R/W-0h        | R/W-0h        | R/W-0h          | R/W-0h          | R-0h   |
| 15            | 14                | 13          | 12            | 11            | 10              | 9               | 8      |
| COMP_DM       | CHG_DET_OSC_CNTRL |             |               | CHG_DET_TIMER |                 |                 |        |
| R-0h          | R/W-0h            |             |               | R/W-0h        |                 |                 |        |
| 7             | 6                 | 5           | 4             | 3             | 2               | 1               | 0      |
| CHG_DET_TIMER | Reserved          |             | CHG_DET_ICTRL | CHG_DET_VCTRL | FOR_CE          |                 |        |
| R/W-0h        | R/W-0h            |             | R/W-0h        | R/W-0h        | R/W-0h          |                 | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-149. CHRG\_DET Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 31-30 | Reserved          | R/W  | 0h    |  |
| 29    | USE_CHG_DET_REG   | R/W  | 0h    | Use bits 28-24 and 18-17 from this register  |
| 28    | DIS_CHG_DET       | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 27    | SRC_ON_DM         | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 26    | SINK_ON_DP        | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 25    | CHG_DET_EXT_CTL   | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 24    | RESTART_CHG_DET   | R/W  | 0h    | Restart the charger detection protocol when this goes from 0 to 1  |
| 23    | CHG_DET_DONE      | R    | 0h    | Charger detect protocol has completed  |
| 22    | CHG_DETECTED      | R    | 0h    | Same signal as CE  |
| 21    | DATA_DET          | R    | 0h    | Output of the data det comparator  |
| 20-19 | Reserved          | R/W  | 0h    |  |
| 18    | CHG_ISINK_EN      | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 17    | CHG_VSRC_EN       | R/W  | 0h    | When read, returns current value of charger detect input. When USE_CHG_DET_REG = 1 , the value written to this field overrides the corresponding charger detect input. |
| 16    | COMP_DP           | R    | 0h    | Comparator on the DP line value  |
| 15    | COMP_DM           | R    | 0h    | Comparator on the DM line value  |
| 14-13 | CHG_DET_OSC_CNTRL | R/W  | 0h    | Charger detect osc control   |

**Table 16-149. CHRG\_DET Register Field Descriptions (continued)**

| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 12-7 | CHG_DET_TIMER | R/W  | 0h    | Charger detect timer control.<br>See charger detect section for details |
| 6-5  | Reserved      | R/W  | 0h    |   |
| 4-3  | CHG_DET_ICTRL | R/W  | 0h    | Charger detect current control  |
| 2-1  | CHG_DET_VCTRL | R/W  | 0h    | Charger detect voltage buffer control                                   |
| 0    | FOR_CE        | R/W  | 0h    | Force CE = 1 when this bit is set                                       |

#### 16.5.4.6 PWR\_CNTL Register (offset = 18h) [reset = 400000h]

PWR\_CNTL is shown in [Figure 16-139](#) and described in [Table 16-150](#).

has all the power control bits

**Figure 16-139. PWR\_CNTL Register**

| 31            | 30               | 29                      | 28                    | 27               | 26            | 25         | 24       |
|---------------|------------------|-------------------------|-----------------------|------------------|---------------|------------|----------|
| RESETDONETCLK | RESET_DONE_VMAIN | VMAIN_GLOBAL_RESET_DONE | RESETDONEMCLK         | RESETDONE_CHGDET | LDOPWRCOUNTER |            |          |
| R-0h          | R-0h             | R-0h                    | R-0h                  | R-0h             | R/W-400h      |            |          |
| LDOPWRCOUNTER |                  |                         |                       |                  |               |            |          |
| 23            | 22               | 21                      | 20                    | 19               | 18            | 17         | 16       |
| R/W-400h      |                  |                         |                       |                  |               |            |          |
| 15            | 14               | 13                      | 12                    | 11               | 10            | 9          | 8        |
| LDOPWRCOUNTER |                  |                         |                       | FORCEPLLSLOWCLK  | FORCELDOON    | FORCEPLLON | Reserved |
| R/W-400h      |                  |                         |                       | R/W-0h           | R/W-0h        | R/W-0h     | R/W-0h   |
| 7             | 6                | 5                       | 4                     | 3                | 2             | 1          | 0        |
| Reserved      | PLLLOCK          | USEPLLLOCK              | USE_DATAPOLARITYN_REG | DATAPOLARITYN    | USE_PD_REG    | PD         | Reserved |
| R/W-0h        | R-0h             | R/W-0h                  | R/W-0h                | R/W-0h           | R/W-0h        | R/W-0h     | R/W-0h   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-150. PWR\_CNTL Register Field Descriptions**

| Bit   | Field                   | Type | Reset | Description  |
|-------|-------------------------|------|-------|--|
| 31    | RESETDONETCLK           | R    | 0h    | Goes high when the RESET is synchronized to TCLK   |
| 30    | RESET_DONE_VMAIN        | R    | 0h    | Goes high when LDO domain is up and PLL LOCK is available and utmi_reset is de-asserted.   |
| 29    | VMAIN_GLOBAL_RESET_DONE | R    | 0h    | Goes high when LDO domain is up and PLL LOCK is available.   |
| 28    | RESETDONEMCLK           | R    | 0h    | Goes high when the RESET is synchronized to MCLK   |
| 27    | RESETDONE_CHGDET        | R    | 0h    | Goes high when the RESET is synchronized to charger detect oscillator clock domain   |
| 26-12 | LDOPWRCOUNTER           | R/W  | 400h  | This is the value of the counter used for LDO power up. RESET to default.  |
| 11    | FORCEPLLSLOWCLK         | R/W  | 0h    | Forces the PLL to the slow clk mode  |
| 10    | FORCELDOON              | R/W  | 0h    | Forces the LDO to be ON.   |
| 9     | FORCEPLLON              | R/W  | 0h    | Forces the PLL to be ON.   |
| 8-7   | Reserved                | R/W  | 0h    |  |
| 6     | PLLLOCK                 | R    | 0h    | Lock signal from the PLL   |
| 5     | USEPLLLOCK              | R/W  | 0h    | This signal is used to indicate to the Phy, not to do any clock related activity until PLLLOCK = 1 .This is not the default option.0 do not use PLLLOCK.<br>1 use PLLLOCK as a clock gate. |
| 4     | USE_DATAPOLARITYN_REG   | R/W  | 0h    | 1 -andgt; use bit 3 as override for the DATAPOLARITYN signal.  |
| 3     | DATAPOLARITYN           | R/W  | 0h    | Override value of datapolarityn  |
| 2     | USE_PD_REG              | R/W  | 0h    | Use bit 1 from this register as PD override when set to '1'  |
| 1     | PD                      | R/W  | 0h    | Override value for PD  |
| 0     | Reserved                | R/W  | 0h    |  |

**16.5.4.7 UTMI\_INTERFACE\_CNTL\_1 Register (offset = 1Ch) [reset = 0h]**

UTMI\_INTERFACE\_CNTL\_1 is shown in [Figure 16-140](#) and described in [Table 16-151](#).

register to override UTMI interface control pins.

**Figure 16-140. UTMI\_INTERFACE\_CNTL\_1 Register**

| 31             | 30         | 29              | 28           | 27           | 26            | 25      | 24               |
|----------------|------------|-----------------|--------------|--------------|---------------|---------|------------------|
| USEUTMIDATAREG | UTMIDATAIN |                 |              |              |               |         |                  |
| R/W-0h         | R/W-0h     |                 |              |              |               |         |                  |
| 23             | 22         | 21              | 20           | 19           | 18            | 17      | 16               |
| UTMIDATAIN     |            |                 |              |              |               |         |                  |
| R/W-0h         | R/W-0h     |                 |              |              |               |         |                  |
| 15             | 14         | 13              | 12           | 11           | 10            | 9       | 8                |
| UTMIDATAIN     | Reserved   | USEDATABUSREG   | DATABUS16OR8 | USEOPMODEREG | OPMODE        |         | OVERRIDESUSRESET |
| R/W-0h         | R/W-0h     | R/W-0h          | R/W-0h       | R/W-0h       | R/W-0h        |         | R/W-0h           |
| 7              | 6          | 5               | 4            | 3            | 2             | 1       | 0                |
| SUSPENDM       | UTMIRESET  | OVERRIDEXCVRSEL | XCVRSEL      |              | USETXVALIDREG | TXVALID | TXVALIDH         |
| R/W-0h         | R/W-0h     | R/W-0h          | R/W-0h       |              | R/W-0h        | R/W-0h  | R/W-0h           |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-151. UTMI\_INTERFACE\_CNTL\_1 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description   |
|-------|------------------|------|-------|---|
| 31    | USEUTMIDATAREG   | R/W  | 0h    | Use datain from UTMI interface register                       |
| 30-15 | UTMIDATAIN       | R/W  | 0h    | Override value for the UTMIDATAIN                             |
| 14    | Reserved         | R/W  | 0h    |   |
| 13    | USEDATABUSREG    | R/W  | 0h    | When set to 1 use bit 12 from register instead of interface   |
| 12    | DATABUS16OR8     | R/W  | 0h    | Override value for UTMI signal DATABUS16OR8                   |
| 11    | USEOPMODEREG     | R/W  | 0h    | When set to 1 use bit 10-9 from register instead of interface |
| 10-9  | OPMODE           | R/W  | 0h    | Override value for UTMI signal OPMODE [1:0]                   |
| 8     | OVERRIDESUSRESET | R/W  | 0h    | Override the suspend and reset values. Use bits 6 and 7       |
| 7     | SUSPENDM         | R/W  | 0h    | Override value for UTMI signal SUSPENDM                       |
| 6     | UTMIRESET        | R/W  | 0h    | Override value for UTMI signal UTMIRESET                      |
| 5     | OVERRIDEXCVRSEL  | R/W  | 0h    | When set to 1 use bit 4-3 from register instead of interface  |
| 4-3   | XCVRSEL          | R/W  | 0h    | Override value for UTMI signal XCVRSEL [1:0]                  |
| 2     | USETXVALIDREG    | R/W  | 0h    | When set to 1 use bit 1-0 from register instead of interface  |
| 1     | TXVALID          | R/W  | 0h    | Override value for UTMI signal TXVALID                        |
| 0     | TXVALIDH         | R/W  | 0h    | Override value for UTMI signal TXVALIDH                       |

#### 16.5.4.8 UTMI\_INTERFACE\_CNTL\_2 Register (offset = 20h) [reset = 0h]

UTMI\_INTERFACE\_CNTL\_2 is shown in [Figure 16-141](#) and described in [Table 16-152](#).

UTMI interface override and observe register 2

**Figure 16-141. UTMI\_INTERFACE\_CNTL\_2 Register**

| 31       | 30            | 29         | 28             | 27             | 26               | 25                 | 24                |
|----------|---------------|------------|----------------|----------------|------------------|--------------------|-------------------|
| RXRCV    | RXDP          | RXDM       | HOSTDISCONNECT | LINESTATE      | RXVALID          | RXVALIDH           |                   |
| R-0h     | R-0h          | R-0h       | R-0h           | R-0h           | R-0h             | R-0h               | R-0h              |
| 23       | 22            | 21         | 20             | 19             | 18               | 17                 | 16                |
| RXACTIVE | RXERROR       | TXREADY    | UTMIRESETDONE  | USEBITSTUFFREG | TXBITSTUFFENABLE | TXBITSTUFFENABLEH  | USETERMCONTROLREG |
| R-0h     | R-0h          | R-0h       | R-0h           | R/W-0h         | R/W-0h           | R/W-0h             | R/W-0h            |
| 15       | 14            | 13         | 12             | 11             | 10               | 9                  | 8                 |
| TERMSEL  | DPPULLDOWN    | DMPULLDOWN |                | Reserved       | USEREGSERIALMODE | TXSE0              |                   |
| R/W-0h   | R/W-0h        | R/W-0h     |                | R/W-0h         | R/W-0h           | R/W-0h             | R/W-0h            |
| 7        | 6             | 5          | 4              | 3              | 2                | 1                  | 0                 |
| TXDAT    | FSLSERIALMODE | TXENABLEN  |                | Reserved       |                  | sig_bypass_suspend |                   |
| R/W-0h   | R/W-0h        | R/W-0h     |                | R/W-0h         |                  | mpulse_incr        |                   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-152. UTMI\_INTERFACE\_CNTL\_2 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RXRCV             | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on. Read value is valid only if VDDLDO is on. |
| 30    | RXDP              | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 29    | RXDM              | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 28    | HOSTDISCONNECT    | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 27-26 | LINESTATE         | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 25    | RXVALID           | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 24    | RXVALIDH          | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 23    | RXACTIVE          | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 22    | RXERROR           | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 21    | TXREADY           | R    | 0h    | Read for UTMI signal. Read value is valid only if VDDLDO is on.   |
| 20    | UTMIRESETDONE     | R    | 0h    | Read for UTMIRESETDONE signal   |
| 19    | USEBITSTUFFREG    | R/W  | 0h    | When set to 1 use bits 18-17 from register instead of interface   |
| 18    | TXBITSTUFFENABLE  | R/W  | 0h    | Override value for signal TXBITSTUFFENABLE  |
| 17    | TXBITSTUFFENABLEH | R/W  | 0h    | Override value for pin TXBITSTUFFENABLE   |
| 16    | USETERMCONTROLREG | R/W  | 0h    | -When set to 1 use bits 15-13 from register instead of interface  |
| 15    | TERMSEL           | R/W  | 0h    | Override value for signal TERMSEL   |
| 14    | DPPULLDOWN        | R/W  | 0h    | Override value for signal DPPULLDOWN  |
| 13    | DMPULLDOWN        | R/W  | 0h    | Override value for signal DMPULLDOWN  |
| 12-10 | Reserved          | R/W  | 0h    |   |
| 9     | USEREGSERIALMODE  | R/W  | 0h    | When set to 1 use bits 8-5 from register instead of interface   |
| 8     | TXSE0             | R/W  | 0h    | Override value for signal TXSE0   |

**Table 16-152. UTMI\_INTERFACE\_CNTL\_2 Register Field Descriptions (continued)**

| Bit | Field                        | Type | Reset | Description  |
|-----|------------------------------|------|-------|--|
| 7   | TXDAT                        | R/W  | 0h    | Override value for signal TXDAT  |
| 6   | FSLSSERIALMODE               | R/W  | 0h    | Override value for signal FSLSSERIALMODE   |
| 5   | TXENABLEN                    | R/W  | 0h    | Override value for signal TXENABLEN  |
| 4-1 | Reserved                     | R/W  | 0h    |  |
| 0   | sig_bypass_suspendpulse_incr | R/W  | 0h    | If the suspend signal is asserted for very short-time, it is pulse extended so that all the sampling logic samples it reliably. This pulse extention can be bypassed by writin a '1' to this bit ( so that IP's behaviour is similar to previous versions) |

#### 16.5.4.9 BIST Register (offset = 24h) [reset = 0h]

BIST is shown in [Figure 16-142](#) and described in [Table 16-153](#).

Contains bits related to the built in self test of the phy

**Figure 16-142. BIST Register**

| 31              | 30            | 29               | 28              | 27          | 26                   | 25               | 24                 |
|-----------------|---------------|------------------|-----------------|-------------|----------------------|------------------|--------------------|
| BIST_START      | REDUCED_SWING | BIST_CRC_CALC_EN | BIST_PKT_LENGTH |             |                      |                  |                    |
| R/W-0h          | R/W-0h        | R/W-0h           |                 |             | R/W-0h               |                  |                    |
| 23              | 22            | 21               | 20              | 19          | 18                   | 17               | 16                 |
| BIST_PKT_LENGTH |               |                  |                 | LOOPBACK_EN | BIST_OP_PHASE_SEL    |                  |                    |
| R/W-0h          |               |                  |                 | R/W-0h      | R/W-0h               |                  |                    |
| 15              | 14            | 13               | 12              | 11          | 10                   | 9                | 8                  |
| SWEEP_EN        | SWEEP_MODE    |                  |                 | BIST_PASS   | BIST_BUSY            | Reserved         |                    |
| R/W-0h          | R/W-0h        |                  |                 | R-0h        | R-0h                 | R/W-0h           |                    |
| 7               | 6             | 5                | 4               | 3           | 2                    | 1                | 0                  |
| Reserved        | OP_CODE       |                  | RX_TEST_MODE    | Reserved    | INTER_PKT_DELAY_TEST | HS_ALL_ONES_TEST | USE_BIST_TX_PHASES |
| R/W-0h          | R/W-0h        |                  | R/W-0h          | R/W-0h      | R/W-0h               | R/W-0h           | R/W-0h             |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-153. BIST Register Field Descriptions**

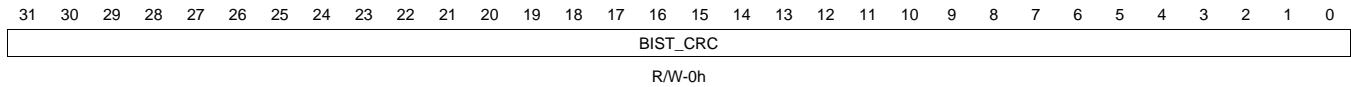
| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31    | BIST_START           | R/W  | 0h    | When set to 1 the BIST mode is started.  |
| 30    | REDUCED_SWING        | R/W  | 0h    | When 1 the TX swing is reduced in BIST mode                                      |
| 29    | BIST_CRC_CALC_EN     | R/W  | 0h    | Enables CRC calculation during BIST when set to 1                                |
| 28-20 | BIST_PKT_LENGTH      | R/W  | 0h    | Address for which BIST to select   |
| 19    | LOOPBACK_EN          | R/W  | 0h    | Enables the loopback mode  |
| 18-16 | BIST_OP_PHASE_SEL    | R/W  | 0h    | Selects which phase to use for data transmission during BIST                     |
| 15    | SWEEP_EN             | R/W  | 0h    | Enables freq sweep on CDR  |
| 14-12 | SWEEP_MODE           | R/W  | 0h    | Selects the freq sweep mode.<br>Details in DFT spec.                             |
| 11    | BIST_PASS            | R    | 0h    | Indicates that the BIST has passed.<br>Read value is valid only if VDDLDO is on. |
| 10    | BIST_BUSY            | R    | 0h    | Indicates that BIST is running.<br>Read value is valid only if VDDLDO is on.     |
| 9-7   | Reserved             | R/W  | 0h    |  |
| 6-5   | OP_CODE              | R/W  | 0h    | Defined in DFT spec  |
| 4     | RX_TEST_MODE         | R/W  | 0h    | Defined in DFT spec  |
| 3     | Reserved             | R/W  | 0h    |  |
| 2     | INTER_PKT_DELAY_TEST | R/W  | 0h    | Defined in DFT spec  |
| 1     | HS_ALL_ONES_TEST     | R/W  | 0h    | Defined in DFT spec  |
| 0     | USE_BIST_TX_PHASES   | R/W  | 0h    | When set to 1 bits 18-16 are activated for choosing the transmitting phase.      |

#### 16.5.4.10 BIST\_CRC Register (offset = 28h) [reset = 0h]

BIST\_CRC is shown in [Figure 16-143](#) and described in [Table 16-154](#).

the CRC code for BIST test

**Figure 16-143. BIST\_CRC Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-154. BIST\_CRC Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description                  |
|------|----------|------|-------|------------------------------|
| 31-0 | BIST_CRC | R/W  | 0h    | The CRC value from the BIST. |

#### 16.5.4.11 CDR\_BIST2 Register (offset = 2Ch) [reset = 0h]

CDR\_BIST2 is shown in [Figure 16-144](#) and described in [Table 16-155](#).

clock data recovery register and BIST register 2

**Figure 16-144. CDR\_BIST2 Register**

| 31               | 30              | 29                 | 28              | 27                     | 26            | 25   | 24               |
|------------------|-----------------|--------------------|-----------------|------------------------|---------------|------|------------------|
| CDR_EXE_EN       | CDR_EXE_MODE    |                    |                 | NUM_DECISIONS          |               |      | CDR_CHOSEN_PHASE |
| R/W-0h           |                 | R/W-0h             |                 | R/W-0h                 |               | R-0h |                  |
| 23               | 22              | 21                 | 20              | 19                     | 18            | 17   | 16               |
| CDR_CHOSEN_PHASE | FORCE_CDR_PHASE |                    |                 | DISABLE_CDR_FREQ_TRACK | CDR_CONFIGURE |      |                  |
| R-0h             |                 | R-0h               |                 | R-0h                   |               | R-0h |                  |
| 15               | 14              | 13                 | 12              | 11                     | 10            | 9    | 8                |
| CDR_CONFIGURE    |                 | FORCE_CDR_PHASE_EN | Bist_start_addr |                        |               |      |                  |
| R-0h             |                 | R-0h               | R/W-0h          |                        |               |      |                  |
| 7                | 6               | 5                  | 4               | 3                      | 2             | 1    | 0                |
| Bist_start_addr  | Bist_end_addr   |                    |                 | R/W-0h                 |               |      |                  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-155. CDR\_BIST2 Register Field Descriptions**

| Bit   | Field                  | Type | Reset | Description  |
|-------|------------------------|------|-------|--|
| 31    | CDR_EXE_EN             | R/W  | 0h    | CDR debug bits                                       |
| 30-28 | CDR_EXE_MODE           | R/W  | 0h    | CDR debug bits                                       |
| 27-25 | NUM_DECISIONS          | R/W  | 0h    | CDR debug bits                                       |
| 24-22 | CDR_CHOSEN_PHASE       | R    | 0h    |  |
| 21-19 | FORCE_CDR_PHASE        | R    | 0h    |  |
| 18    | DISABLE_CDR_FREQ_TRACK | R    | 0h    |  |
| 17-13 | CDR_CONFIGURE          | R    | 0h    |  |
| 12    | FORCE_CDR_PHASE_EN     | R    | 0h    | Use bits 21-19 as the phase to be forced on the CDR. |
| 11-6  | Bist_start_addr        | R/W  | 0h    | See DFT spec for details                             |
| 5-0   | Bist_end_addr          | R/W  | 0h    | See DFT spec for details                             |

#### 16.5.4.12 GPIO Register (offset = 30h) [reset = 0h]

GPIO is shown in [Figure 16-145](#) and described in [Table 16-156](#).

GPIO mode configurations and reads

**Figure 16-145. GPIO Register**

| 31             | 30       | 29         | 28       | 27        | 26        | 25       | 24      |
|----------------|----------|------------|----------|-----------|-----------|----------|---------|
| USEGPIOMODEREG | GPIOMODE | DPGPIOGZ   | DMGPIOGZ | DPGPIOA   | DMGPIOA   | DPGPIOY  | DMGPIOY |
| R/W-0h         | R/W-0h   | R/W-0h     | R/W-0h   | R/W-0h    | R/W-0h    | R-0h     | R-0h    |
| 23             | 22       | 21         | 20       | 19        | 18        | 17       | 16      |
| GPIO1P8VCONFIG |          | GPIOCONFIG |          | DMGPIOPID | DPGPIOPID | Reserved |         |
| R/W-0h         |          | R/W-0h     |          | R/W-0h    | R/W-0h    | R/W-0h   |         |
| 15             | 14       | 13         | 12       | 11        | 10        | 9        | 8       |
| Reserved       |          |            |          |           |           |          |         |
| 7              | 6        | 5          | 4        | 3         | 2         | 1        | 0       |
| R/W-0h         |          |            |          |           |           |          |         |
| Reserved       |          |            |          |           |           |          |         |
| R/W-0h         |          |            |          |           |           |          |         |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-156. GPIO Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description   |
|-------|----------------|------|-------|---|
| 31    | USEGPIOMODEREG | R/W  | 0h    | When set to 1 use bits 31-24 from this register instead of primary inputs     |
| 30    | GPIOMODE       | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 29    | DPGPIOGZ       | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 28    | DMGPIOGZ       | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 27    | DPGPIOA        | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 26    | DMGPIOA        | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 25    | DPGPIOY        | R    | 0h    | The GPIO Y output is stored here  |
| 24    | DMGPIOY        | R    | 0h    | The GPIO Y output is stored here  |
| 23    | GPIO1P8VCONFIG | R/W  | 0h    | Overrides the corresponding primary input                                     |
| 22-20 | GPIOCONFIG     | R/W  | 0h    | Used for configuring the GPIOs.<br>Details to be updated.                     |
| 19    | DMGPIOPID      | R/W  | 0h    | GPIO mode DM pull down enabled.<br>Overrides the corresponding primary input  |
| 18    | DPGPIOPID      | R/W  | 0h    | GPIO mode DP pull-down enabled.<br>Overrides the corresponding primary input. |
| 17-0  | Reserved       | R/W  | 0h    |   |

#### 16.5.4.13 DLLHS Register (offset = 34h) [reset = 8000h]

DLLHS is shown in [Figure 16-146](#) and described in [Table 16-157](#).

bits for control and debug of the DLL inside the phy

**Figure 16-146. DLLHS Register**

| 31                   | 30               | 29              | 28                   | 27         | 26          | 25             | 24     |
|----------------------|------------------|-----------------|----------------------|------------|-------------|----------------|--------|
| Reserved             |                  | DLLHS_LOCK      | DLLHS_GENERATED_CODE |            |             |                |        |
| R/W-0h               |                  | R-0h            |                      | R-0h       |             | R-0h           |        |
| 23                   | 22               | 21              | 20                   | 19         | 18          | 17             | 16     |
| DLLHS_GENERATED_CODE | DLL_SEL_CODE_PHS | DLL_LOCKCHK     |                      |            | DLL_SEL_COD |                |        |
| R-0h                 | R/W-0h           | R/W-0h          |                      |            | R/W-0h      |                |        |
| 15                   | 14               | 13              | 12                   | 11         | 10          | 9              | 8      |
| DLL_PHS0_8           |                  | DLL_FORCED_CODE |                      |            |             | FORCE_DLL_CODE |        |
| R/W-1h               |                  | R/W-0h          |                      |            |             | R/W-0h         |        |
| 7                    | 6                | 5               | 4                    | 3          | 2           | 1              | 0      |
| DLL_RATE             | DLL_FILT         | DLL_CDR_MODE    | DLL_IDLE             | DLL_FREEZE | Reserved    |                |        |
| R/W-0h               | R/W-0h           | R/W-0h          | R/W-0h               | R/W-0h     | R/W-0h      | R/W-0h         | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-157. DLLHS Register Field Descriptions**

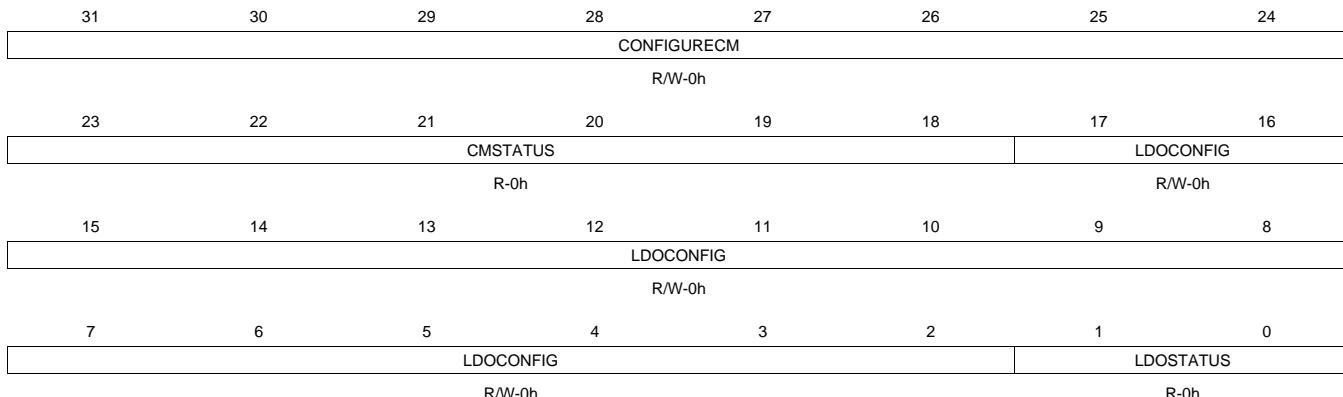
| Bit   | Field                | Type | Reset | Description  |
|-------|----------------------|------|-------|--|
| 31-29 | Reserved             | R/W  | 0h    |  |
| 28    | DLLHS_LOCK           | R    | 0h    | Read the AFE output by this name   |
| 27-22 | DLLHS_GENERATED_CODE | R    | 0h    | Read the AFE output by this name. Read value is valid only if VDDLDO is on.  |
| 21    | DLL_SEL_CODE_PHS     | R/W  | 0h    | Connect to DLLHS_TEST_LDO[0] on AFE interface. see DFT spec for details.     |
| 20-19 | DLL_LOCKCHK          | R/W  | 0h    | Connect to DLLHS_TEST_LDO [2:1] on AFE interface. see DFT spec for details.  |
| 18-16 | DLL_SEL_COD          | R/W  | 0h    | Connect to DLLHS_TEST_LDO [5:3] on AFE interface. see DFT spec for details.  |
| 15    | DLL_PHS0_8           | R/W  | 1h    | Connect to DLLHS_TEST_LDO[6] on AFE interface. see DFT spec for details.     |
| 14-9  | DLL_FORCED_CODE      | R/W  | 0h    | Connect to the pin of this name on AFE interface. see DFT spec for details.  |
| 8     | FORCE_DLL_CODE       | R/W  | 0h    | Connect to DLLHS_TEST_LDO[11] on AFE interface. see DFT spec for details.    |
| 7-6   | DLL_RATE             | R/W  | 0h    | Connect to DLLHS_TEST_LDO [8:7] on AFE interface. see DFT spec for details.  |
| 5-4   | DLL_FILT             | R/W  | 0h    | Connect to DLLHS_TEST_LDO [10:9] on AFE interface. see DFT spec for details. |
| 3     | DLL_CDR_MODE         | R/W  | 0h    | Connect to the pin of this name on AFE interface. see DFT spec for details.  |
| 2     | DLL_IDLE             | R/W  | 0h    | Connect to DLLHS_TEST_LDO[12] on AFE interface. see DFT spec for details.    |
| 1     | DLL_FREEZE           | R/W  | 0h    | Connect to DLLHS_TEST_LDO[13] on AFE interface. see DFT spec for details.    |
| 0     | Reserved             | R/W  | 0h    |  |

#### 16.5.4.14 USB2PHYCM\_CONFIG Register (offset = 3Ch) [reset = 0h]

USB2PHYCM\_CONFIG is shown in [Figure 16-147](#) and described in [Table 16-158](#).

Config and status register for the USB2PHYCM and LDO

**Figure 16-147. USB2PHYCM\_CONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-158. USB2PHYCM\_CONFIG Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-24 | CONFIGURECM | R/W  | 0h    | Connects to the CONFIGURECM pins.<br>see DFT spec for details. |
| 23-18 | CMSTATUS    | R    | 0h    | Reads the CMSTATUS bits.<br>see DFT spec for details.          |
| 17-2  | LDOCONFIG   | R/W  | 0h    | The LDOCONFIG bit settings.<br>See DFT spec for details.       |
| 1-0   | LDOSTATUS   | R    | 0h    | Reads the LDOSTATUS bits.<br>see DFT spec for details.         |

#### 16.5.4.15 AD\_INTERFACE\_REG1 Register (offset = 44h) [reset = 0h]

AD\_INTERFACE\_REG1 is shown in [Figure 16-148](#) and described in [Table 16-159](#).

All bits (unless defined) are bypass bits for internal analog to digital interface pins with the same name. All the bits of this register, except the over-ride bits return a '0' on read, if VDDLDO is off.

**Figure 16-148. AD\_INTERFACE\_REG1 Register**

| 31                     | 30           | 29          | 28                 | 27                  | 26           | 25           | 24            |
|------------------------|--------------|-------------|--------------------|---------------------|--------------|--------------|---------------|
| USE_AD_DATA_REG        | HS_TX_DATA   | FS_TX_DATA  | TEST_PRE_EN_CNT_RL | SQ_PRE_EN           | HS_TX_PRE_EN | HS_RX_PRE_EN | TEST_EN_CNTRL |
| R/W-0h                 | R/W-0h       | R/W-0h      | R/W-0h             | R/W-0h              | R/W-0h       | R/W-0h       | R/W-0h        |
| 23                     | 22           | 21          | 20                 | 19                  | 18           | 17           | 16            |
| HS_TX_EN               | FS_RX_EN     | Reserved    | SQ_EN              | HS_RX_EN            | TEST_HS_MODE | HS_HV_SW     | HS_CHIRP      |
| R/W-0h                 | R/W-0h       | R/W-0h      | R/W-0h             | R/W-0h              | R/W-0h       | R/W-0h       | R/W-0h        |
| 15                     | 14           | 13          | 12                 | 11                  | 10           | 9            | 8             |
| TEST_FS_MODE           | FSTX_GZ      | FSTX_PRE_EN | Reserved           | TEST_SQ_CAL_CONTROL | SQ_CAL_EN3   | SQ_CAL_EN1   | SQ_CAL_EN2    |
| R/W-0h                 | R/W-0h       | R/W-0h      | R/W-0h             | R/W-0h              | R/W-0h       | R/W-0h       | R/W-0h        |
| 7                      | 6            | 5           | 4                  | 3                   | 2            | 1            | 0             |
| TEST_RTERM_CAL_CONTROL | RTERM_CAL_EN | DLL_RX_DATA | DISCON_DETECT      | USE_LSHOST_REG      | LHOSTMODE    | LSFS_RX_DATA | SQUELCH       |
| R/W-0h                 | R/W-0h       | R-0h        | R-0h               | R/W-0h              | R/W-0h       | R-0h         | R-0h          |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-159. AD\_INTERFACE\_REG1 Register Field Descriptions**

| Bit | Field               | Type | Reset | Description             |
|-----|---------------------|------|-------|-------------------------|
| 31  | USE_AD_DATA_REG     | R/W  | 0h    | Override for bits 30-29 |
| 30  | HS_TX_DATA          | R/W  | 0h    |                         |
| 29  | FS_TX_DATA          | R/W  | 0h    |                         |
| 28  | TEST_PRE_EN_CNTRL   | R/W  | 0h    | Override for bits 27-25 |
| 27  | SQ_PRE_EN           | R/W  | 0h    |                         |
| 26  | HS_TX_PRE_EN        | R/W  | 0h    |                         |
| 25  | HS_RX_PRE_EN        | R/W  | 0h    |                         |
| 24  | TEST_EN_CNTRL       | R/W  | 0h    | Override for bits 23-19 |
| 23  | HS_TX_EN            | R/W  | 0h    |                         |
| 22  | FS_RX_EN            | R/W  | 0h    |                         |
| 21  | Reserved            | R/W  | 0h    |                         |
| 20  | SQ_EN               | R/W  | 0h    |                         |
| 19  | HS_RX_EN            | R/W  | 0h    |                         |
| 18  | TEST_HS_MODE        | R/W  | 0h    | Override for bits 17-16 |
| 17  | HS_HV_SW            | R/W  | 0h    |                         |
| 16  | HS_CHIRP            | R/W  | 0h    |                         |
| 15  | TEST_FS_MODE        | R/W  | 0h    | Override for bits 14-12 |
| 14  | FSTX_GZ             | R/W  | 0h    |                         |
| 13  | FSTX_PRE_EN         | R/W  | 0h    |                         |
| 12  | Reserved            | R/W  | 0h    |                         |
| 11  | TEST_SQ_CAL CONTROL | R/W  | 0h    | Override for bits 10-8  |
| 10  | SQ_CAL_EN3          | R/W  | 0h    |                         |

**Table 16-159. AD\_INTERFACE\_REG1 Register Field Descriptions (continued)**

| Bit | Field                   | Type | Reset | Description            |
|-----|-------------------------|------|-------|------------------------|
| 9   | SQ_CAL_EN1              | R/W  | 0h    |                        |
| 8   | SQ_CAL_EN2              | R/W  | 0h    |                        |
| 7   | TEST_RTERM_CAL_CO NTROL | R/W  | 0h    | Override for bits 6    |
| 6   | RTERM_CAL_EN            | R/W  | 0h    |                        |
| 5   | DLL_RX_DATA             | R    | 0h    |                        |
| 4   | DISCON_DETECT           | R    | 0h    |                        |
| 3   | USE_LSHOST_REG          | R/W  | 0h    | Use bit 2 for this reg |
| 2   | LSHOSTMODE              | R/W  | 0h    |                        |
| 1   | LSFS_RX_DATA            | R    | 0h    |                        |
| 0   | SQUELCH                 | R    | 0h    |                        |

#### 16.5.4.16 AD\_INTERFACE\_REG2 Register (offset = 48h) [reset = 0h]

AD\_INTERFACE\_REG2 is shown in [Figure 16-149](#) and described in [Table 16-160](#).

All bits (unless defined) are bypass bits for internal analog to digital interface pins with the same name. All the bits of this register, except the over-ride bits return a '0' on read, if VDDLDO is off.

**Figure 16-149. AD\_INTERFACE\_REG2 Register**

| 31                  | 30              | 29              | 28                 | 27                 | 26                 | 25                 | 24                  |
|---------------------|-----------------|-----------------|--------------------|--------------------|--------------------|--------------------|---------------------|
| USE_SUSP_DRV_REG    | SUS_DRV_DP_DATA | SUS_DRV_DP_EN   | SUS_DRV_DM_DATA    | SUS_DRV_DM_EN      | USE_DISCON_REG     | DISCON_EN          | DISCON_PRE_EN       |
| R/W-0h              | R/W-0h          | R/W-0h          | R/W-0h             | R/W-0h             | R/W-0h             | R/W-0h             | R/W-0h              |
| 23                  | 22              | 21              | 20                 | 19                 | 18                 | 17                 | 16                  |
| Reserved            |                 | SPARE_OUT_CORE  |                    |                    |                    | SERX_DP_CORE       | SERX_DM_CORE        |
| R/W-0h              |                 | R-0h            |                    |                    |                    | R-0h               | R-0h                |
| 15                  | 14              | 13              | 12                 | 11                 | 10                 | 9                  | 8                   |
| USE_HSRX_CAL_EN_REG | HSRX_CAL_EN     | USE_RPU_RPD_REG | RPU_DP_SW1_EN_CORE | RPU_DP_SW2_EN_CORE | RPU_DM_SW1_EN_CORE | RPU_DM_SW2_EN_CORE | DP_PULLDOWN_EN_CORE |
| R/W-0h              | R/W-0h          | R/W-0h          | R/W-0h             | R/W-0h             | R/W-0h             | R/W-0h             | R/W-0h              |
| 7                   | 6               | 5               | 4                  | 3                  | 2                  | 1                  | 0                   |
| DM_PULLDOWN_EN_CORE | DP_DM_5V_SHORT  |                 | SPARE_IN_CORE      |                    |                    |                    | PORZ                |
| R/W-0h              | R-0h            |                 | R/W-0h             |                    |                    |                    | R-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-160. AD\_INTERFACE\_REG2 Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description                                    |
|-------|---------------------|------|-------|--|
| 31    | USE_SUSP_DRV_REG    | R/W  | 0h    | Use bits 27-30 from this register as overrides |
| 30    | SUS_DRV_DP_DATA     | R/W  | 0h    |  |
| 29    | SUS_DRV_DP_EN       | R/W  | 0h    |  |
| 28    | SUS_DRV_DM_DATA     | R/W  | 0h    |  |
| 27    | SUS_DRV_DM_EN       | R/W  | 0h    |  |
| 26    | USE_DISCON_REG      | R/W  | 0h    | Use bits 24-25 from this register as override  |
| 25    | DISCON_EN           | R/W  | 0h    |  |
| 24    | DISCON_PRE_EN       | R/W  | 0h    |  |
| 23    | Reserved            | R/W  | 0h    |  |
| 22-18 | SPARE_OUT_CORE      | R    | 0h    |  |
| 17    | SERX_DP_CORE        | R    | 0h    |  |
| 16    | SERX_DM_CORE        | R    | 0h    |  |
| 15    | USE_HSRX_CAL_EN_REG | R/W  | 0h    | Use bit 14 from this register as override      |
| 14    | HSRX_CAL_EN         | R/W  | 0h    |  |
| 13    | USE_RPU_RPD_REG     | R/W  | 0h    | Use override from bits 7-12                    |
| 12    | RPU_DP_SW1_EN_CORE  | R/W  | 0h    |  |
| 11    | RPU_DP_SW2_EN_CORE  | R/W  | 0h    |  |
| 10    | RPU_DM_SW1_EN_CORE  | R/W  | 0h    |  |
| 9     | RPU_DM_SW2_EN_CORE  | R/W  | 0h    |  |
| 8     | DP_PULLDOWN_EN_CORE | R/W  | 0h    |  |

**Table 16-160. AD\_INTERFACE\_REG2 Register Field Descriptions (continued)**

| Bit | Field               | Type | Reset | Description   |
|-----|---------------------|------|-------|---|
| 7   | DM_PULLDOWN_EN_CORE | R/W  | 0h    |   |
| 6   | DP_DM_5V_SHORT      | R    | 0h    |   |
| 5-1 | SPARE_IN_CORE       | R/W  | 0h    |   |
| 0   | PORZ                | R    | 0h    | Read only bit -andgt;<br>the PORZ generated from the digital registered on the A-D interface. |

#### 16.5.4.17 AD\_INTERFACE\_REG3 Register (offset = 4Ch) [reset = 0h]

AD\_INTERFACE\_REG3 is shown in [Figure 16-150](#) and described in [Table 16-161](#).

All bits (unless defined) are bypass bits for internal analog to digital interface pins with the same name. All the bits of this register, except the over-ride bits return a '0' on read, if VDDLDO is off.

**Figure 16-150. AD\_INTERFACE\_REG3 Register**

| 31                | 30          | 29            | 28       | 27                  | 26          | 25              | 24           |
|-------------------|-------------|---------------|----------|---------------------|-------------|-----------------|--------------|
| USE_HSOS_DATA_REG |             |               |          | HSOS_DATA           |             |                 |              |
| R/W-0h            |             |               |          | R/W-0h              |             |                 |              |
| 23                | 22          | 21            | 20       | 19                  | 18          | 17              | 16           |
| HSOS_DATA         | USE_FS_REG3 | FSTX_MODE     | FSTX_SE0 | USE_HS_TERM_RES_REG | HS_TERM_RES |                 | SPARE_IN_LDO |
| R/W-0h            | R/W-0h      | R/W-0h        | R/W-0h   | R/W-0h              | R/W-0h      |                 | R/W-0h       |
| 15                | 14          | 13            | 12       | 11                  | 10          | 9               | 8            |
|                   |             | SPARE_IN_LDO  |          |                     |             | SPARE_OUT_LDO   |              |
|                   |             | R/W-0h        |          |                     |             | R-0h            |              |
| 7                 | 6           | 5             | 4        | 3                   | 2           | 1               | 0            |
|                   |             | SPARE_OUT_LDO |          |                     |             | USE_FARCORE_REG | FARCORE      |
|                   |             |               |          |                     |             | R/W-0h          | R/W-0h       |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-161. AD\_INTERFACE\_REG3 Register Field Descriptions**

| Bit   | Field               | Type | Reset | Description                                    |
|-------|---------------------|------|-------|--|
| 31    | USE_HSOS_DATA_REG   | R/W  | 0h    | Use bits 30-23 in this register as bypass bits |
| 30-23 | HSOS_DATA           | R/W  | 0h    |  |
| 22    | USE_FS_REG3         | R/W  | 0h    | Use bits 20-21 as bypass bits                  |
| 21    | FSTX_MODE           | R/W  | 0h    |  |
| 20    | FSTX_SE0            | R/W  | 0h    |  |
| 19    | USE_HS_TERM_RES_REG | R/W  | 0h    | Use bit 18 as override bit                     |
| 18    | HS_TERM_RES         | R/W  | 0h    |  |
| 17-10 | SPARE_IN_LDO        | R/W  | 0h    |  |
| 9-2   | SPARE_OUT_LDO       | R    | 0h    |  |
| 1     | USE_FARCORE_REG     | R/W  | 0h    | Use bit 0 from this register as bypass         |
| 0     | FARCORE             | R/W  | 0h    |  |

#### 16.5.4.18 ANA\_CONFIG2 Register (offset = 54h) [reset = 0h]

ANA\_CONFIG2 is shown in [Figure 16-151](#) and described in [Table 16-162](#).

Used to configure and debug the analog blocks.

**Figure 16-151. ANA\_CONFIG2 Register**

|              |    |          |    |          |    |              |    |
|--------------|----|----------|----|----------|----|--------------|----|
| 31           | 30 | 29       | 28 | 27       | 26 | 25           | 24 |
| Reserved     |    |          |    |          |    | REF_GEN_TEST |    |
| R-0h         |    |          |    |          |    | R/W-0h       |    |
| 23           | 22 | 21       | 20 | 19       | 18 | 17           | 16 |
| REF_GEN_TEST |    |          |    | Reserved |    | RTERM_TEST   |    |
| R/W-0h       |    |          |    | R/W-0h   |    | R/W-0h       |    |
| 15           | 14 | 13       | 12 | 11       | 10 | 9            | 8  |
| RTERM_TEST   |    | Reserved |    |          |    |              |    |
| R/W-0h       |    | R/W-0h   |    |          |    |              |    |
| 7            | 6  | 5        | 4  | 3        | 2  | 1            | 0  |
| Reserved     |    |          |    |          |    |              |    |
| R/W-0h       |    |          |    |          |    |              |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-162. ANA\_CONFIG2 Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 31-27 | Reserved     | R    | 0h    | Reserved.   |
| 26-20 | REF_GEN_TEST | R/W  | 0h    | 0000000b - Produces the typical vertical eye diagram amplitude (default).<br>1100000b - Increases vertical eye diagram amplitude by 15 mV.<br>1010000b - Decreases vertical eye diagram amplitude by 15 mV.<br>All other values are reserved. |
| 19-18 | Reserved     | R/W  | 0h    | Reserved.   |
| 17-15 | RTERM_TEST   | R/W  | 0h    | 000b - Typical termination impedance (default).<br>011b - Decreases the termination impedance by 2 to 3% and can be used to increase the vertical eye diagram amplitude by 1 to 1.5%.<br>All other values reserved.                           |
| 14-0  | Reserved     | R/W  | 0h    | Reserved.   |

#### 16.5.5 CPPI\_DMA Registers

[Table 16-163](#) lists the memory-mapped registers for the CPPI\_DMA. All register offset addresses not listed in [Table 16-163](#) should be considered as reserved locations and the register contents should not be modified.

**Table 16-163. CPPI\_DMA REGISTERS**

| Offset | Acronym  | Register Name | Section                           |
|--------|----------|---------------|-----------------------------------|
| 0h     | DMAREVID |               | <a href="#">Section 16.5.5.1</a>  |
| 4h     | TDFDQ    |               | <a href="#">Section 16.5.5.2</a>  |
| 8h     | DMAEMU   |               | <a href="#">Section 16.5.5.3</a>  |
| 800h   | TXGCR0   |               | <a href="#">Section 16.5.5.4</a>  |
| 808h   | RXGCR0   |               | <a href="#">Section 16.5.5.5</a>  |
| 80Ch   | RXHPCRA0 |               | <a href="#">Section 16.5.5.6</a>  |
| 810h   | RXHPCR0B |               | <a href="#">Section 16.5.5.7</a>  |
| 820h   | TXGCR1   |               | <a href="#">Section 16.5.5.8</a>  |
| 828h   | RXGCR1   |               | <a href="#">Section 16.5.5.9</a>  |
| 82Ch   | RXHPCRA1 |               | <a href="#">Section 16.5.5.10</a> |

**Table 16-163. CPPI\_DMA REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b> | <b>Section</b>    |
|---------------|----------------|----------------------|-------------------|
| 830h          | RXHPCRB1       |                      | Section 16.5.5.11 |
| 840h          | TXGCR2         |                      | Section 16.5.5.12 |
| 848h          | RXGCR2         |                      | Section 16.5.5.13 |
| 84Ch          | RXHPCRA2       |                      | Section 16.5.5.14 |
| 850h          | RXHPCRB2       |                      | Section 16.5.5.15 |
| 860h          | TXGCR3         |                      | Section 16.5.5.16 |
| 868h          | RXGCR3         |                      | Section 16.5.5.17 |
| 86Ch          | RXHPCRA3       |                      | Section 16.5.5.18 |
| 870h          | RXHPCRB3       |                      | Section 16.5.5.19 |
| 880h          | TXGCR4         |                      | Section 16.5.5.20 |
| 888h          | RXGCR4         |                      | Section 16.5.5.21 |
| 88Ch          | RXHPCRA4       |                      | Section 16.5.5.22 |
| 890h          | RXHPCRB4       |                      | Section 16.5.5.23 |
| 8A0h          | TXGCR5         |                      | Section 16.5.5.24 |
| 8A8h          | RXGCR5         |                      | Section 16.5.5.25 |
| 8ACh          | RXHPCRA5       |                      | Section 16.5.5.26 |
| 8B0h          | RXHPCRB5       |                      | Section 16.5.5.27 |
| 8C0h          | TXGCR6         |                      | Section 16.5.5.28 |
| 8C8h          | RXGCR6         |                      | Section 16.5.5.29 |
| 8CCh          | RXHPCRA6       |                      | Section 16.5.5.30 |
| 8D0h          | RXHPCRB6       |                      | Section 16.5.5.31 |
| 8E0h          | TXGCR7         |                      | Section 16.5.5.32 |
| 8E8h          | RXGCR7         |                      | Section 16.5.5.33 |
| 8ECh          | RXHPCRA7       |                      | Section 16.5.5.34 |
| 8F0h          | RXHPCRB7       |                      | Section 16.5.5.35 |
| 900h          | TXGCR8         |                      | Section 16.5.5.36 |
| 908h          | RXGCR8         |                      | Section 16.5.5.37 |
| 90Ch          | RXHPCRA8       |                      | Section 16.5.5.38 |
| 910h          | RXHPCRB8       |                      | Section 16.5.5.39 |
| 920h          | TXGCR9         |                      | Section 16.5.5.40 |
| 928h          | RXGCR9         |                      | Section 16.5.5.41 |
| 92Ch          | RXHPCRA9       |                      | Section 16.5.5.42 |
| 930h          | RXHPCRB9       |                      | Section 16.5.5.43 |
| 940h          | TXGCR10        |                      | Section 16.5.5.44 |
| 948h          | RXGCR10        |                      | Section 16.5.5.45 |
| 94Ch          | RXHPCRA10      |                      | Section 16.5.5.46 |
| 950h          | RXHPCRB10      |                      | Section 16.5.5.47 |
| 960h          | TXGCR11        |                      | Section 16.5.5.48 |
| 968h          | RXGCR11        |                      | Section 16.5.5.49 |
| 96Ch          | RXHPCRA11      |                      | Section 16.5.5.50 |
| 970h          | RXHPCRB11      |                      | Section 16.5.5.51 |
| 980h          | TXGCR12        |                      | Section 16.5.5.52 |
| 988h          | RXGCR12        |                      | Section 16.5.5.53 |
| 98Ch          | RXHPCRA12      |                      | Section 16.5.5.54 |
| 990h          | RXHPCRB12      |                      | Section 16.5.5.55 |
| 9A0h          | TXGCR13        |                      | Section 16.5.5.56 |
| 9A8h          | RXGCR13        |                      | Section 16.5.5.57 |

**Table 16-163. CPPI\_DMA REGISTERS (continued)**

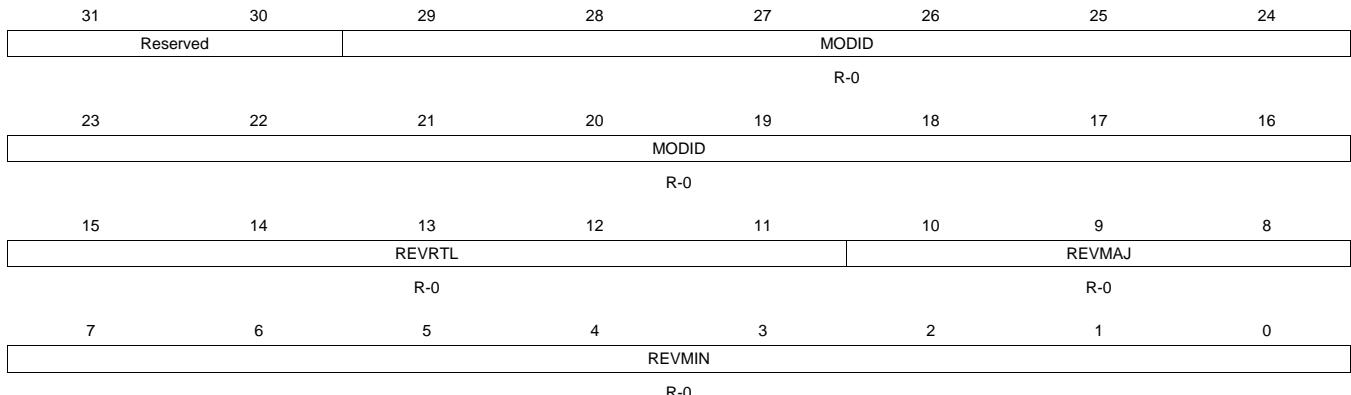
| Offset | Acronym   | Register Name | Section            |
|--------|-----------|---------------|--------------------|
| 9ACh   | RXHPCRA13 |               | Section 16.5.5.58  |
| 9B0h   | RXHPCRB13 |               | Section 16.5.5.59  |
| 9C0h   | TXGCR14   |               | Section 16.5.5.60  |
| 9C8h   | RXGCR14   |               | Section 16.5.5.61  |
| 9CCh   | RXHPCRA14 |               | Section 16.5.5.62  |
| 9D0h   | RXHPCRB14 |               | Section 16.5.5.63  |
| 9E0h   | TXGCR15   |               | Section 16.5.5.64  |
| 9E8h   | RXGCR15   |               | Section 16.5.5.65  |
| 9ECh   | RXHPCRA15 |               | Section 16.5.5.66  |
| 9F0h   | RXHPCRB15 |               | Section 16.5.5.67  |
| A00h   | TXGCR16   |               | Section 16.5.5.68  |
| A08h   | RXGCR16   |               | Section 16.5.5.69  |
| A0Ch   | RXHPCRA16 |               | Section 16.5.5.70  |
| A10h   | RXHPCRB16 |               | Section 16.5.5.71  |
| A20h   | TXGCR17   |               | Section 16.5.5.72  |
| A28h   | RXGCR17   |               | Section 16.5.5.73  |
| A2Ch   | RXHPCRA17 |               | Section 16.5.5.74  |
| A30h   | RXHPCRB17 |               | Section 16.5.5.75  |
| A40h   | TXGCR18   |               | Section 16.5.5.76  |
| A48h   | RXGCR18   |               | Section 16.5.5.77  |
| A4Ch   | RXHPCRA18 |               | Section 16.5.5.78  |
| A50h   | RXHPCRB18 |               | Section 16.5.5.79  |
| A60h   | TXGCR19   |               | Section 16.5.5.80  |
| A68h   | RXGCR19   |               | Section 16.5.5.81  |
| A6Ch   | RXHPCRA19 |               | Section 16.5.5.82  |
| A70h   | RXHPCRB19 |               | Section 16.5.5.83  |
| A80h   | TXGCR20   |               | Section 16.5.5.84  |
| A88h   | RXGCR20   |               | Section 16.5.5.85  |
| A8Ch   | RXHPCRA20 |               | Section 16.5.5.86  |
| A90h   | RXHPCRB20 |               | Section 16.5.5.87  |
| AA0h   | TXGCR21   |               | Section 16.5.5.88  |
| AA8h   | RXGCR21   |               | Section 16.5.5.89  |
| AACh   | RXHPCRA21 |               | Section 16.5.5.90  |
| AB0h   | RXHPCRB21 |               | Section 16.5.5.91  |
| AC0h   | TXGCR22   |               | Section 16.5.5.92  |
| AC8h   | RXGCR22   |               | Section 16.5.5.93  |
| ACCh   | RXHPCRA22 |               | Section 16.5.5.94  |
| AD0h   | RXHPCRB22 |               | Section 16.5.5.95  |
| AE0h   | TXGCR23   |               | Section 16.5.5.96  |
| AE8h   | RXGCR23   |               | Section 16.5.5.97  |
| AECh   | RXHPCRA23 |               | Section 16.5.5.98  |
| AF0h   | RXHPCRB23 |               | Section 16.5.5.99  |
| B00h   | TXGCR24   |               | Section 16.5.5.100 |
| B08h   | RXGCR24   |               | Section 16.5.5.101 |
| B0Ch   | RXHPCRA24 |               | Section 16.5.5.102 |
| B10h   | RXHPCRB24 |               | Section 16.5.5.103 |
| B20h   | TXGCR25   |               | Section 16.5.5.104 |

**Table 16-163. CPPI\_DMA REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b> | <b>Section</b>     |
|---------------|----------------|----------------------|--------------------|
| B28h          | RXGCR25        |                      | Section 16.5.5.105 |
| B2Ch          | RXHPCRA25      |                      | Section 16.5.5.106 |
| B30h          | RXHPCRB25      |                      | Section 16.5.5.107 |
| B40h          | TXGCR26        |                      | Section 16.5.5.108 |
| B48h          | RXGCR26        |                      | Section 16.5.5.109 |
| B4Ch          | RXHPCRA26      |                      | Section 16.5.5.110 |
| B50h          | RXHPCRB26      |                      | Section 16.5.5.111 |
| B60h          | TXGCR27        |                      | Section 16.5.5.112 |
| B68h          | RXGCR27        |                      | Section 16.5.5.113 |
| B6Ch          | RXHPCRA27      |                      | Section 16.5.5.114 |
| B70h          | RXHPCRB27      |                      | Section 16.5.5.115 |
| B80h          | TXGCR28        |                      | Section 16.5.5.116 |
| B88h          | RXGCR28        |                      | Section 16.5.5.117 |
| B8Ch          | RXHPCRA28      |                      | Section 16.5.5.118 |
| B90h          | RXHPCRB28      |                      | Section 16.5.5.119 |
| BA0h          | TXGCR29        |                      | Section 16.5.5.120 |
| BA8h          | RXGCR29        |                      | Section 16.5.5.121 |
| BACh          | RXHPCRA29      |                      | Section 16.5.5.122 |
| BB0h          | RXHPCRB29      |                      | Section 16.5.5.123 |

**16.5.5.1 DMAREVID Register (offset = 0h) [reset = 530901h]**

DMAREVID is shown in [Figure 16-152](#) and described in [Table 16-164](#).

**Figure 16-152. DMAREVID Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-164. DMAREVID Register Field Descriptions**

| Bit   | Field  | Type | Reset | Description                                      |
|-------|--------|------|-------|--|
| 29-16 | MODID  | R-0  | 0     | Module ID field                                  |
| 15-11 | REVRTL | R-0  | 0     | RTL revision.<br>Will vary depending on release. |
| 10-8  | REVMAJ | R-0  | 0     | Major revision.                                  |
| 7-0   | REVMIN | R-0  | 0     | Minor revision.<br>CPPI DMA Revision Register    |

### 16.5.5.2 TDFDQ Register (offset = 4h) [reset = 0h]

TDFDQ is shown in [Figure 16-153](#) and described in [Table 16-165](#).

**Figure 16-153. TDFDQ Register**

|          |    |              |    |              |        |    |    |
|----------|----|--------------|----|--------------|--------|----|----|
| 31       | 30 | 29           | 28 | 27           | 26     | 25 | 24 |
| Reserved |    |              |    |              |        |    |    |
| 23       | 22 | 21           | 20 | 19           | 18     | 17 | 16 |
| Reserved |    |              |    |              |        |    |    |
| 15       | 14 | 13           | 12 | 11           | 10     | 9  | 8  |
| Reserved |    | TD_DESC_QMGR |    | TD_DESC_QNUM |        |    |    |
|          |    | R/W-0h       |    |              | R/W-0h |    |    |
| 7        | 6  | 5            | 4  | 3            | 2      | 1  | 0  |
|          |    | TD_DESC_QNUM |    |              |        |    |    |
|          |    | R/W-0h       |    |              | R/W-0h |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-165. TDFDQ Register Field Descriptions**

| Bit   | Field        | Type | Reset | Description   |
|-------|--------------|------|-------|---|
| 13-12 | TD_DESC_QMGR | R/W  | 0h    | This field controls which of the 4 Queue Managers the DMA will access in order to allocate a channel teardown descriptor from the teardown descriptor queue.  |
| 11-0  | TD_DESC_QNUM | R/W  | 0h    | This field controls which of the 2K queues in the indicated queue manager should be read in order to allocate channel teardown descriptors.<br>CPPI DMA Teardown Free Descriptor Queue Control Register |

### 16.5.5.3 DMAEMU Register (offset = 8h) [reset = 0h]

DMAEMU is shown in [Figure 16-154](#) and described in [Table 16-166](#).

**Figure 16-154. DMAEMU Register**

|          |    |    |    |    |    |        |        |
|----------|----|----|----|----|----|--------|--------|
| 31       | 30 | 29 | 28 | 27 | 26 | 25     | 24     |
| Reserved |    |    |    |    |    |        |        |
| 23       | 22 | 21 | 20 | 19 | 18 | 17     | 16     |
| Reserved |    |    |    |    |    |        |        |
| 15       | 14 | 13 | 12 | 11 | 10 | 9      | 8      |
| Reserved |    |    |    |    |    |        |        |
| 7        | 6  | 5  | 4  | 3  | 2  | 1      | 0      |
| Reserved |    |    |    |    |    | SOFT   | FREE   |
|          |    |    |    |    |    | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

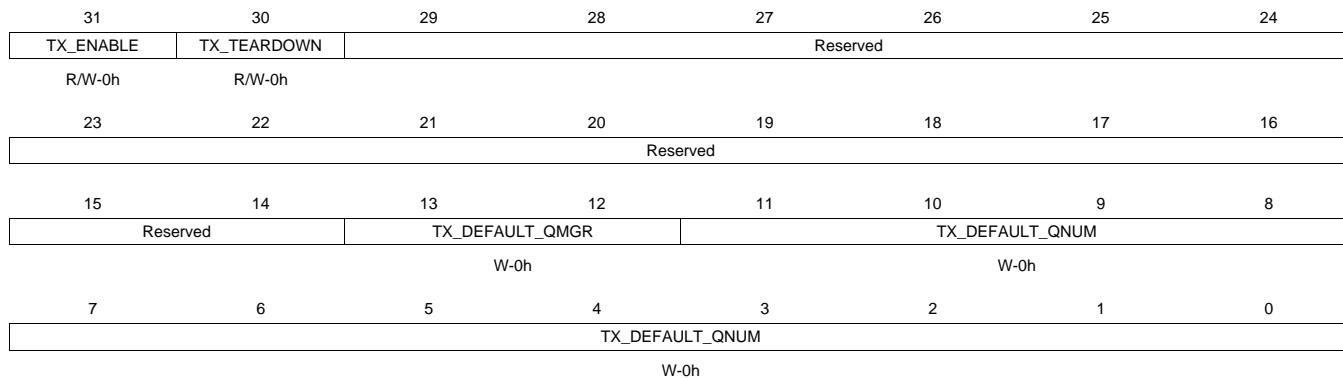
**Table 16-166. DMAEMU Register Field Descriptions**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 1   | SOFT  | R/W  | 0h    | Control for emulation pause request<br>1 = Does not force emu_pause_req low<br>0 = Forces emu_pause_req low |
| 0   | FREE  | R/W  | 0h    | Enable for emulation suspend CPPI DMA Emulation Control Register  |

#### 16.5.5.4 TXGCR0 Register (offset = 800h) [reset = 0h]

TXGCR0 is shown in [Figure 16-155](#) and described in [Table 16-167](#).

**Figure 16-155. TXGCR0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-167. TXGCR0 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.5 RXGCR0 Register (offset = 808h) [reset = 0h]

RXGCR0 is shown in [Figure 16-156](#) and described in [Table 16-168](#).

**Figure 16-156. RXGCR0 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-168. RXGCR0 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-168. RXGCR0 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.6 RXHPCRA0 Register (offset = 80Ch) [reset = 0h]**

 RXHPCRA0 is shown in [Figure 16-157](#) and described in [Table 16-169](#).

**Figure 16-157. RXHPCRA0 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-169. RXHPCRA0 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.7 RXHPCRB0 Register (offset = 810h) [reset = 0h]

RXHPCRB0 is shown in [Figure 16-158](#) and described in [Table 16-170](#).

**Figure 16-158. RXHPCRB0 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

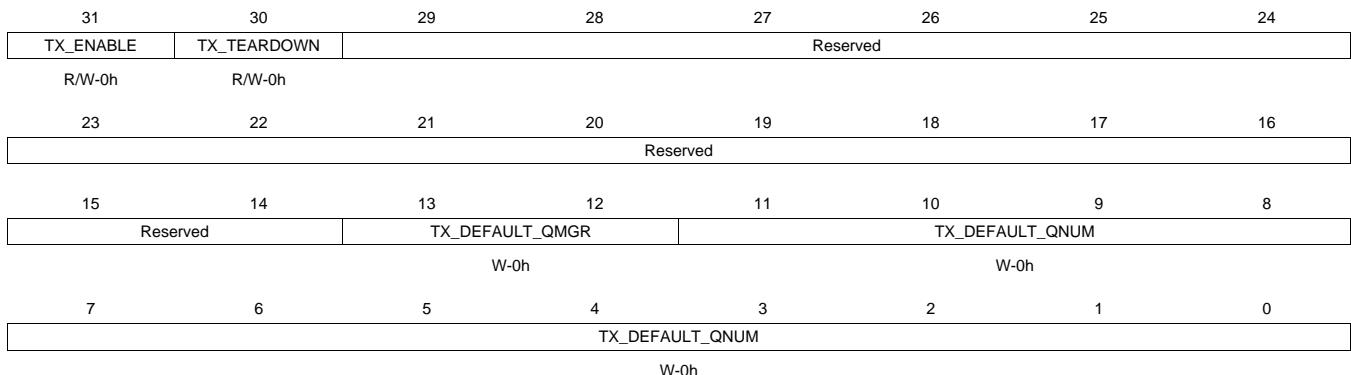
**Table 16-170. RXHPCRB0 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.8 TXGCR1 Register (offset = 820h) [reset = 0h]

TXGCR1 is shown in [Figure 16-159](#) and described in [Table 16-171](#).

**Figure 16-159. TXGCR1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-171. TXGCR1 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.9 RXGCR1 Register (offset = 828h) [reset = 0h]

RXGCR1 is shown in [Figure 16-160](#) and described in [Table 16-172](#).

**Figure 16-160. RXGCR1 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-172. RXGCR1 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-172. RXGCR1 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.10 RXHPCRA1 Register (offset = 82Ch) [reset = 0h]

RXHPCRA1 is shown in [Figure 16-161](#) and described in [Table 16-173](#).

**Figure 16-161. RXHPCRA1 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-173. RXHPCRA1 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.11 RXHPCRB1 Register (offset = 830h) [reset = 0h]**

 RXHPCRB1 is shown in [Figure 16-162](#) and described in [Table 16-174](#).

**Figure 16-162. RXHPCRB1 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ3_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ2_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

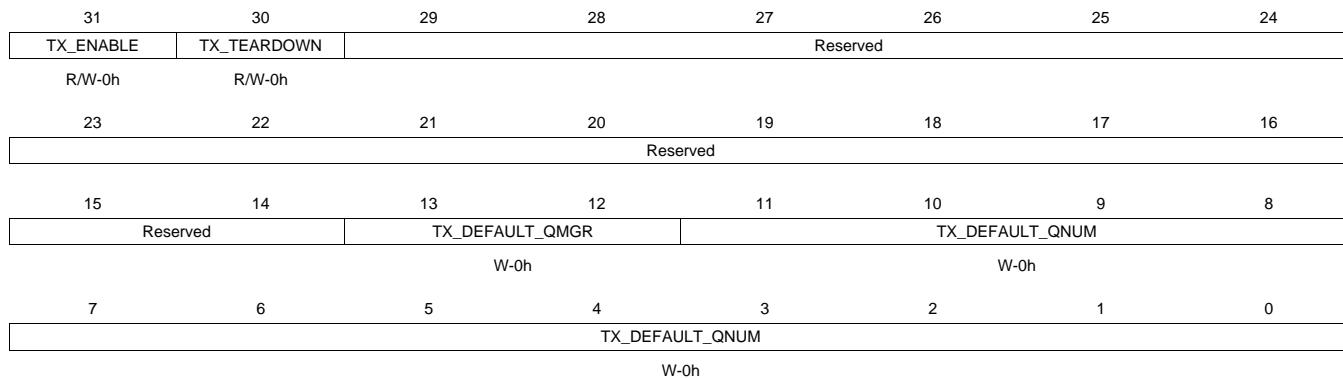
**Table 16-174. RXHPCRB1 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.12 TXGCR2 Register (offset = 840h) [reset = 0h]

TXGCR2 is shown in [Figure 16-163](#) and described in [Table 16-175](#).

**Figure 16-163. TXGCR2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-175. TXGCR2 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.13 RXGCR2 Register (offset = 848h) [reset = 0h]**

RXGCR2 is shown in [Figure 16-164](#) and described in [Table 16-176](#).

**Figure 16-164. RXGCR2 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-176. RXGCR2 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-176. RXGCR2 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.14 RXHPCRA2 Register (offset = 84Ch) [reset = 0h]**

 RXHPCRA2 is shown in [Figure 16-165](#) and described in [Table 16-177](#).

**Figure 16-165. RXHPCRA2 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-177. RXHPCRA2 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.15 RXHPCRB2 Register (offset = 850h) [reset = 0h]**

 RXHPCRB2 is shown in [Figure 16-166](#) and described in [Table 16-178](#).

**Figure 16-166. RXHPCRB2 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

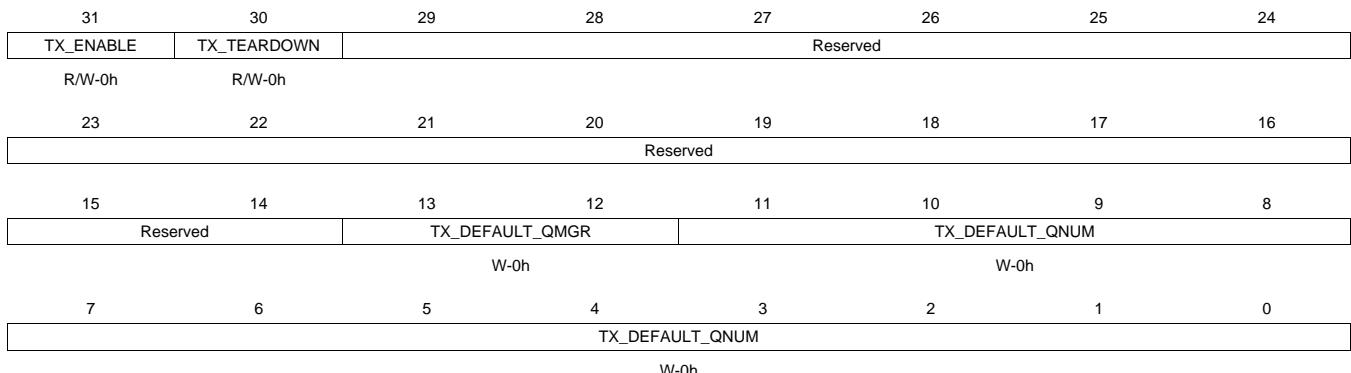
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-178. RXHPCRB2 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.16 TXGCR3 Register (offset = 860h) [reset = 0h]**

TXGCR3 is shown in [Figure 16-167](#) and described in [Table 16-179](#).

**Figure 16-167. TXGCR3 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-179. TXGCR3 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.17 RXGCR3 Register (offset = 868h) [reset = 0h]

RXGCR3 is shown in [Figure 16-168](#) and described in [Table 16-180](#).

**Figure 16-168. RXGCR3 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25 | 24                |
|----------------------|-------------|--------------------|----|----------|--------------------|----|-------------------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    |    | RX_ERROR_HANDLING |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |    | W-0h              |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17 | 16                |
| RX_SOP_OFFSET        |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9  | 8                 |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |    |                   |
| W-0h                 |             | W-0h               |    |          | W-0h               |    |                   |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1  | 0                 |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-180. RXGCR3 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-180. RXGCR3 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.18 RXHPCRA3 Register (offset = 86Ch) [reset = 0h]

RXHPCRA3 is shown in [Figure 16-169](#) and described in [Table 16-181](#).

**Figure 16-169. RXHPCRA3 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-181. RXHPCRA3 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.19 RXHPCRB3 Register (offset = 870h) [reset = 0h]**

 RXHPCRB3 is shown in [Figure 16-170](#) and described in [Table 16-182](#).

**Figure 16-170. RXHPCRB3 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

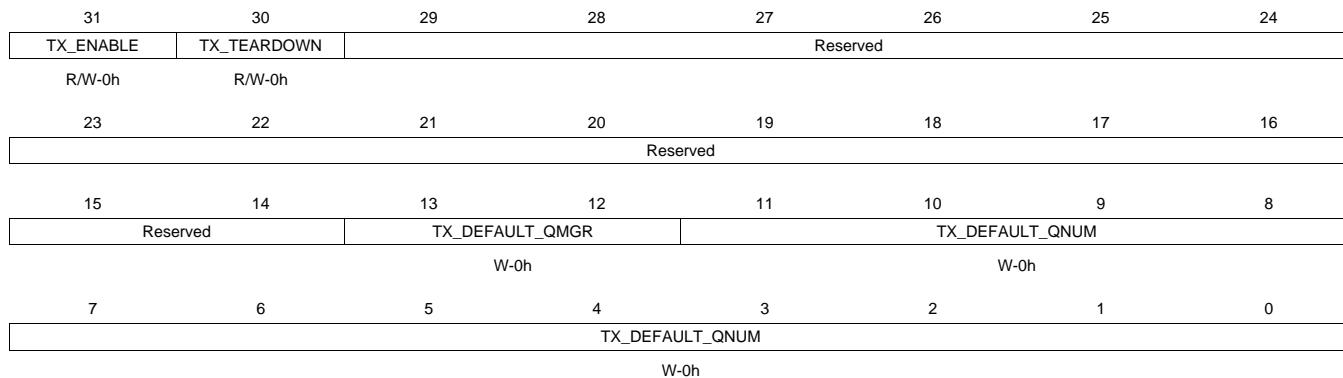
**Table 16-182. RXHPCRB3 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.20 TXGCR4 Register (offset = 880h) [reset = 0h]

TXGCR4 is shown in [Figure 16-171](#) and described in [Table 16-183](#).

**Figure 16-171. TXGCR4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-183. TXGCR4 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.21 RXGCR4 Register (offset = 888h) [reset = 0h]**

RXGCR4 is shown in [Figure 16-172](#) and described in [Table 16-184](#).

**Figure 16-172. RXGCR4 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-184. RXGCR4 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-184. RXGCR4 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.22 RXHPCRA4 Register (offset = 88Ch) [reset = 0h]**

 RXHPCRA4 is shown in [Figure 16-173](#) and described in [Table 16-185](#).

**Figure 16-173. RXHPCRA4 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-185. RXHPCRA4 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.23 RXHPCRB4 Register (offset = 890h) [reset = 0h]**

 RXHPCRB4 is shown in [Figure 16-174](#) and described in [Table 16-186](#).

**Figure 16-174. RXHPCRB4 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

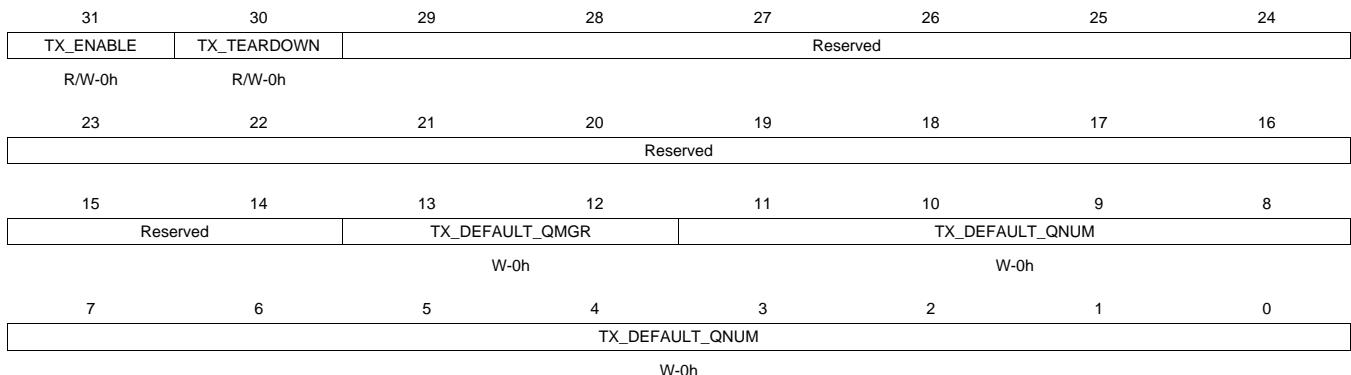
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-186. RXHPCRB4 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.24 TXGCR5 Register (offset = 8A0h) [reset = 0h]**

TXGCR5 is shown in [Figure 16-175](#) and described in [Table 16-187](#).

**Figure 16-175. TXGCR5 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-187. TXGCR5 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.25 RXGCR5 Register (offset = 8A8h) [reset = 0h]

RXGCR5 is shown in [Figure 16-176](#) and described in [Table 16-188](#).

**Figure 16-176. RXGCR5 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-188. RXGCR5 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-188. RXGCR5 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.26 RXHPCRA5 Register (offset = 8ACh) [reset = 0h]**

 RXHPCRA5 is shown in [Figure 16-177](#) and described in [Table 16-189](#).

**Figure 16-177. RXHPCRA5 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-189. RXHPCRA5 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.27 RXHPCRB5 Register (offset = 8B0h) [reset = 0h]**

 RXHPCRB5 is shown in [Figure 16-178](#) and described in [Table 16-190](#).

**Figure 16-178. RXHPCRB5 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

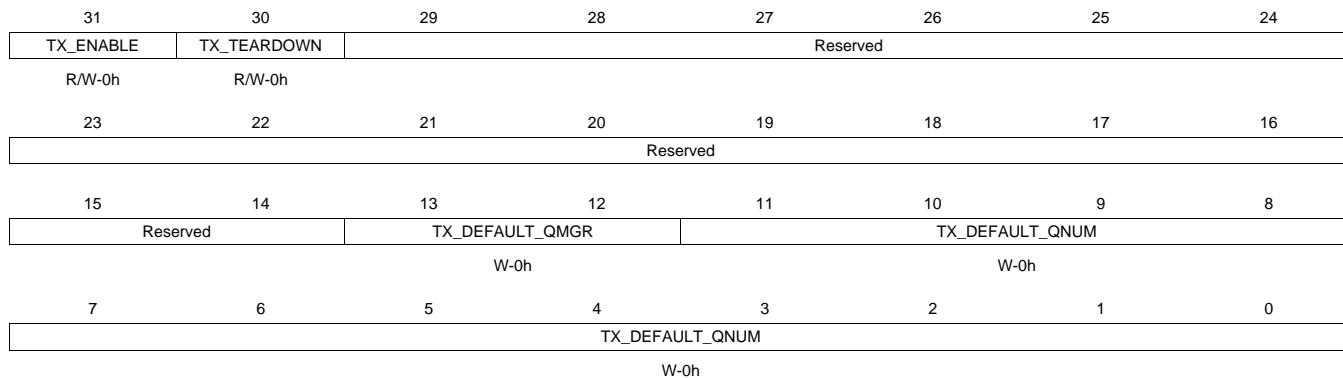
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-190. RXHPCRB5 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.28 TXGCR6 Register (offset = 8C0h) [reset = 0h]**

 TXGCR6 is shown in [Figure 16-179](#) and described in [Table 16-191](#).

**Figure 16-179. TXGCR6 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-191. TXGCR6 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.29 RXGCR6 Register (offset = 8C8h) [reset = 0h]**

RXGCR6 is shown in [Figure 16-180](#) and described in [Table 16-192](#).

**Figure 16-180. RXGCR6 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-192. RXGCR6 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-192. RXGCR6 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.30 RXHPCRA6 Register (offset = 8CCh) [reset = 0h]**

 RXHPCRA6 is shown in [Figure 16-181](#) and described in [Table 16-193](#).

**Figure 16-181. RXHPCRA6 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-193. RXHPCRA6 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.31 RXHPCRB6 Register (offset = 8D0h) [reset = 0h]

RXHPCRB6 is shown in [Figure 16-182](#) and described in [Table 16-194](#).

**Figure 16-182. RXHPCRB6 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDK3_QMGR |    |    |    | RX_HOST_FDK3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDK3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDK2_QMGR |    |    |    | RX_HOST_FDK2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDK2_QNUM |    |    |    | W-0h              |    |    |

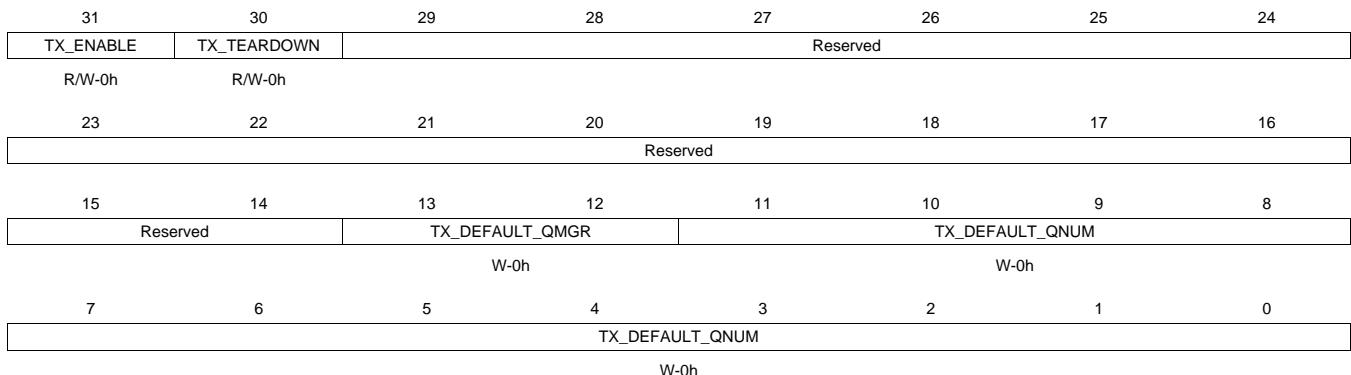
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-194. RXHPCRB6 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDK3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDK3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDK2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDK2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.32 TXGCR7 Register (offset = 8E0h) [reset = 0h]**

TXGCR7 is shown in [Figure 16-183](#) and described in [Table 16-195](#).

**Figure 16-183. TXGCR7 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-195. TXGCR7 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.33 RXGCR7 Register (offset = 8E8h) [reset = 0h]

RXGCR7 is shown in [Figure 16-184](#) and described in [Table 16-196](#).

**Figure 16-184. RXGCR7 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25 | 24                |
|----------------------|-------------|--------------------|----|----------|--------------------|----|-------------------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    |    | RX_ERROR_HANDLING |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |    | W-0h              |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17 | 16                |
| RX_SOP_OFFSET        |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9  | 8                 |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |    |                   |
| W-0h                 |             | W-0h               |    |          | W-0h               |    |                   |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1  | 0                 |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-196. RXGCR7 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-196. RXGCR7 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.34 RXHPCRA7 Register (offset = 8ECh) [reset = 0h]**

 RXHPCRA7 is shown in [Figure 16-185](#) and described in [Table 16-197](#).

**Figure 16-185. RXHPCRA7 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-197. RXHPCRA7 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.35 RXHPCRB7 Register (offset = 8F0h) [reset = 0h]**

 RXHPCRB7 is shown in [Figure 16-186](#) and described in [Table 16-198](#).

**Figure 16-186. RXHPCRB7 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ3_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ2_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

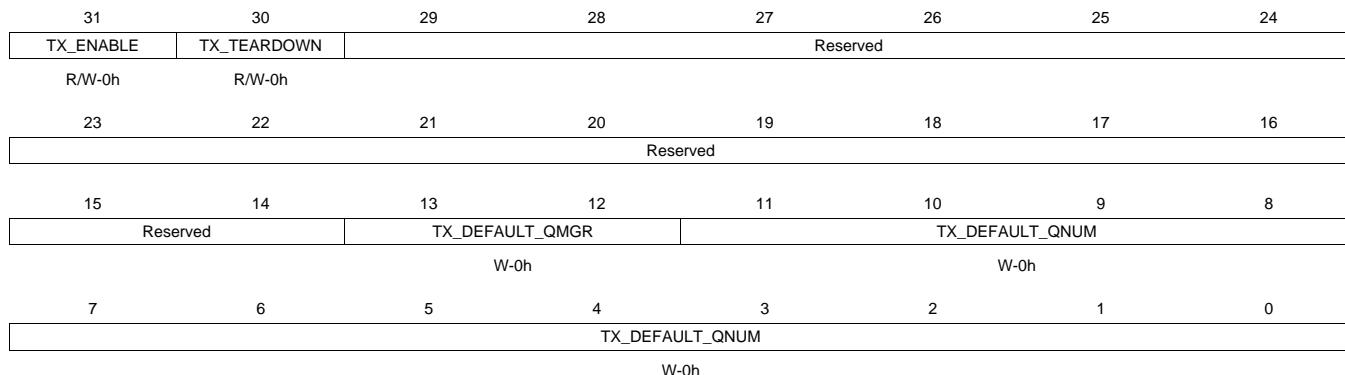
**Table 16-198. RXHPCRB7 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.36 TXGCR8 Register (offset = 900h) [reset = 0h]

TXGCR8 is shown in [Figure 16-187](#) and described in [Table 16-199](#).

**Figure 16-187. TXGCR8 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-199. TXGCR8 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.37 RXGCR8 Register (offset = 908h) [reset = 0h]**

RXGCR8 is shown in [Figure 16-188](#) and described in [Table 16-200](#).

**Figure 16-188. RXGCR8 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-200. RXGCR8 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-200. RXGCR8 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.38 RXHPCRA8 Register (offset = 90Ch) [reset = 0h]**

 RXHPCRA8 is shown in [Figure 16-189](#) and described in [Table 16-201](#).

**Figure 16-189. RXHPCRA8 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-201. RXHPCRA8 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.39 RXHPCRB8 Register (offset = 910h) [reset = 0h]**

 RXHPCRB8 is shown in [Figure 16-190](#) and described in [Table 16-202](#).

**Figure 16-190. RXHPCRB8 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

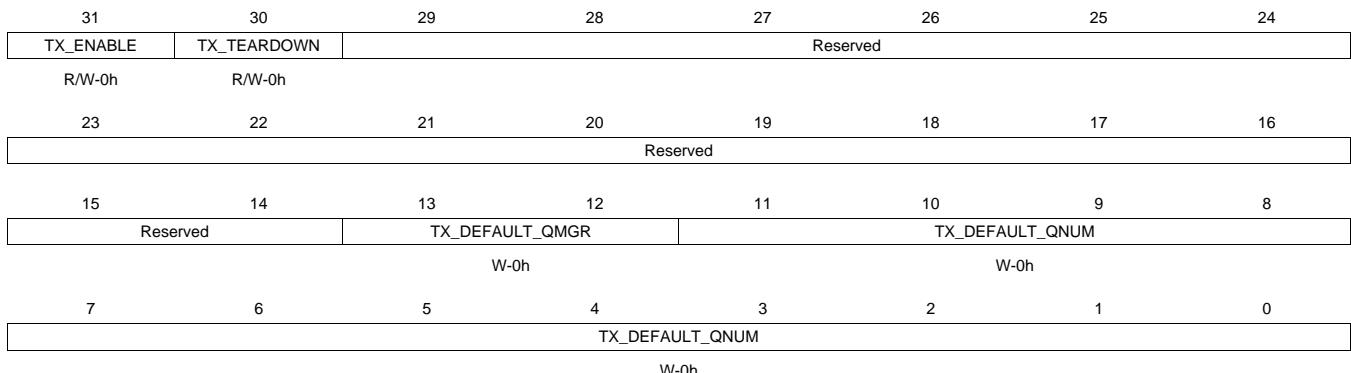
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-202. RXHPCRB8 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.40 TXGCR9 Register (offset = 920h) [reset = 0h]**

TXGCR9 is shown in [Figure 16-191](#) and described in [Table 16-203](#).

**Figure 16-191. TXGCR9 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-203. TXGCR9 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

#### 16.5.5.41 RXGCR9 Register (offset = 928h) [reset = 0h]

RXGCR9 is shown in [Figure 16-192](#) and described in [Table 16-204](#).

**Figure 16-192. RXGCR9 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25 | 24                |
|----------------------|-------------|--------------------|----|----------|--------------------|----|-------------------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    |    | RX_ERROR_HANDLING |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |    | W-0h              |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17 | 16                |
| RX_SOP_OFFSET        |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9  | 8                 |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |    |                   |
| W-0h                 |             | W-0h               |    |          | W-0h               |    |                   |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1  | 0                 |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |    |                   |
|                      |             |                    |    |          |                    |    | W-0h              |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-204. RXGCR9 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-204. RXGCR9 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.42 RXHPCRA9 Register (offset = 92Ch) [reset = 0h]**

 RXHPCRA9 is shown in [Figure 16-193](#) and described in [Table 16-205](#).

**Figure 16-193. RXHPCRA9 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-205. RXHPCRA9 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.43 RXHPCRB9 Register (offset = 930h) [reset = 0h]**

 RXHPCRB9 is shown in [Figure 16-194](#) and described in [Table 16-206](#).

**Figure 16-194. RXHPCRB9 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ3_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ2_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

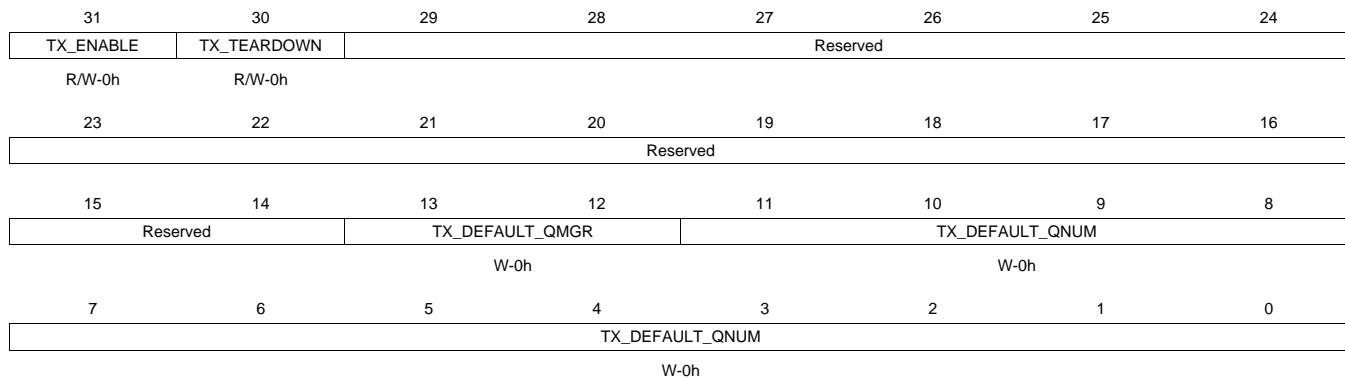
**Table 16-206. RXHPCRB9 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

#### 16.5.5.44 TXGCR10 Register (offset = 940h) [reset = 0h]

TXGCR10 is shown in [Figure 16-195](#) and described in [Table 16-207](#).

**Figure 16-195. TXGCR10 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-207. TXGCR10 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.45 RXGCR10 Register (offset = 948h) [reset = 0h]**

 RXGCR10 is shown in [Figure 16-196](#) and described in [Table 16-208](#).

**Figure 16-196. RXGCR10 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-208. RXGCR10 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-208. RXGCR10 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.46 RXHPCRA10 Register (offset = 94Ch) [reset = 0h]**

RXHPCRA10 is shown in [Figure 16-197](#) and described in [Table 16-209](#).

**Figure 16-197. RXHPCRA10 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-209. RXHPCRA10 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

#### 16.5.5.47 RXHPCRB10 Register (offset = 950h) [reset = 0h]

RXHPCRB10 is shown in [Figure 16-198](#) and described in [Table 16-210](#).

**Figure 16-198. RXHPCRB10 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ3_QMGR |      |    | RX_HOST_FDQ3_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ3_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ2_QMGR |      |    | RX_HOST_FDQ2_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ2_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |

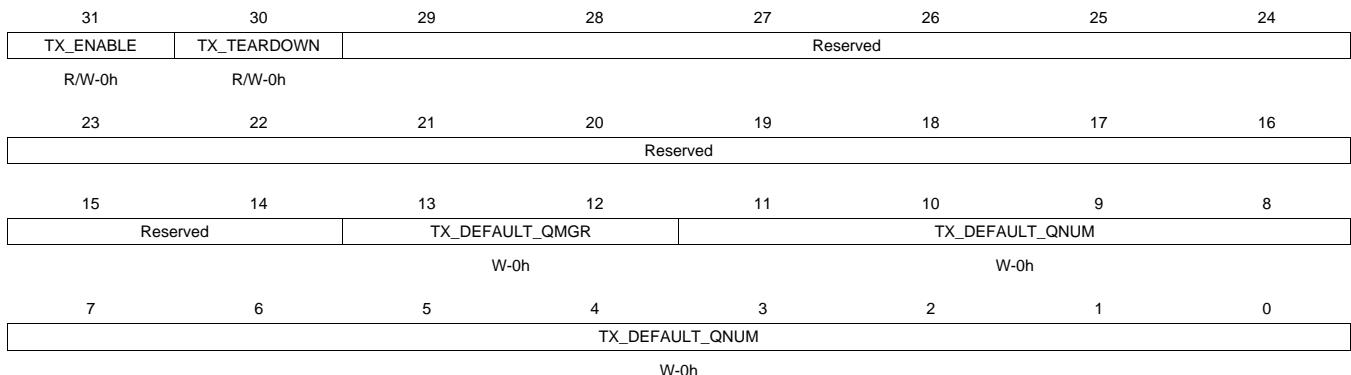
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-210. RXHPCRB10 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.48 TXGCR11 Register (offset = 960h) [reset = 0h]**

 TXGCR11 is shown in [Figure 16-199](#) and described in [Table 16-211](#).

**Figure 16-199. TXGCR11 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-211. TXGCR11 Register Field Descriptions**

| Bit   | Field                  | Type | Reset | Description   |
|-------|------------------------|------|-------|---|
| 31    | <b>TX_ENABLE</b>       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | <b>TX_TEARDOWN</b>     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | <b>TX_DEFAULT_QMGR</b> | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | <b>TX_DEFAULT_QNUM</b> | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.49 RXGCR11 Register (offset = 968h) [reset = 0h]**

 RXGCR11 is shown in [Figure 16-200](#) and described in [Table 16-212](#).

**Figure 16-200. RXGCR11 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-212. RXGCR11 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-212. RXGCR11 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.50 RXHPCRA11 Register (offset = 96Ch) [reset = 0h]

RXHPCRA11 is shown in [Figure 16-201](#) and described in [Table 16-213](#).

**Figure 16-201. RXHPCRA11 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ1_QMGR |      |    | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ1_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |      |    | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ0_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-213. RXHPCRA11 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.51 RXHPCRB11 Register (offset = 970h) [reset = 0h]

RXHPCRB11 is shown in [Figure 16-202](#) and described in [Table 16-214](#).

**Figure 16-202. RXHPCRB11 Register**

|          |                   |    |      |    |                   |      |    |
|----------|-------------------|----|------|----|-------------------|------|----|
| 31       | 30                | 29 | 28   | 27 | 26                | 25   | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |      |    |
|          |                   |    | W-0h |    |                   | W-0h |    |
| 23       | 22                | 21 | 20   | 19 | 18                | 17   | 16 |
|          | RX_HOST_FDQ3_QNUM |    |      |    | W-0h              |      |    |
| 15       | 14                | 13 | 12   | 11 | 10                | 9    | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |      |    |
|          |                   |    | W-0h |    |                   | W-0h |    |
| 7        | 6                 | 5  | 4    | 3  | 2                 | 1    | 0  |
|          | RX_HOST_FDQ2_QNUM |    |      |    | W-0h              |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

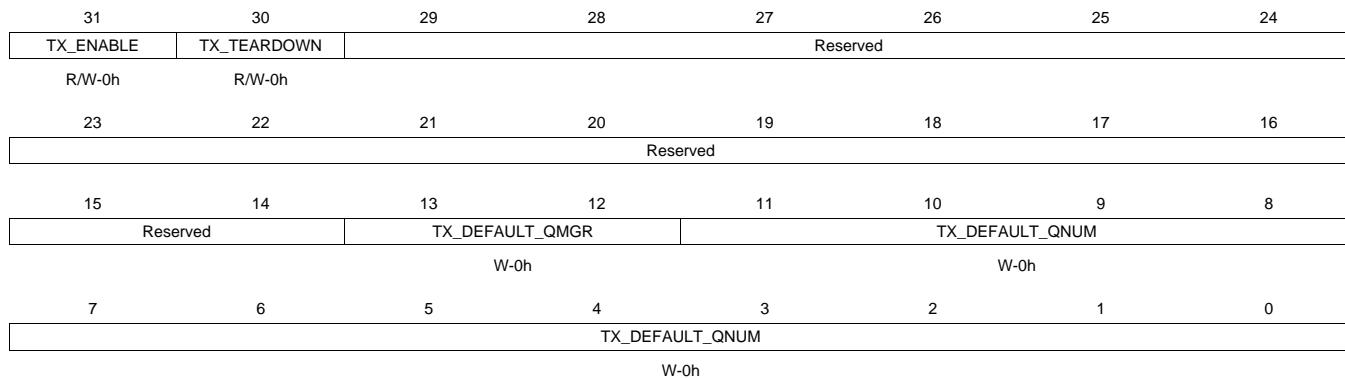
**Table 16-214. RXHPCRB11 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.52 TXGCR12 Register (offset = 980h) [reset = 0h]

TXGCR12 is shown in [Figure 16-203](#) and described in [Table 16-215](#).

**Figure 16-203. TXGCR12 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-215. TXGCR12 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.53 RXGCR12 Register (offset = 988h) [reset = 0h]**

RXGCR12 is shown in [Figure 16-204](#) and described in [Table 16-216](#).

**Figure 16-204. RXGCR12 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-216. RXGCR12 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-216. RXGCR12 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.54 RXHPCRA12 Register (offset = 98Ch) [reset = 0h]**

RXHPCRA12 is shown in [Figure 16-205](#) and described in [Table 16-217](#).

**Figure 16-205. RXHPCRA12 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-217. RXHPCRA12 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.55 RXHPCRB12 Register (offset = 990h) [reset = 0h]

RXHPCRB12 is shown in [Figure 16-206](#) and described in [Table 16-218](#).

**Figure 16-206. RXHPCRB12 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ3_QMGR |      |    | RX_HOST_FDQ3_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ3_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ2_QMGR |      |    | RX_HOST_FDQ2_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ2_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |

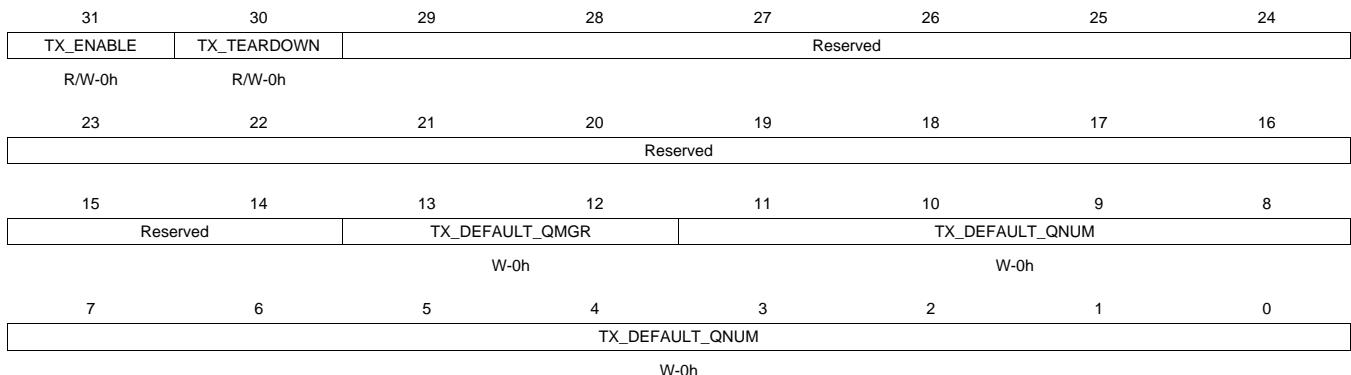
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-218. RXHPCRB12 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.56 TXGCR13 Register (offset = 9A0h) [reset = 0h]**

 TXGCR13 is shown in [Figure 16-207](#) and described in [Table 16-219](#).

**Figure 16-207. TXGCR13 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-219. TXGCR13 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.57 RXGCR13 Register (offset = 9A8h) [reset = 0h]

RXGCR13 is shown in [Figure 16-208](#) and described in [Table 16-220](#).

**Figure 16-208. RXGCR13 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-220. RXGCR13 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-220. RXGCR13 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.58 RXHPCRA13 Register (offset = 9ACh) [reset = 0h]

RXHPCRA13 is shown in [Figure 16-209](#) and described in [Table 16-221](#).

**Figure 16-209. RXHPCRA13 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-221. RXHPCRA13 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.59 RXHPCRB13 Register (offset = 9B0h) [reset = 0h]**

RXHPCRB13 is shown in [Figure 16-210](#) and described in [Table 16-222](#).

**Figure 16-210. RXHPCRB13 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

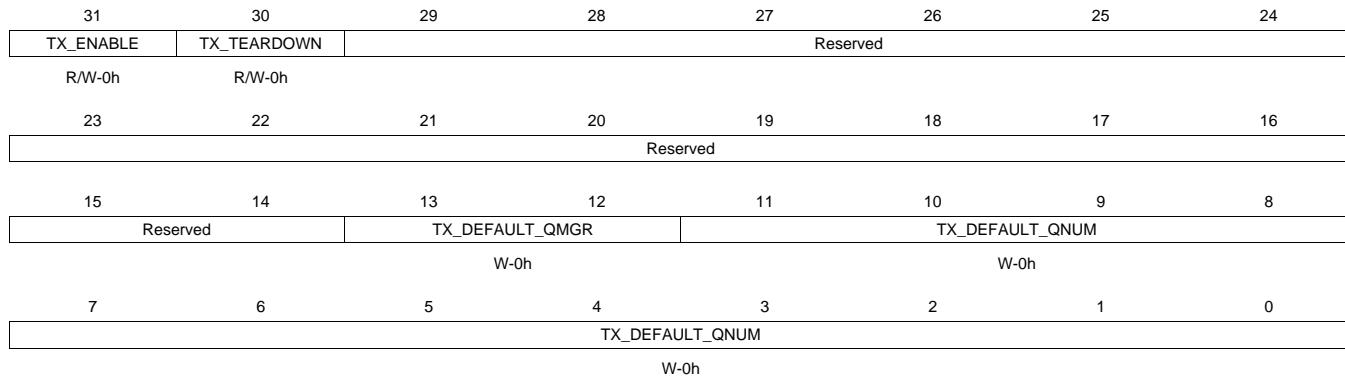
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-222. RXHPCRB13 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.60 TXGCR14 Register (offset = 9C0h) [reset = 0h]**

 TXGCR14 is shown in [Figure 16-211](#) and described in [Table 16-223](#).

**Figure 16-211. TXGCR14 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-223. TXGCR14 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.61 RXGCR14 Register (offset = 9C8h) [reset = 0h]**

RXGCR14 is shown in [Figure 16-212](#) and described in [Table 16-224](#).

**Figure 16-212. RXGCR14 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-224. RXGCR14 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-224. RXGCR14 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.62 RXHPCRA14 Register (offset = 9CCh) [reset = 0h]

RXHPCRA14 is shown in [Figure 16-213](#) and described in [Table 16-225](#).

**Figure 16-213. RXHPCRA14 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-225. RXHPCRA14 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.63 RXHPCRB14 Register (offset = 9D0h) [reset = 0h]

RXHPCRB14 is shown in [Figure 16-214](#) and described in [Table 16-226](#).

**Figure 16-214. RXHPCRB14 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDK3_QMGR |      |    | RX_HOST_FDK3_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDK3_QNUM |      |    | W-0h              |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDK2_QMGR |      |    | RX_HOST_FDK2_QNUM |      |    |
|          |    | W-0h              |      |    | W-0h              |      |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDK2_QNUM |      |    | W-0h              |      |    |

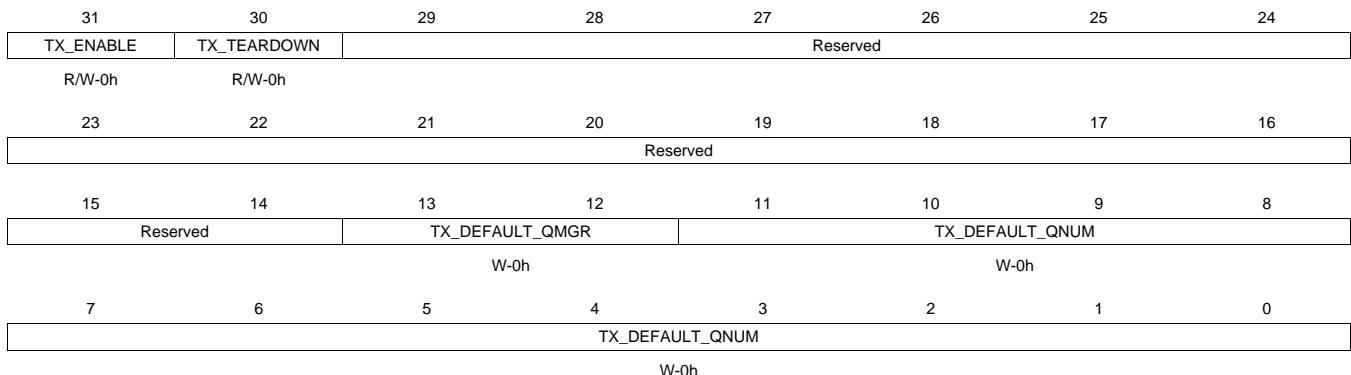
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-226. RXHPCRB14 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDK3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDK3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDK2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDK2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.64 TXGCR15 Register (offset = 9E0h) [reset = 0h]**

TXGCR15 is shown in [Figure 16-215](#) and described in [Table 16-227](#).

**Figure 16-215. TXGCR15 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-227. TXGCR15 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.65 RXGCR15 Register (offset = 9E8h) [reset = 0h]

RXGCR15 is shown in [Figure 16-216](#) and described in [Table 16-228](#).

**Figure 16-216. RXGCR15 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-228. RXGCR15 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-228. RXGCR15 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.66 RXHPCRA15 Register (offset = 9ECh) [reset = 0h]

RXHPCRA15 is shown in [Figure 16-217](#) and described in [Table 16-229](#).

**Figure 16-217. RXHPCRA15 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-229. RXHPCRA15 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.67 RXHPCRB15 Register (offset = 9F0h) [reset = 0h]

RXHPCRB15 is shown in [Figure 16-218](#) and described in [Table 16-230](#).

**Figure 16-218. RXHPCRB15 Register**

|          |    |                   |      |      |                   |      |    |
|----------|----|-------------------|------|------|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27   | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ3_QMGR |      |      | RX_HOST_FDQ3_QNUM |      |    |
|          |    |                   | W-0h |      |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19   | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ3_QNUM |      |      |                   |      |    |
|          |    |                   |      | W-0h |                   |      |    |
| 15       | 14 | 13                | 12   | 11   | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ2_QMGR |      |      | RX_HOST_FDQ2_QNUM |      |    |
|          |    |                   | W-0h |      |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3    | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ2_QNUM |      |      |                   |      |    |
|          |    |                   |      | W-0h |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

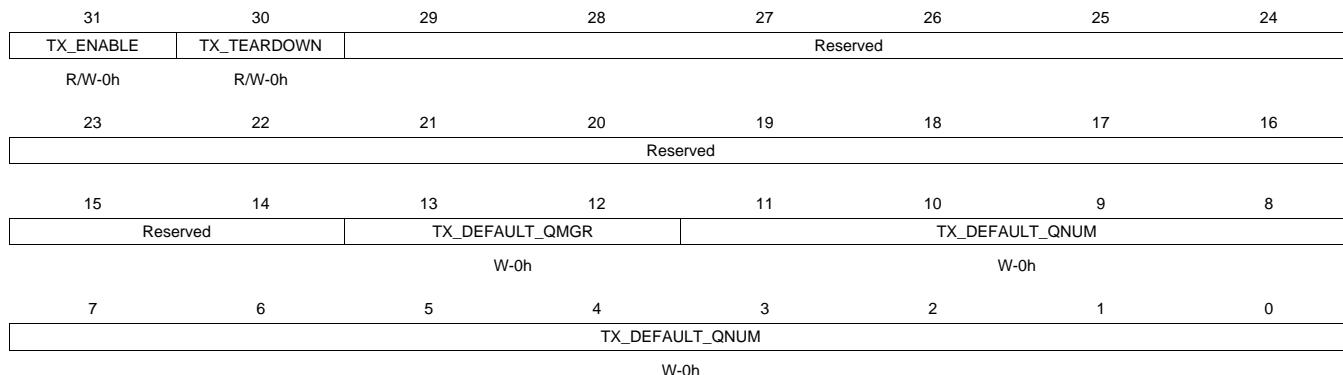
**Table 16-230. RXHPCRB15 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.68 TXGCR16 Register (offset = A00h) [reset = 0h]

TXGCR16 is shown in [Figure 16-219](#) and described in [Table 16-231](#).

**Figure 16-219. TXGCR16 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-231. TXGCR16 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.69 RXGCR16 Register (offset = A08h) [reset = 0h]**

RXGCR16 is shown in [Figure 16-220](#) and described in [Table 16-232](#).

**Figure 16-220. RXGCR16 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-232. RXGCR16 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-232. RXGCR16 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.70 RXHPCRA16 Register (offset = A0Ch) [reset = 0h]**

RXHPCRA16 is shown in [Figure 16-221](#) and described in [Table 16-233](#).

**Figure 16-221. RXHPCRA16 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-233. RXHPCRA16 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.71 RXHPCRB16 Register (offset = A10h) [reset = 0h]

RXHPCRB16 is shown in [Figure 16-222](#) and described in [Table 16-234](#).

**Figure 16-222. RXHPCRB16 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ3_QMGR |      |    | RX_HOST_FDQ3_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ3_QNUM |      |    | W-0h              |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ2_QMGR |      |    | RX_HOST_FDQ2_QNUM |      |    |
|          |    | W-0h              |      |    | W-0h              |      |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ2_QNUM |      |    | W-0h              |      |    |

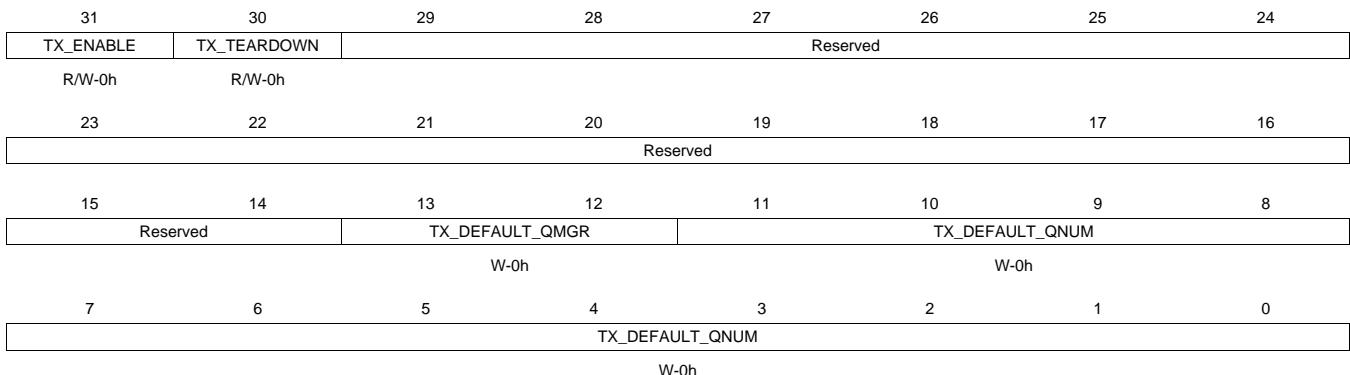
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-234. RXHPCRB16 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.72 TXGCR17 Register (offset = A20h) [reset = 0h]**

TXGCR17 is shown in [Figure 16-223](#) and described in [Table 16-235](#).

**Figure 16-223. TXGCR17 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-235. TXGCR17 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.73 RXGCR17 Register (offset = A28h) [reset = 0h]

RXGCR17 is shown in [Figure 16-224](#) and described in [Table 16-236](#).

**Figure 16-224. RXGCR17 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25 | 24                |
|----------------------|-------------|--------------------|----|----------|--------------------|----|-------------------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    |    | RX_ERROR_HANDLING |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |    | W-0h              |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17 | 16                |
| RX_SOP_OFFSET        |             |                    |    |          |                    |    |                   |
| W-0h                 |             |                    |    |          |                    |    |                   |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9  | 8                 |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |    |                   |
| W-0h                 |             | W-0h               |    |          | W-0h               |    |                   |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1  | 0                 |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |    |                   |
| W-0h                 |             |                    |    |          |                    |    |                   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-236. RXGCR17 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-236. RXGCR17 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.74 RXHPCRA17 Register (offset = A2Ch) [reset = 0h]

RXHPCRA17 is shown in [Figure 16-225](#) and described in [Table 16-237](#).

**Figure 16-225. RXHPCRA17 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-237. RXHPCRA17 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.75 RXHPCRB17 Register (offset = A30h) [reset = 0h]**

RXHPCRB17 is shown in [Figure 16-226](#) and described in [Table 16-238](#).

**Figure 16-226. RXHPCRB17 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

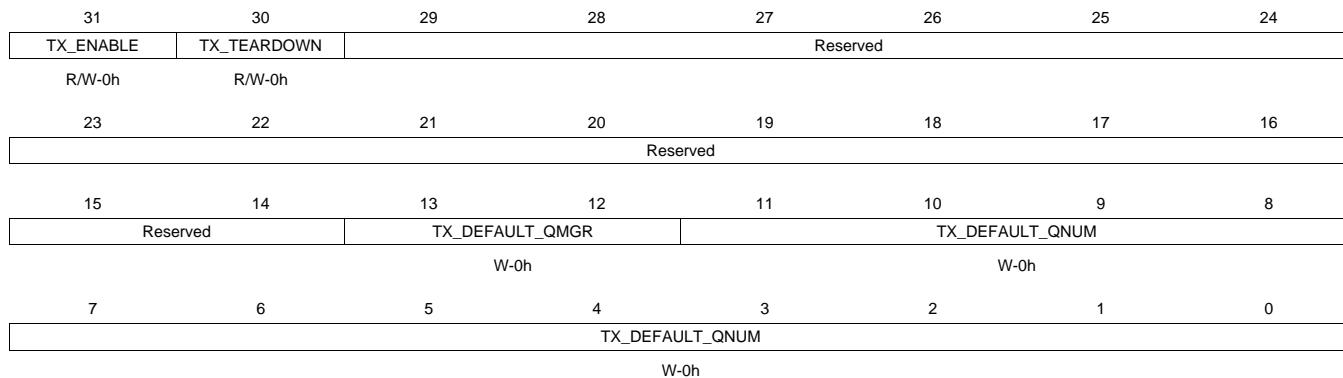
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-238. RXHPCRB17 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.76 TXGCR18 Register (offset = A40h) [reset = 0h]**

 TXGCR18 is shown in [Figure 16-227](#) and described in [Table 16-239](#).

**Figure 16-227. TXGCR18 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-239. TXGCR18 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.77 RXGCR18 Register (offset = A48h) [reset = 0h]**

RXGCR18 is shown in [Figure 16-228](#) and described in [Table 16-240](#).

**Figure 16-228. RXGCR18 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-240. RXGCR18 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-240. RXGCR18 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.78 RXHPCRA18 Register (offset = A4Ch) [reset = 0h]**

RXHPCRA18 is shown in [Figure 16-229](#) and described in [Table 16-241](#).

**Figure 16-229. RXHPCRA18 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ1_QMGR |    |      |    | RX_HOST_FDQ1_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ1_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ0_QMGR |    |      |    | RX_HOST_FDQ0_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ0_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-241. RXHPCRA18 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.79 RXHPCRB18 Register (offset = A50h) [reset = 0h]

RXHPCRB18 is shown in [Figure 16-230](#) and described in [Table 16-242](#).

**Figure 16-230. RXHPCRB18 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ3_QMGR |      |    | RX_HOST_FDQ3_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ3_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ2_QMGR |      |    | RX_HOST_FDQ2_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ2_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |

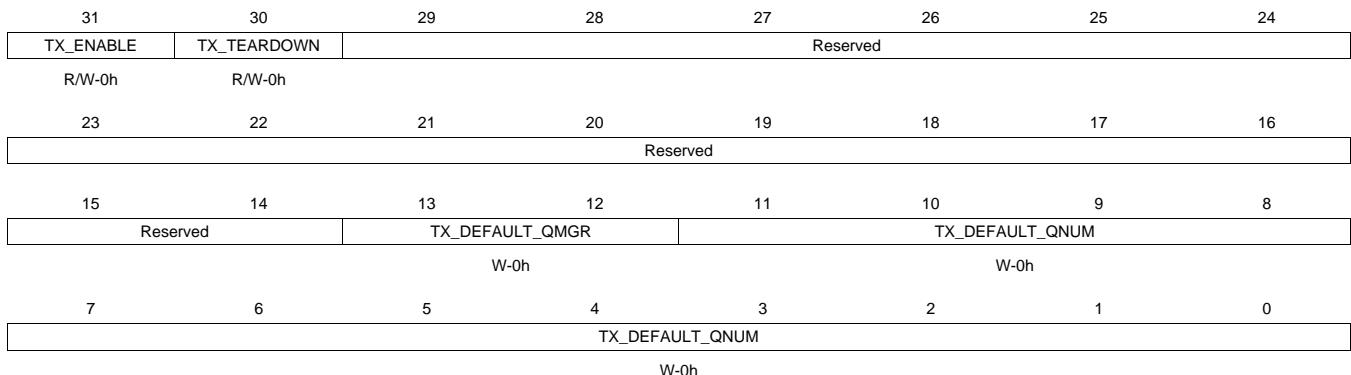
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-242. RXHPCRB18 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.80 TXGCR19 Register (offset = A60h) [reset = 0h]**

 TXGCR19 is shown in [Figure 16-231](#) and described in [Table 16-243](#).

**Figure 16-231. TXGCR19 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-243. TXGCR19 Register Field Descriptions**

| Bit   | Field                  | Type | Reset | Description   |
|-------|------------------------|------|-------|---|
| 31    | <b>TX_ENABLE</b>       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | <b>TX_TEARDOWN</b>     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | <b>TX_DEFAULT_QMGR</b> | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | <b>TX_DEFAULT_QNUM</b> | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.81 RXGCR19 Register (offset = A68h) [reset = 0h]

RXGCR19 is shown in [Figure 16-232](#) and described in [Table 16-244](#).

**Figure 16-232. RXGCR19 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25 | 24                |
|----------------------|-------------|--------------------|----|----------|--------------------|----|-------------------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    |    | RX_ERROR_HANDLING |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |    | W-0h              |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17 | 16                |
| RX_SOP_OFFSET        |             |                    |    |          |                    |    |                   |
| W-0h                 |             |                    |    |          |                    |    |                   |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9  | 8                 |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |    |                   |
| W-0h                 |             | W-0h               |    |          | W-0h               |    |                   |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1  | 0                 |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |    |                   |
| W-0h                 |             |                    |    |          |                    |    |                   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-244. RXGCR19 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-244. RXGCR19 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.82 RXHPCRA19 Register (offset = A6Ch) [reset = 0h]

RXHPCRA19 is shown in [Figure 16-233](#) and described in [Table 16-245](#).

**Figure 16-233. RXHPCRA19 Register**

|          |    |                   |    |    |                   |    |    |
|----------|----|-------------------|----|----|-------------------|----|----|
| 31       | 30 | 29                | 28 | 27 | 26                | 25 | 24 |
| Reserved |    | RX_HOST_FDQ1_QMGR |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          |    | W-0h              |    |    |                   |    |    |
| 23       | 22 | 21                | 20 | 19 | 18                | 17 | 16 |
|          |    | RX_HOST_FDQ1_QNUM |    |    | W-0h              |    |    |
| 15       | 14 | 13                | 12 | 11 | 10                | 9  | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          |    | W-0h              |    |    |                   |    |    |
| 7        | 6  | 5                 | 4  | 3  | 2                 | 1  | 0  |
|          |    | RX_HOST_FDQ0_QNUM |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-245. RXHPCRA19 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.83 RXHPCRB19 Register (offset = A70h) [reset = 0h]**

RXHPCRB19 is shown in [Figure 16-234](#) and described in [Table 16-246](#).

**Figure 16-234. RXHPCRB19 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

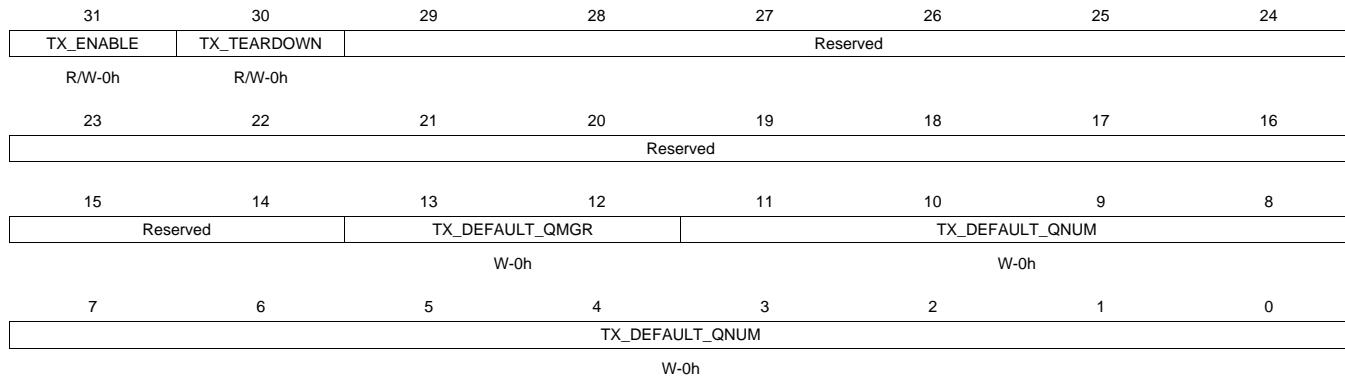
**Table 16-246. RXHPCRB19 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

#### 16.5.5.84 TXGCR20 Register (offset = A80h) [reset = 0h]

TXGCR20 is shown in [Figure 16-235](#) and described in [Table 16-247](#).

**Figure 16-235. TXGCR20 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-247. TXGCR20 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.85 RXGCR20 Register (offset = A88h) [reset = 0h]**

RXGCR20 is shown in [Figure 16-236](#) and described in [Table 16-248](#).

**Figure 16-236. RXGCR20 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-248. RXGCR20 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-248. RXGCR20 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.86 RXHPCRA20 Register (offset = A8Ch) [reset = 0h]

RXHPCRA20 is shown in [Figure 16-237](#) and described in [Table 16-249](#).

**Figure 16-237. RXHPCRA20 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-249. RXHPCRA20 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.87 RXHPCRB20 Register (offset = A90h) [reset = 0h]

RXHPCRB20 is shown in [Figure 16-238](#) and described in [Table 16-250](#).

**Figure 16-238. RXHPCRB20 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

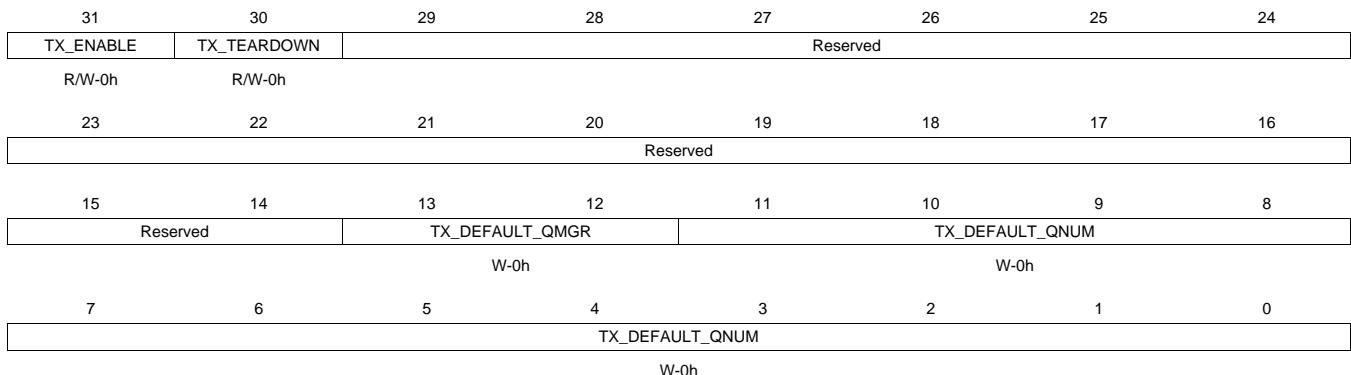
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-250. RXHPCRB20 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.88 TXGCR21 Register (offset = AA0h) [reset = 0h]**

TXGCR21 is shown in [Figure 16-239](#) and described in [Table 16-251](#).

**Figure 16-239. TXGCR21 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-251. TXGCR21 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.89 RXGCR21 Register (offset = AA8h) [reset = 0h]

RXGCR21 is shown in [Figure 16-240](#) and described in [Table 16-252](#).

**Figure 16-240. RXGCR21 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-252. RXGCR21 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-252. RXGCR21 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.90 RXHPCRA21 Register (offset = AACh) [reset = 0h]

RXHPCRA21 is shown in [Figure 16-241](#) and described in [Table 16-253](#).

**Figure 16-241. RXHPCRA21 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-253. RXHPCRA21 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.91 RXHPCRB21 Register (offset = AB0h) [reset = 0h]

RXHPCRB21 is shown in [Figure 16-242](#) and described in [Table 16-254](#).

**Figure 16-242. RXHPCRB21 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

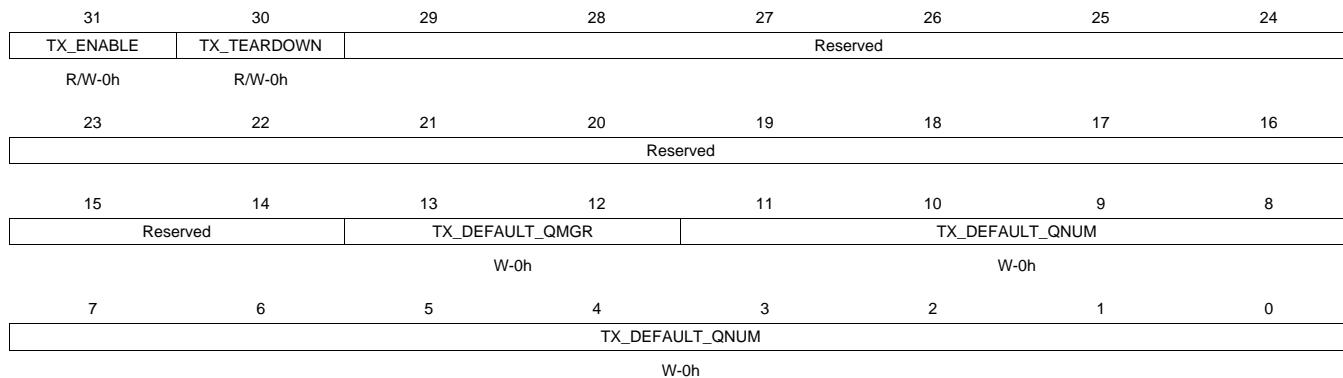
**Table 16-254. RXHPCRB21 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.92 TXGCR22 Register (offset = AC0h) [reset = 0h]

TXGCR22 is shown in [Figure 16-243](#) and described in [Table 16-255](#).

**Figure 16-243. TXGCR22 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-255. TXGCR22 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.93 RXGCR22 Register (offset = AC8h) [reset = 0h]**

RXGCR22 is shown in [Figure 16-244](#) and described in [Table 16-256](#).

**Figure 16-244. RXGCR22 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-256. RXGCR22 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-256. RXGCR22 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.94 RXHPCRA22 Register (offset = ACCh) [reset = 0h]**

RXHPCRA22 is shown in [Figure 16-245](#) and described in [Table 16-257](#).

**Figure 16-245. RXHPCRA22 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-257. RXHPCRA22 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.95 RXHPCRB22 Register (offset = AD0h) [reset = 0h]

RXHPCRB22 is shown in [Figure 16-246](#) and described in [Table 16-258](#).

**Figure 16-246. RXHPCRB22 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

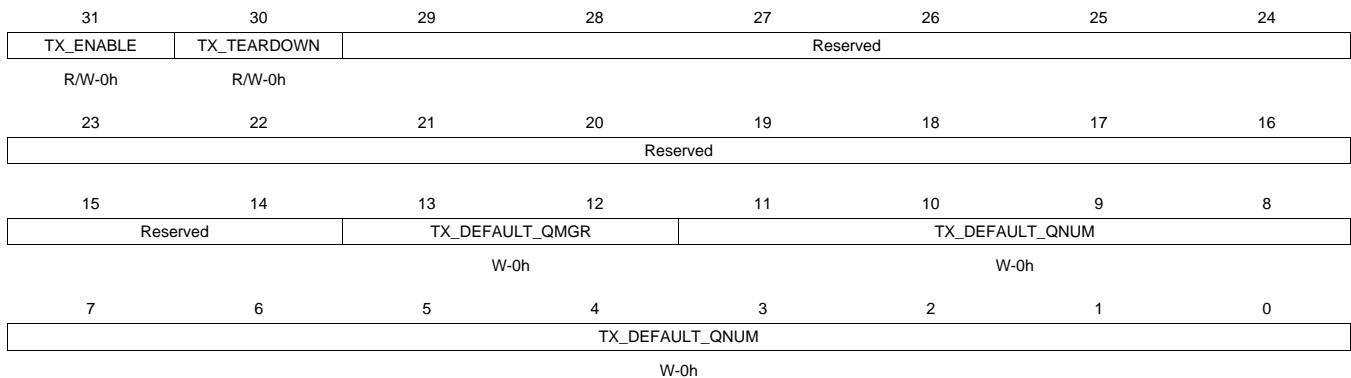
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-258. RXHPCRB22 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.96 TXGCR23 Register (offset = AE0h) [reset = 0h]**

 TXGCR23 is shown in [Figure 16-247](#) and described in [Table 16-259](#).

**Figure 16-247. TXGCR23 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-259. TXGCR23 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.97 RXGCR23 Register (offset = AE8h) [reset = 0h]

RXGCR23 is shown in [Figure 16-248](#) and described in [Table 16-260](#).

**Figure 16-248. RXGCR23 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-260. RXGCR23 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-260. RXGCR23 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.98 RXHPCRA23 Register (offset = AECh) [reset = 0h]

RXHPCRA23 is shown in [Figure 16-249](#) and described in [Table 16-261](#).

**Figure 16-249. RXHPCRA23 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-261. RXHPCRA23 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.99 RXHPCRB23 Register (offset = AF0h) [reset = 0h]**

RXHPCRB23 is shown in [Figure 16-250](#) and described in [Table 16-262](#).

**Figure 16-250. RXHPCRB23 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

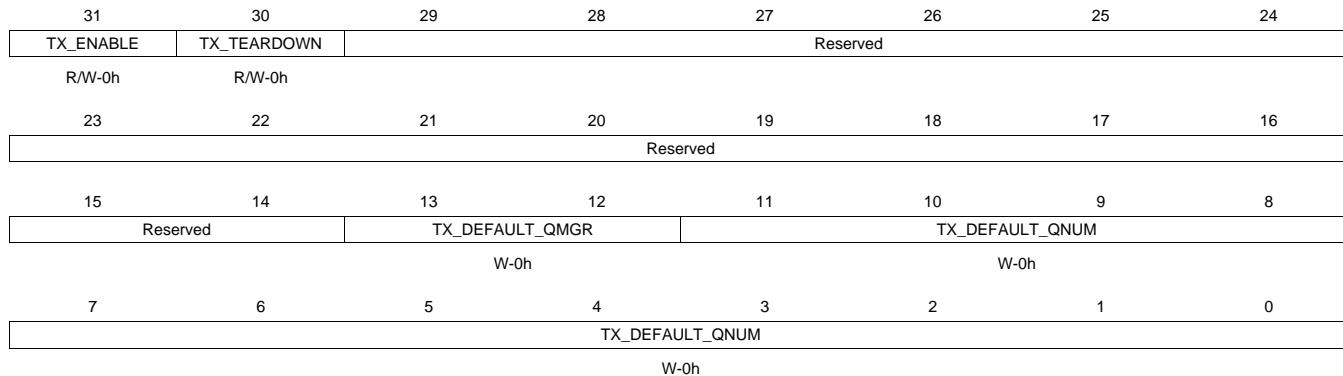
**Table 16-262. RXHPCRB23 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.100 TXGCR24 Register (offset = B00h) [reset = 0h]

TXGCR24 is shown in [Figure 16-251](#) and described in [Table 16-263](#).

**Figure 16-251. TXGCR24 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-263. TXGCR24 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.101 RXGCR24 Register (offset = B08h) [reset = 0h]

RXGCR24 is shown in [Figure 16-252](#) and described in [Table 16-264](#).

**Figure 16-252. RXGCR24 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-264. RXGCR24 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-264. RXGCR24 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.102 RXHPCRA24 Register (offset = B0Ch) [reset = 0h]

RXHPCRA24 is shown in [Figure 16-253](#) and described in [Table 16-265](#).

**Figure 16-253. RXHPCRA24 Register**

| 31       | 30 | 29                | 28                | 27   | 26                | 25   | 24 |
|----------|----|-------------------|-------------------|------|-------------------|------|----|
| Reserved |    | RX_HOST_FDQ1_QMGR |                   |      | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 23       | 22 | 21                | 20                | 19   | 18                | 17   | 16 |
|          |    |                   | RX_HOST_FDQ1_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |
| 15       | 14 | 13                | 12                | 11   | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |                   |      | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 7        | 6  | 5                 | 4                 | 3    | 2                 | 1    | 0  |
|          |    |                   | RX_HOST_FDQ0_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-265. RXHPCRA24 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.103 RXHPCRB24 Register (offset = B10h) [reset = 0h]

RXHPCRB24 is shown in [Figure 16-254](#) and described in [Table 16-266](#).

**Figure 16-254. RXHPCRB24 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-266. RXHPCRB24 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.104 TXGCR25 Register (offset = B20h) [reset = 0h]**

TXGCR25 is shown in [Figure 16-255](#) and described in [Table 16-267](#).

**Figure 16-255. TXGCR25 Register**

| 31        | 30          | 29              | 28   | 27              | 26              | 25   | 24 |
|-----------|-------------|-----------------|------|-----------------|-----------------|------|----|
| TX_ENABLE | TX_TEARDOWN |                 |      | Reserved        |                 |      |    |
| R/W-0h    | R/W-0h      |                 |      |                 |                 |      |    |
| 23        | 22          | 21              | 20   | 19              | 18              | 17   | 16 |
|           |             |                 |      | Reserved        |                 |      |    |
| 15        | 14          | 13              | 12   | 11              | 10              | 9    | 8  |
| Reserved  |             | TX_DEFAULT_QMGR |      |                 | TX_DEFAULT_QNUM |      |    |
|           |             |                 | W-0h |                 |                 | W-0h |    |
| 7         | 6           | 5               | 4    | 3               | 2               | 1    | 0  |
|           |             |                 |      | TX_DEFAULT_QNUM |                 |      |    |
|           |             |                 |      |                 | W-0h            |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-267. TXGCR25 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.105 RXGCR25 Register (offset = B28h) [reset = 0h]

RXGCR25 is shown in [Figure 16-256](#) and described in [Table 16-268](#).

**Figure 16-256. RXGCR25 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-268. RXGCR25 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-268. RXGCR25 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.106 RXHPCRA25 Register (offset = B2Ch) [reset = 0h]

RXHPCRA25 is shown in [Figure 16-257](#) and described in [Table 16-269](#).

**Figure 16-257. RXHPCRA25 Register**

|          |    |                   |                   |      |                   |      |    |
|----------|----|-------------------|-------------------|------|-------------------|------|----|
| 31       | 30 | 29                | 28                | 27   | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ1_QMGR |                   |      | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 23       | 22 | 21                | 20                | 19   | 18                | 17   | 16 |
|          |    |                   | RX_HOST_FDQ1_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |
| 15       | 14 | 13                | 12                | 11   | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |                   |      | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 7        | 6  | 5                 | 4                 | 3    | 2                 | 1    | 0  |
|          |    |                   | RX_HOST_FDQ0_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-269. RXHPCRA25 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.107 RXHPCRB25 Register (offset = B30h) [reset = 0h]**

RXHPCRB25 is shown in [Figure 16-258](#) and described in [Table 16-270](#).

**Figure 16-258. RXHPCRB25 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

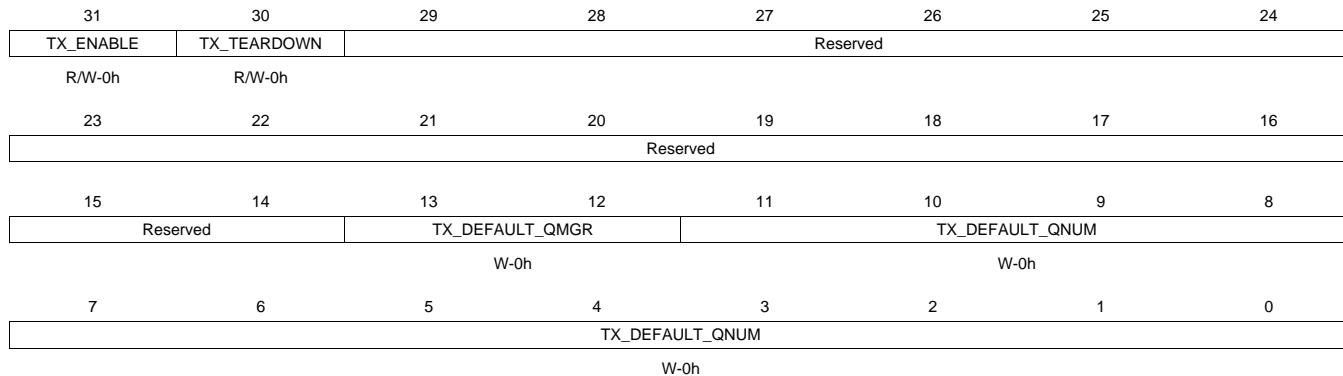
**Table 16-270. RXHPCRB25 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.108 TXGCR26 Register (offset = B40h) [reset = 0h]

TXGCR26 is shown in [Figure 16-259](#) and described in [Table 16-271](#).

**Figure 16-259. TXGCR26 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-271. TXGCR26 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

**16.5.5.109 RXGCR26 Register (offset = B48h) [reset = 0h]**

RXGCR26 is shown in [Figure 16-260](#) and described in [Table 16-272](#).

**Figure 16-260. RXGCR26 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-272. RXGCR26 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-272. RXGCR26 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.110 RXHPCRA26 Register (offset = B4Ch) [reset = 0h]

RXHPCRA26 is shown in [Figure 16-261](#) and described in [Table 16-273](#).

**Figure 16-261. RXHPCRA26 Register**

|          |    |                   |      |    |                   |      |    |
|----------|----|-------------------|------|----|-------------------|------|----|
| 31       | 30 | 29                | 28   | 27 | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ1_QMGR |      |    | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 23       | 22 | 21                | 20   | 19 | 18                | 17   | 16 |
|          |    | RX_HOST_FDQ1_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |
| 15       | 14 | 13                | 12   | 11 | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |      |    | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h |    |                   | W-0h |    |
| 7        | 6  | 5                 | 4    | 3  | 2                 | 1    | 0  |
|          |    | RX_HOST_FDQ0_QNUM |      |    |                   |      |    |
|          |    |                   | W-0h |    |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-273. RXHPCRA26 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.111 RXHPCRB26 Register (offset = B50h) [reset = 0h]

RXHPCRB26 is shown in [Figure 16-262](#) and described in [Table 16-274](#).

**Figure 16-262. RXHPCRB26 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

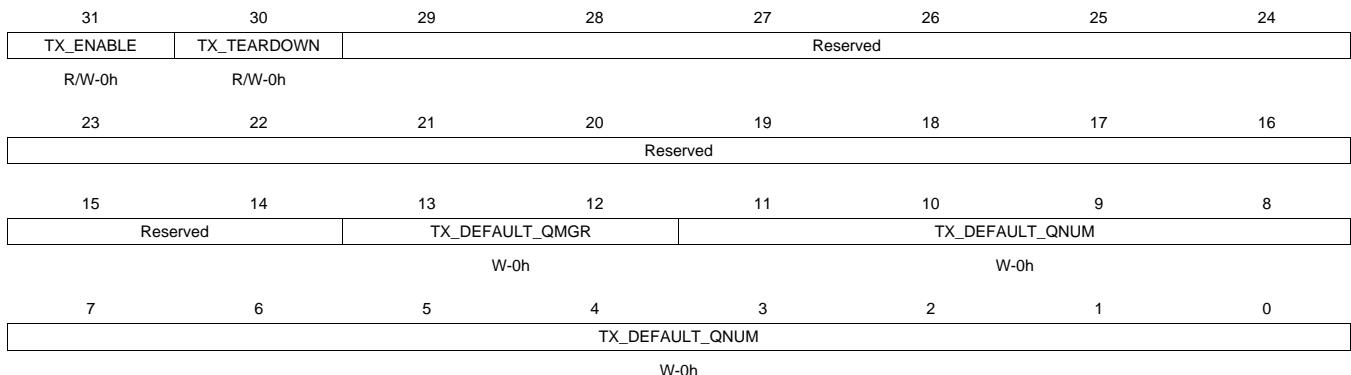
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-274. RXHPCRB26 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.112 TXGCR27 Register (offset = B60h) [reset = 0h]**

 TXGCR27 is shown in [Figure 16-263](#) and described in [Table 16-275](#).

**Figure 16-263. TXGCR27 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-275. TXGCR27 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.113 RXGCR27 Register (offset = B68h) [reset = 0h]

RXGCR27 is shown in [Figure 16-264](#) and described in [Table 16-276](#).

**Figure 16-264. RXGCR27 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-276. RXGCR27 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-276. RXGCR27 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.114 RXHPCRA27 Register (offset = B6Ch) [reset = 0h]

RXHPCRA27 is shown in [Figure 16-265](#) and described in [Table 16-277](#).

**Figure 16-265. RXHPCRA27 Register**

|          |    |                   |                   |      |                   |      |    |
|----------|----|-------------------|-------------------|------|-------------------|------|----|
| 31       | 30 | 29                | 28                | 27   | 26                | 25   | 24 |
| Reserved |    | RX_HOST_FDQ1_QMGR |                   |      | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 23       | 22 | 21                | 20                | 19   | 18                | 17   | 16 |
|          |    |                   | RX_HOST_FDQ1_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |
| 15       | 14 | 13                | 12                | 11   | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |                   |      | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 7        | 6  | 5                 | 4                 | 3    | 2                 | 1    | 0  |
|          |    |                   | RX_HOST_FDQ0_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-277. RXHPCRA27 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

**16.5.5.115 RXHPCRB27 Register (offset = B70h) [reset = 0h]**

RXHPCRB27 is shown in [Figure 16-266](#) and described in [Table 16-278](#).

**Figure 16-266. RXHPCRB27 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ3_QNUM |    |                   |    | W-0h |    |                   |    |
| 15                | 14 | 13                | 12 | 11   | 10 | 9                 | 8  |
| Reserved          |    | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ2_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

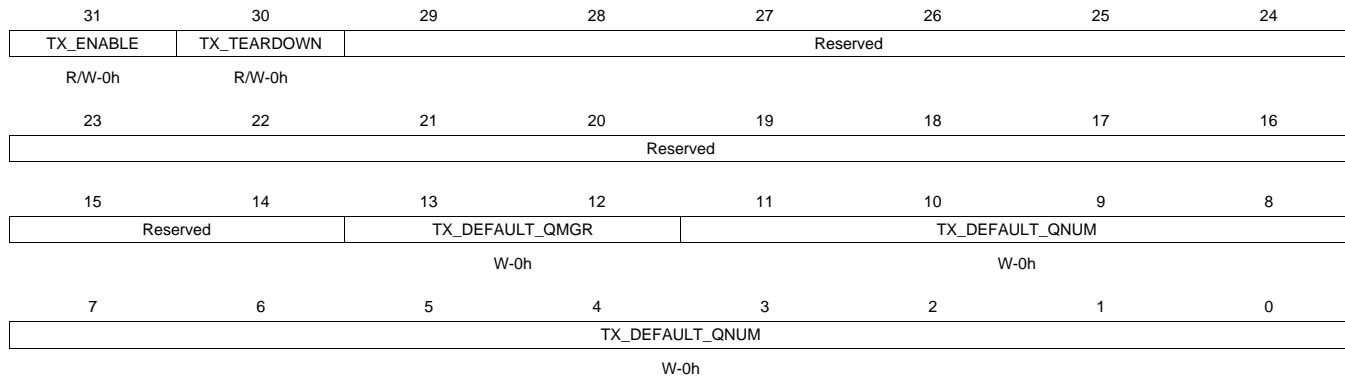
**Table 16-278. RXHPCRB27 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.5.116 TXGCR28 Register (offset = B80h) [reset = 0h]

TXGCR28 is shown in [Figure 16-267](#) and described in [Table 16-279](#).

**Figure 16-267. TXGCR28 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-279. TXGCR28 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.117 RXGCR28 Register (offset = B88h) [reset = 0h]

RXGCR28 is shown in [Figure 16-268](#) and described in [Table 16-280](#).

**Figure 16-268. RXGCR28 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
|                      |             |                    |    |          |                    |                   | W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-280. RXGCR28 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-280. RXGCR28 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

**16.5.5.118 RXHPCRA28 Register (offset = B8Ch) [reset = 0h]**

 RXHPCRA28 is shown in [Figure 16-269](#) and described in [Table 16-281](#).

**Figure 16-269. RXHPCRA28 Register**

| 31       | 30 | 29                | 28                | 27   | 26                | 25   | 24 |
|----------|----|-------------------|-------------------|------|-------------------|------|----|
| Reserved |    | RX_HOST_FDQ1_QMGR |                   |      | RX_HOST_FDQ1_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 23       | 22 | 21                | 20                | 19   | 18                | 17   | 16 |
|          |    |                   | RX_HOST_FDQ1_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |
| 15       | 14 | 13                | 12                | 11   | 10                | 9    | 8  |
| Reserved |    | RX_HOST_FDQ0_QMGR |                   |      | RX_HOST_FDQ0_QNUM |      |    |
|          |    |                   | W-0h              |      |                   | W-0h |    |
| 7        | 6  | 5                 | 4                 | 3    | 2                 | 1    | 0  |
|          |    |                   | RX_HOST_FDQ0_QNUM |      |                   |      |    |
|          |    |                   |                   | W-0h |                   |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-281. RXHPCRA28 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.119 RXHPCRB28 Register (offset = B90h) [reset = 0h]

RXHPCRB28 is shown in [Figure 16-270](#) and described in [Table 16-282](#).

**Figure 16-270. RXHPCRB28 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ3_QMGR |    |    |    | RX_HOST_FDQ3_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ3_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ2_QMGR |    |    |    | RX_HOST_FDQ2_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ2_QNUM |    |    |    | W-0h              |    |    |

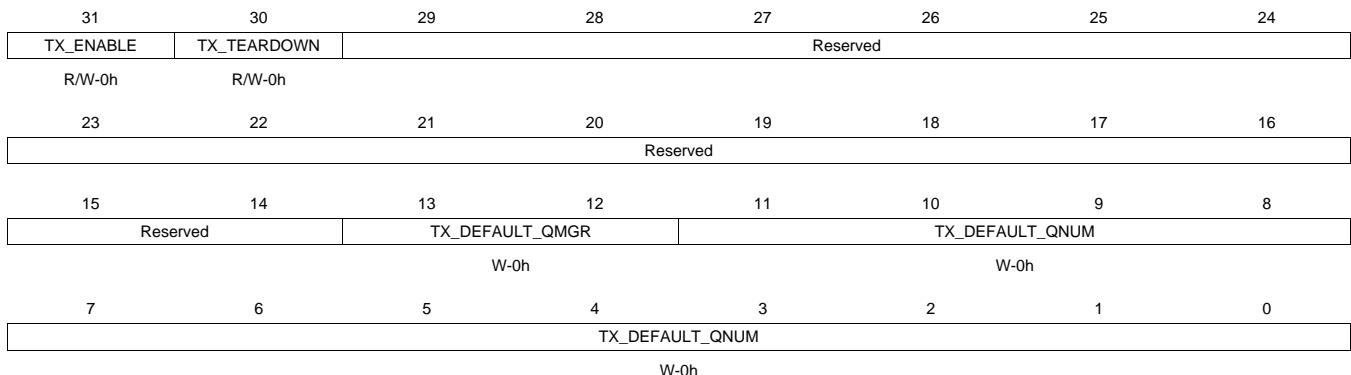
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-282. RXHPCRB28 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

**16.5.5.120 TXGCR29 Register (offset = BA0h) [reset = 0h]**

 TXGCR29 is shown in [Figure 16-271](#) and described in [Table 16-283](#).

**Figure 16-271. TXGCR29 Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-283. TXGCR29 Register Field Descriptions**

| Bit   | Field           | Type | Reset | Description   |
|-------|-----------------|------|-------|---|
| 31    | TX_ENABLE       | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | TX_TEARDOWN     | R/W  | 0h    | Setting this bit will request the channel to be torn down.<br>This field will remain set after a channel teardown is complete.  |
| 13-12 | TX_DEFAULT_QMGR | W    | 0h    | This field controls the default queue manager number that will be used to queue teardown descriptors back to the host.  |
| 11-0  | TX_DEFAULT_QNUM | W    | 0h    | This field controls the default queue number within the selected queue manager onto which teardown descriptors will be queued back to the host.<br>Table<br>98 -Tx Channel N Global Configuration Registers |

### 16.5.5.121 RXGCR29 Register (offset = BA8h) [reset = 0h]

RXGCR29 is shown in [Figure 16-272](#) and described in [Table 16-284](#).

**Figure 16-272. RXGCR29 Register**

| 31                   | 30          | 29                 | 28 | 27       | 26                 | 25                | 24   |
|----------------------|-------------|--------------------|----|----------|--------------------|-------------------|------|
| RX_ENABLE            | RX_TEARDOWN | RX_PAUSE           |    | Reserved |                    | RX_ERROR_HANDLING |      |
| R/W-0h               | R/W-0h      | R/W-0h             |    |          |                    |                   | W-0h |
| 23                   | 22          | 21                 | 20 | 19       | 18                 | 17                | 16   |
| RX_SOP_OFFSET        |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |
| 15                   | 14          | 13                 | 12 | 11       | 10                 | 9                 | 8    |
| RX_DEFAULT_DESC_TYPE |             | RX_DEFAULT_RQ_QMGR |    |          | RX_DEFAULT_RQ_QNUM |                   |      |
| W-0h                 |             | W-0h               |    |          | W-0h               |                   |      |
| 7                    | 6           | 5                  | 4  | 3        | 2                  | 1                 | 0    |
| RX_DEFAULT_RQ_QNUM   |             |                    |    |          |                    |                   |      |
| W-0h                 |             |                    |    |          |                    |                   |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-284. RXGCR29 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31    | RX_ENABLE         | R/W  | 0h    | This field enables or disables the channel<br>0 = channel is disabled<br>1 = channel is enabled This field will be cleared after a channel teardown is complete.  |
| 30    | RX_TEARDOWN       | R/W  | 0h    | This field indicates whether or not an Rx teardown operation is complete.<br>This field should be cleared when a channel is initialized.<br>This field will be set after a channel teardown is complete.  |
| 29    | RX_PAUSE          | R/W  | 0h    | Setting this bit causes the CPPI DMA to be suspended for rx channels.<br>If a pause is being requested and the channel is not in a packet then drop the credit.   |
| 24    | RX_ERROR_HANDLING | W    | 0h    | This bit controls the error handling mode for the channel and is only used when channel errors (i.e. descriptor or buffer starvation occurs):<br>0 = Starvation errors result in dropping packet and reclaiming any used descriptor or buffer resources back to the original queues/pools they were allocated to<br>1 = Starvation errors result in subsequent re-try of the descriptor allocation operation.<br>In this mode, the DMA will return to the IDLE state without saving its internal operational state back to the internal state RAM and without issuing an advance operation on the FIFO interface.<br>This results in the DMA re-initiating the FIFO block transfer at a later time with the intention that additional free buffers and/or descriptors will have been added.<br>Regardless of the value of this bit, the DMA will assert the cdma_rx_sof_overrun (for SOP) or cdma_rx_mof_overrun (for non-SOP) when |
| 23-16 | RX_SOP_OFFSET     | W    | 0h    | This field specifies the number of bytes that are to be skipped in the SOP buffer before beginning to write the payload.<br>This value must be less than the minimum size of a buffer in the system.<br>Valid values are 0 - 255 bytes.   |

**Table 16-284. RXGCR29 Register Field Descriptions (continued)**

| Bit   | Field                    | Type | Reset | Description  |
|-------|--------------------------|------|-------|--|
| 15-14 | RX_DEFAULT_DESC_TY<br>PE | W    | 0h    | This field indicates the default descriptor type to use:<br>0 = Reserved<br>1 = Host<br>2 = Reserved<br>3 = Reserved The actual descriptor type that will be used for reception can be overridden by information provided in the CPPI FIFO data block.                   |
| 13-12 | RX_DEFAULT_RQ_QMG<br>R   | W    | 0h    | This field indicates the default receive queue manager that this channel should use.<br>The actual receive queue manager index can be overridden by information provided in the CPPI FIFO data block.  |
| 11-0  | RX_DEFAULT_RQ_QNU<br>M   | W    | 0h    | This field indicates the default receive queue that this channel should use.<br>The actual receive queue that will be used for reception can be overridden by information provided in the CPPI FIFO data block.<br>Table 99 -Rx Channel N Global Configuration Registers |

### 16.5.5.122 RXHPCRA29 Register (offset = BACh) [reset = 0h]

RXHPCRA29 is shown in [Figure 16-273](#) and described in [Table 16-285](#).

**Figure 16-273. RXHPCRA29 Register**

|          |                   |    |    |    |                   |    |    |
|----------|-------------------|----|----|----|-------------------|----|----|
| 31       | 30                | 29 | 28 | 27 | 26                | 25 | 24 |
| Reserved | RX_HOST_FDQ1_QMGR |    |    |    | RX_HOST_FDQ1_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 23       | 22                | 21 | 20 | 19 | 18                | 17 | 16 |
|          | RX_HOST_FDQ1_QNUM |    |    |    | W-0h              |    |    |
| 15       | 14                | 13 | 12 | 11 | 10                | 9  | 8  |
| Reserved | RX_HOST_FDQ0_QMGR |    |    |    | RX_HOST_FDQ0_QNUM |    |    |
|          | W-0h              |    |    |    | W-0h              |    |    |
| 7        | 6                 | 5  | 4  | 3  | 2                 | 1  | 0  |
|          | RX_HOST_FDQ0_QNUM |    |    |    | W-0h              |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-285. RXHPCRA29 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ1_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 27-16 | RX_HOST_FDQ1_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 2nd Rx buffer in a host type packet   |
| 13-12 | RX_HOST_FDQ0_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the second Rx buffer in a host type packet.  |
| 11-0  | RX_HOST_FDQ0_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 1st Rx buffer in a host type packet Table 100 -Rx Channel N Host Packet Configuration Registers A |

### 16.5.5.123 RXHPCRB29 Register (offset = BB0h) [reset = 0h]

RXHPCRB29 is shown in [Figure 16-274](#) and described in [Table 16-286](#).

**Figure 16-274. RXHPCRB29 Register**

|                   |    |                   |    |      |    |                   |    |
|-------------------|----|-------------------|----|------|----|-------------------|----|
| 31                | 30 | 29                | 28 | 27   | 26 | 25                | 24 |
| Reserved          |    | RX_HOST_FDQ3_QMGR |    |      |    | RX_HOST_FDQ3_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 23                | 22 | 21                | 20 | 19   | 18 | 17                | 16 |
| RX_HOST_FDQ3_QNUM |    |                   |    | W-0h |    |                   |    |
| Reserved          |    | RX_HOST_FDQ2_QMGR |    |      |    | RX_HOST_FDQ2_QNUM |    |
| W-0h              |    |                   |    | W-0h |    |                   |    |
| 7                 | 6  | 5                 | 4  | 3    | 2  | 1                 | 0  |
| RX_HOST_FDQ2_QNUM |    |                   |    | W-0h |    |                   |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-286. RXHPCRB29 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 29-28 | RX_HOST_FDQ3_QMGR | W    | 0h    | This field specifies which Manager should be used for the 4th or later Rx buffers in a host type packet   |
| 27-16 | RX_HOST_FDQ3_QNUM | W    | 0h    | This field specifies which Free Descriptor Queue should be used for the 4th or later Rx buffers in a host type packet   |
| 13-12 | RX_HOST_FDQ2_QMGR | W    | 0h    | This field specifies which Buffer Manager should be used for the 3rd Rx buffer in a host type packet  |
| 11-0  | RX_HOST_FDQ2_QNUM | W    | 0h    | This field specifies which Free Descriptor / Buffer Pool should be used for the 3rd Rx buffer in a host type packet Table 101 -Rx Channel N Host Packet Configuration Registers B |

### 16.5.6 CPPI\_DMA\_SCHEDULER Registers

[Table 16-287](#) lists the memory-mapped registers for the CPPI\_DMA\_SCHEDULER. All register offset addresses not listed in [Table 16-287](#) should be considered as reserved locations and the register contents should not be modified.

**Table 16-287. CPPI\_DMA\_SCHEDULER REGISTERS**

| Offset       | Acronym         | Register Name | Section                          |
|--------------|-----------------|---------------|----------------------------------|
| 0h           | DMA_SCHED_CTRL  |               | <a href="#">Section 16.5.6.1</a> |
| 800h to 900h | WORD0 to WORD63 |               | <a href="#">Section 16.5.6.2</a> |

**16.5.6.1 DMA\_SCHED\_CTRL Register (offset = 0h) [reset = 0h]**

DMA\_SCHED\_CTRL is shown in [Figure 16-275](#) and described in [Table 16-288](#).

**Figure 16-275. DMA\_SCHED\_CTRL Register**

| 31     | 30 | 29 | 28 | 27         | 26 | 25 | 24 |
|--------|----|----|----|------------|----|----|----|
| ENABLE |    |    |    | Reserved   |    |    |    |
| R/W-0h |    |    |    |            |    |    |    |
| 23     | 22 | 21 | 20 | 19         | 18 | 17 | 16 |
|        |    |    |    | Reserved   |    |    |    |
| 15     | 14 | 13 | 12 | 11         | 10 | 9  | 8  |
|        |    |    |    | Reserved   |    |    |    |
| 7      | 6  | 5  | 4  | 3          | 2  | 1  | 0  |
|        |    |    |    | LAST_ENTRY |    |    |    |
| R/W-0h |    |    |    |            |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-288. DMA\_SCHED\_CTRL Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 31  | ENABLE     | R/W  | 0h    | This is the enable bit for the scheduler and is encoded as follows:<br>0 = scheduler is disabled and will no longer fetch entries from the scheduler table or pass credits to the DMA controller<br>1 = scheduler is enabled This bit should only be set after the table has been initialized.   |
| 7-0 | LAST_ENTRY | R/W  | 0h    | This field indicates the last valid entry in the scheduler table.<br>There are 64 words in the table and there are 4 entries in each word.<br>The table can be programmed with any integer number of entries from 1 to 256.<br>The corresponding encoding for this field is as follows:<br>0 = 1 entry<br>1 = 2 entries ...<br>254 = 255 entries<br>255 = 256 entries CPPI DMA Scheduler Control Registers |

### 16.5.6.2 WORDx Register (offset = 800h to 900h) [reset = 0h]

WORD0 to WORD63 is shown in [Figure 16-276](#) and described in [Table 16-289](#).

**Figure 16-276. WORD0 to WORD63 Register**

| 31          | 30       | 29 | 28 | 27 | 26             | 25 | 24 |
|-------------|----------|----|----|----|----------------|----|----|
| ENTRY3_RXTX | Reserved |    |    |    | ENTRY3_CHANNEL |    |    |
| W-0h        |          |    |    |    | R/W-           |    |    |
| 23          | 22       | 21 | 20 | 19 | 18             | 17 | 16 |
| ENTRY2_RXTX | Reserved |    |    |    | ENTRY2_CHANNEL |    |    |
| W-0h        |          |    |    |    | R/W-           |    |    |
| 15          | 14       | 13 | 12 | 11 | 10             | 9  | 8  |
| ENTRY1_RXTX | Reserved |    |    |    | ENTRY1_CHANNEL |    |    |
| W-0h        |          |    |    |    | R/W-           |    |    |
| 7           | 6        | 5  | 4  | 3  | 2              | 1  | 0  |
| ENTRY0_RXTX | Reserved |    |    |    | ENTRY0_CHANNEL |    |    |
| W-0h        |          |    |    |    | R/W-           |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-289. WORD0 to WORD63 Register Field Descriptions**

| Bit   | Field          | Type | Reset | Description  |
|-------|----------------|------|-------|--|
| 31    | ENTRY3_RXTX    | W    | 0h    | This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:<br>0 = Tx Channel<br>1 = Rx Channel   |
| 28-24 | ENTRY3_CHANNEL | R/W  |       | This field indicates the channel number that is to be given an opportunity to transfer data.<br>If this is a Tx entry, the DMA will be presented with a scheduling credit for that exact Tx channel.<br>If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel.<br>For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. |
| 23    | ENTRY2_RXTX    | W    | 0h    | This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:<br>0 = Tx Channel<br>1 = Rx Channel   |
| 20-16 | ENTRY2_CHANNEL | R/W  |       | This field indicates the channel number that is to be given an opportunity to transfer data.<br>If this is a Tx entry, the DMA will be presented with a scheduling credit for that exact Tx channel.<br>If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel.<br>For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. |
| 15    | ENTRY1_RXTX    | W    | 0h    | This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:<br>0 = Tx Channel<br>1 = Rx Channel   |

**Table 16-289. WORD0 to WORD63 Register Field Descriptions (continued)**

| Bit  | Field          | Type | Reset | Description  |
|------|----------------|------|-------|--|
| 12-8 | ENTRY1_CHANNEL | R/W  |       | This field indicates the channel number that is to be given an opportunity to transfer data.<br>If this is a Tx entry, the DMA will be presented with a scheduling credit for that exact Tx channel.<br>If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel.<br>For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. |
| 7    | ENTRY0_RXTX    | W    | 0h    | This bit indicates if this entry is for a Tx or an Rx channel and is encoded as follows:<br>0 = Tx Channel<br>1 = Rx Channel   |
| 4-0  | ENTRY0_CHANNEL | R/W  |       | This field indicates the channel number that is to be given an opportunity to transfer data.<br>If this is a Tx entry, the DMA will be presented with a scheduling credit for that exact Tx channel.<br>If this is an Rx entry, the DMA will be presented with a scheduling credit for the Rx FIFO that is associated with this channel.<br>For Rx FIFOs which carry traffic for more than 1 Rx DMA channel, the exact channel number that is given in the Rx credit will actually be the channel number which is currently on the head element of that Rx FIFO, which is not necessarily the channel number given in the scheduler table entry. |

### 16.5.7 QUEUE\_MGR Registers

Table 16-290 lists the memory-mapped registers for the QUEUE\_MGR. All register offset addresses not listed in Table 16-290 should be considered as reserved locations and the register contents should not be modified.

**Table 16-290. QUEUE\_MGR REGISTERS**

| Offset | Acronym    | Register Name | Section           |
|--------|------------|---------------|-------------------|
| 0h     | QMGRREVID  |               | Section 16.5.7.1  |
| 8h     | QMGRRST    |               | Section 16.5.7.2  |
| 20h    | FDBSC0     |               | Section 16.5.7.3  |
| 24h    | FDBSC1     |               | Section 16.5.7.4  |
| 28h    | FDBSC2     |               | Section 16.5.7.5  |
| 2Ch    | FDBSC3     |               | Section 16.5.7.6  |
| 30h    | FDBSC4     |               | Section 16.5.7.7  |
| 34h    | FDBSC5     |               | Section 16.5.7.8  |
| 38h    | FDBSC6     |               | Section 16.5.7.9  |
| 3Ch    | FDBSC7     |               | Section 16.5.7.10 |
| 80h    | LRAM0BASE  |               | Section 16.5.7.11 |
| 84h    | LRAM0SIZE  |               | Section 16.5.7.12 |
| 88h    | LRAM1BASE  |               | Section 16.5.7.13 |
| 90h    | PEND0      |               | Section 16.5.7.14 |
| 94h    | PEND1      |               | Section 16.5.7.15 |
| 98h    | PEND2      |               | Section 16.5.7.16 |
| 9Ch    | PEND3      |               | Section 16.5.7.17 |
| A0h    | PEND4      |               | Section 16.5.7.18 |
| 1000h  | QMEMRBASE0 |               | Section 16.5.7.19 |
| 1004h  | QMEMCTRL0  |               | Section 16.5.7.20 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section           |
|--------|------------|---------------|-------------------|
| 1010h  | QMEMRBASE1 |               | Section 16.5.7.21 |
| 1014h  | QMEMCTRL1  |               | Section 16.5.7.22 |
| 1020h  | QMEMRBASE2 |               | Section 16.5.7.23 |
| 1024h  | QMEMCTRL2  |               | Section 16.5.7.24 |
| 1030h  | QMEMRBASE3 |               | Section 16.5.7.25 |
| 1034h  | QMEMCTRL3  |               | Section 16.5.7.26 |
| 1040h  | QMEMRBASE4 |               | Section 16.5.7.27 |
| 1044h  | QMEMCTRL4  |               | Section 16.5.7.28 |
| 1050h  | QMEMRBASE5 |               | Section 16.5.7.29 |
| 1054h  | QMEMCTRL5  |               | Section 16.5.7.30 |
| 1060h  | QMEMRBASE6 |               | Section 16.5.7.31 |
| 1064h  | QMEMCTRL6  |               | Section 16.5.7.32 |
| 1070h  | QMEMRBASE7 |               | Section 16.5.7.33 |
| 1074h  | QMEMCTRL7  |               | Section 16.5.7.34 |
| 2000h  | QUEUE_0_A  |               | Section 16.5.7.35 |
| 2004h  | QUEUE_0_B  |               | Section 16.5.7.36 |
| 2008h  | QUEUE_0_C  |               | Section 16.5.7.37 |
| 200Ch  | QUEUE_0_D  |               | Section 16.5.7.38 |
| 2010h  | QUEUE_1_A  |               | Section 16.5.7.39 |
| 2014h  | QUEUE_1_B  |               | Section 16.5.7.40 |
| 2018h  | QUEUE_1_C  |               | Section 16.5.7.41 |
| 201Ch  | QUEUE_1_D  |               | Section 16.5.7.42 |
| 2020h  | QUEUE_2_A  |               | Section 16.5.7.43 |
| 2024h  | QUEUE_2_B  |               | Section 16.5.7.44 |
| 2028h  | QUEUE_2_C  |               | Section 16.5.7.45 |
| 202Ch  | QUEUE_2_D  |               | Section 16.5.7.46 |
| 2030h  | QUEUE_3_A  |               | Section 16.5.7.47 |
| 2034h  | QUEUE_3_B  |               | Section 16.5.7.48 |
| 2038h  | QUEUE_3_C  |               | Section 16.5.7.49 |
| 203Ch  | QUEUE_3_D  |               | Section 16.5.7.50 |
| 2040h  | QUEUE_4_A  |               | Section 16.5.7.51 |
| 2044h  | QUEUE_4_B  |               | Section 16.5.7.52 |
| 2048h  | QUEUE_4_C  |               | Section 16.5.7.53 |
| 204Ch  | QUEUE_4_D  |               | Section 16.5.7.54 |
| 2050h  | QUEUE_5_A  |               | Section 16.5.7.55 |
| 2054h  | QUEUE_5_B  |               | Section 16.5.7.56 |
| 2058h  | QUEUE_5_C  |               | Section 16.5.7.57 |
| 205Ch  | QUEUE_5_D  |               | Section 16.5.7.58 |
| 2060h  | QUEUE_6_A  |               | Section 16.5.7.59 |
| 2064h  | QUEUE_6_B  |               | Section 16.5.7.60 |
| 2068h  | QUEUE_6_C  |               | Section 16.5.7.61 |
| 206Ch  | QUEUE_6_D  |               | Section 16.5.7.62 |
| 2070h  | QUEUE_7_A  |               | Section 16.5.7.63 |
| 2074h  | QUEUE_7_B  |               | Section 16.5.7.64 |
| 2078h  | QUEUE_7_C  |               | Section 16.5.7.65 |
| 207Ch  | QUEUE_7_D  |               | Section 16.5.7.66 |
| 2080h  | QUEUE_8_A  |               | Section 16.5.7.67 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 2084h  | QUEUE_8_B  |               | Section 16.5.7.68  |
| 2088h  | QUEUE_8_C  |               | Section 16.5.7.69  |
| 208Ch  | QUEUE_8_D  |               | Section 16.5.7.70  |
| 2090h  | QUEUE_9_A  |               | Section 16.5.7.71  |
| 2094h  | QUEUE_9_B  |               | Section 16.5.7.72  |
| 2098h  | QUEUE_9_C  |               | Section 16.5.7.73  |
| 209Ch  | QUEUE_9_D  |               | Section 16.5.7.74  |
| 20A0h  | QUEUE_10_A |               | Section 16.5.7.75  |
| 20A4h  | QUEUE_10_B |               | Section 16.5.7.76  |
| 20A8h  | QUEUE_10_C |               | Section 16.5.7.77  |
| 20ACh  | QUEUE_10_D |               | Section 16.5.7.78  |
| 20B0h  | QUEUE_11_A |               | Section 16.5.7.79  |
| 20B4h  | QUEUE_11_B |               | Section 16.5.7.80  |
| 20B8h  | QUEUE_11_C |               | Section 16.5.7.81  |
| 20BCh  | QUEUE_11_D |               | Section 16.5.7.82  |
| 20C0h  | QUEUE_12_A |               | Section 16.5.7.83  |
| 20C4h  | QUEUE_12_B |               | Section 16.5.7.84  |
| 20C8h  | QUEUE_12_C |               | Section 16.5.7.85  |
| 20CCh  | QUEUE_12_D |               | Section 16.5.7.86  |
| 20D0h  | QUEUE_13_A |               | Section 16.5.7.87  |
| 20D4h  | QUEUE_13_B |               | Section 16.5.7.88  |
| 20D8h  | QUEUE_13_C |               | Section 16.5.7.89  |
| 20DCh  | QUEUE_13_D |               | Section 16.5.7.90  |
| 20E0h  | QUEUE_14_A |               | Section 16.5.7.91  |
| 20E4h  | QUEUE_14_B |               | Section 16.5.7.92  |
| 20E8h  | QUEUE_14_C |               | Section 16.5.7.93  |
| 20ECh  | QUEUE_14_D |               | Section 16.5.7.94  |
| 20F0h  | QUEUE_15_A |               | Section 16.5.7.95  |
| 20F4h  | QUEUE_15_B |               | Section 16.5.7.96  |
| 20F8h  | QUEUE_15_C |               | Section 16.5.7.97  |
| 20FCh  | QUEUE_15_D |               | Section 16.5.7.98  |
| 2100h  | QUEUE_16_A |               | Section 16.5.7.99  |
| 2104h  | QUEUE_16_B |               | Section 16.5.7.100 |
| 2108h  | QUEUE_16_C |               | Section 16.5.7.101 |
| 210Ch  | QUEUE_16_D |               | Section 16.5.7.102 |
| 2110h  | QUEUE_17_A |               | Section 16.5.7.103 |
| 2114h  | QUEUE_17_B |               | Section 16.5.7.104 |
| 2118h  | QUEUE_17_C |               | Section 16.5.7.105 |
| 211Ch  | QUEUE_17_D |               | Section 16.5.7.106 |
| 2120h  | QUEUE_18_A |               | Section 16.5.7.107 |
| 2124h  | QUEUE_18_B |               | Section 16.5.7.108 |
| 2128h  | QUEUE_18_C |               | Section 16.5.7.109 |
| 212Ch  | QUEUE_18_D |               | Section 16.5.7.110 |
| 2130h  | QUEUE_19_A |               | Section 16.5.7.111 |
| 2134h  | QUEUE_19_B |               | Section 16.5.7.112 |
| 2138h  | QUEUE_19_C |               | Section 16.5.7.113 |
| 213Ch  | QUEUE_19_D |               | Section 16.5.7.114 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 2140h  | QUEUE_20_A |               | Section 16.5.7.115 |
| 2144h  | QUEUE_20_B |               | Section 16.5.7.116 |
| 2148h  | QUEUE_20_C |               | Section 16.5.7.117 |
| 214Ch  | QUEUE_20_D |               | Section 16.5.7.118 |
| 2150h  | QUEUE_21_A |               | Section 16.5.7.119 |
| 2154h  | QUEUE_21_B |               | Section 16.5.7.120 |
| 2158h  | QUEUE_21_C |               | Section 16.5.7.121 |
| 215Ch  | QUEUE_21_D |               | Section 16.5.7.122 |
| 2160h  | QUEUE_22_A |               | Section 16.5.7.123 |
| 2164h  | QUEUE_22_B |               | Section 16.5.7.124 |
| 2168h  | QUEUE_22_C |               | Section 16.5.7.125 |
| 216Ch  | QUEUE_22_D |               | Section 16.5.7.126 |
| 2170h  | QUEUE_23_A |               | Section 16.5.7.127 |
| 2174h  | QUEUE_23_B |               | Section 16.5.7.128 |
| 2178h  | QUEUE_23_C |               | Section 16.5.7.129 |
| 217Ch  | QUEUE_23_D |               | Section 16.5.7.130 |
| 2180h  | QUEUE_24_A |               | Section 16.5.7.131 |
| 2184h  | QUEUE_24_B |               | Section 16.5.7.132 |
| 2188h  | QUEUE_24_C |               | Section 16.5.7.133 |
| 218Ch  | QUEUE_24_D |               | Section 16.5.7.134 |
| 2190h  | QUEUE_25_A |               | Section 16.5.7.135 |
| 2194h  | QUEUE_25_B |               | Section 16.5.7.136 |
| 2198h  | QUEUE_25_C |               | Section 16.5.7.137 |
| 219Ch  | QUEUE_25_D |               | Section 16.5.7.138 |
| 21A0h  | QUEUE_26_A |               | Section 16.5.7.139 |
| 21A4h  | QUEUE_26_B |               | Section 16.5.7.140 |
| 21A8h  | QUEUE_26_C |               | Section 16.5.7.141 |
| 21ACh  | QUEUE_26_D |               | Section 16.5.7.142 |
| 21B0h  | QUEUE_27_A |               | Section 16.5.7.143 |
| 21B4h  | QUEUE_27_B |               | Section 16.5.7.144 |
| 21B8h  | QUEUE_27_C |               | Section 16.5.7.145 |
| 21BCh  | QUEUE_27_D |               | Section 16.5.7.146 |
| 21C0h  | QUEUE_28_A |               | Section 16.5.7.147 |
| 21C4h  | QUEUE_28_B |               | Section 16.5.7.148 |
| 21C8h  | QUEUE_28_C |               | Section 16.5.7.149 |
| 21CCh  | QUEUE_28_D |               | Section 16.5.7.150 |
| 21D0h  | QUEUE_29_A |               | Section 16.5.7.151 |
| 21D4h  | QUEUE_29_B |               | Section 16.5.7.152 |
| 21D8h  | QUEUE_29_C |               | Section 16.5.7.153 |
| 21DCh  | QUEUE_29_D |               | Section 16.5.7.154 |
| 21E0h  | QUEUE_30_A |               | Section 16.5.7.155 |
| 21E4h  | QUEUE_30_B |               | Section 16.5.7.156 |
| 21E8h  | QUEUE_30_C |               | Section 16.5.7.157 |
| 21ECh  | QUEUE_30_D |               | Section 16.5.7.158 |
| 21F0h  | QUEUE_31_A |               | Section 16.5.7.159 |
| 21F4h  | QUEUE_31_B |               | Section 16.5.7.160 |
| 21F8h  | QUEUE_31_C |               | Section 16.5.7.161 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b> | <b>Section</b>     |
|---------------|----------------|----------------------|--------------------|
| 21FCh         | QUEUE_31_D     |                      | Section 16.5.7.162 |
| 2200h         | QUEUE_32_A     |                      | Section 16.5.7.163 |
| 2204h         | QUEUE_32_B     |                      | Section 16.5.7.164 |
| 2208h         | QUEUE_32_C     |                      | Section 16.5.7.165 |
| 220Ch         | QUEUE_32_D     |                      | Section 16.5.7.166 |
| 2210h         | QUEUE_33_A     |                      | Section 16.5.7.167 |
| 2214h         | QUEUE_33_B     |                      | Section 16.5.7.168 |
| 2218h         | QUEUE_33_C     |                      | Section 16.5.7.169 |
| 221Ch         | QUEUE_33_D     |                      | Section 16.5.7.170 |
| 2220h         | QUEUE_34_A     |                      | Section 16.5.7.171 |
| 2224h         | QUEUE_34_B     |                      | Section 16.5.7.172 |
| 2228h         | QUEUE_34_C     |                      | Section 16.5.7.173 |
| 222Ch         | QUEUE_34_D     |                      | Section 16.5.7.174 |
| 2230h         | QUEUE_35_A     |                      | Section 16.5.7.175 |
| 2234h         | QUEUE_35_B     |                      | Section 16.5.7.176 |
| 2238h         | QUEUE_35_C     |                      | Section 16.5.7.177 |
| 223Ch         | QUEUE_35_D     |                      | Section 16.5.7.178 |
| 2240h         | QUEUE_36_A     |                      | Section 16.5.7.179 |
| 2244h         | QUEUE_36_B     |                      | Section 16.5.7.180 |
| 2248h         | QUEUE_36_C     |                      | Section 16.5.7.181 |
| 224Ch         | QUEUE_36_D     |                      | Section 16.5.7.182 |
| 2250h         | QUEUE_37_A     |                      | Section 16.5.7.183 |
| 2254h         | QUEUE_37_B     |                      | Section 16.5.7.184 |
| 2258h         | QUEUE_37_C     |                      | Section 16.5.7.185 |
| 225Ch         | QUEUE_37_D     |                      | Section 16.5.7.186 |
| 2260h         | QUEUE_38_A     |                      | Section 16.5.7.187 |
| 2264h         | QUEUE_38_B     |                      | Section 16.5.7.188 |
| 2268h         | QUEUE_38_C     |                      | Section 16.5.7.189 |
| 226Ch         | QUEUE_38_D     |                      | Section 16.5.7.190 |
| 2270h         | QUEUE_39_A     |                      | Section 16.5.7.191 |
| 2274h         | QUEUE_39_B     |                      | Section 16.5.7.192 |
| 2278h         | QUEUE_39_C     |                      | Section 16.5.7.193 |
| 227Ch         | QUEUE_39_D     |                      | Section 16.5.7.194 |
| 2280h         | QUEUE_40_A     |                      | Section 16.5.7.195 |
| 2284h         | QUEUE_40_B     |                      | Section 16.5.7.196 |
| 2288h         | QUEUE_40_C     |                      | Section 16.5.7.197 |
| 228Ch         | QUEUE_40_D     |                      | Section 16.5.7.198 |
| 2290h         | QUEUE_41_A     |                      | Section 16.5.7.199 |
| 2294h         | QUEUE_41_B     |                      | Section 16.5.7.200 |
| 2298h         | QUEUE_41_C     |                      | Section 16.5.7.201 |
| 229Ch         | QUEUE_41_D     |                      | Section 16.5.7.202 |
| 22A0h         | QUEUE_42_A     |                      | Section 16.5.7.203 |
| 22A4h         | QUEUE_42_B     |                      | Section 16.5.7.204 |
| 22A8h         | QUEUE_42_C     |                      | Section 16.5.7.205 |
| 22ACh         | QUEUE_42_D     |                      | Section 16.5.7.206 |
| 22B0h         | QUEUE_43_A     |                      | Section 16.5.7.207 |
| 22B4h         | QUEUE_43_B     |                      | Section 16.5.7.208 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 22B8h  | QUEUE_43_C |               | Section 16.5.7.209 |
| 22BCh  | QUEUE_43_D |               | Section 16.5.7.210 |
| 22C0h  | QUEUE_44_A |               | Section 16.5.7.211 |
| 22C4h  | QUEUE_44_B |               | Section 16.5.7.212 |
| 22C8h  | QUEUE_44_C |               | Section 16.5.7.213 |
| 22CCh  | QUEUE_44_D |               | Section 16.5.7.214 |
| 22D0h  | QUEUE_45_A |               | Section 16.5.7.215 |
| 22D4h  | QUEUE_45_B |               | Section 16.5.7.216 |
| 22D8h  | QUEUE_45_C |               | Section 16.5.7.217 |
| 22DCh  | QUEUE_45_D |               | Section 16.5.7.218 |
| 22E0h  | QUEUE_46_A |               | Section 16.5.7.219 |
| 22E4h  | QUEUE_46_B |               | Section 16.5.7.220 |
| 22E8h  | QUEUE_46_C |               | Section 16.5.7.221 |
| 22ECh  | QUEUE_46_D |               | Section 16.5.7.222 |
| 22F0h  | QUEUE_47_A |               | Section 16.5.7.223 |
| 22F4h  | QUEUE_47_B |               | Section 16.5.7.224 |
| 22F8h  | QUEUE_47_C |               | Section 16.5.7.225 |
| 22FCh  | QUEUE_47_D |               | Section 16.5.7.226 |
| 2300h  | QUEUE_48_A |               | Section 16.5.7.227 |
| 2304h  | QUEUE_48_B |               | Section 16.5.7.228 |
| 2308h  | QUEUE_48_C |               | Section 16.5.7.229 |
| 230Ch  | QUEUE_48_D |               | Section 16.5.7.230 |
| 2310h  | QUEUE_49_A |               | Section 16.5.7.231 |
| 2314h  | QUEUE_49_B |               | Section 16.5.7.232 |
| 2318h  | QUEUE_49_C |               | Section 16.5.7.233 |
| 231Ch  | QUEUE_49_D |               | Section 16.5.7.234 |
| 2320h  | QUEUE_50_A |               | Section 16.5.7.235 |
| 2324h  | QUEUE_50_B |               | Section 16.5.7.236 |
| 2328h  | QUEUE_50_C |               | Section 16.5.7.237 |
| 232Ch  | QUEUE_50_D |               | Section 16.5.7.238 |
| 2330h  | QUEUE_51_A |               | Section 16.5.7.239 |
| 2334h  | QUEUE_51_B |               | Section 16.5.7.240 |
| 2338h  | QUEUE_51_C |               | Section 16.5.7.241 |
| 233Ch  | QUEUE_51_D |               | Section 16.5.7.242 |
| 2340h  | QUEUE_52_A |               | Section 16.5.7.243 |
| 2344h  | QUEUE_52_B |               | Section 16.5.7.244 |
| 2348h  | QUEUE_52_C |               | Section 16.5.7.245 |
| 234Ch  | QUEUE_52_D |               | Section 16.5.7.246 |
| 2350h  | QUEUE_53_A |               | Section 16.5.7.247 |
| 2354h  | QUEUE_53_B |               | Section 16.5.7.248 |
| 2358h  | QUEUE_53_C |               | Section 16.5.7.249 |
| 235Ch  | QUEUE_53_D |               | Section 16.5.7.250 |
| 2360h  | QUEUE_54_A |               | Section 16.5.7.251 |
| 2364h  | QUEUE_54_B |               | Section 16.5.7.252 |
| 2368h  | QUEUE_54_C |               | Section 16.5.7.253 |
| 236Ch  | QUEUE_54_D |               | Section 16.5.7.254 |
| 2370h  | QUEUE_55_A |               | Section 16.5.7.255 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 2374h  | QUEUE_55_B |               | Section 16.5.7.256 |
| 2378h  | QUEUE_55_C |               | Section 16.5.7.257 |
| 237Ch  | QUEUE_55_D |               | Section 16.5.7.258 |
| 2380h  | QUEUE_56_A |               | Section 16.5.7.259 |
| 2384h  | QUEUE_56_B |               | Section 16.5.7.260 |
| 2388h  | QUEUE_56_C |               | Section 16.5.7.261 |
| 238Ch  | QUEUE_56_D |               | Section 16.5.7.262 |
| 2390h  | QUEUE_57_A |               | Section 16.5.7.263 |
| 2394h  | QUEUE_57_B |               | Section 16.5.7.264 |
| 2398h  | QUEUE_57_C |               | Section 16.5.7.265 |
| 239Ch  | QUEUE_57_D |               | Section 16.5.7.266 |
| 23A0h  | QUEUE_58_A |               | Section 16.5.7.267 |
| 23A4h  | QUEUE_58_B |               | Section 16.5.7.268 |
| 23A8h  | QUEUE_58_C |               | Section 16.5.7.269 |
| 23ACh  | QUEUE_58_D |               | Section 16.5.7.270 |
| 23B0h  | QUEUE_59_A |               | Section 16.5.7.271 |
| 23B4h  | QUEUE_59_B |               | Section 16.5.7.272 |
| 23B8h  | QUEUE_59_C |               | Section 16.5.7.273 |
| 23BCh  | QUEUE_59_D |               | Section 16.5.7.274 |
| 23C0h  | QUEUE_60_A |               | Section 16.5.7.275 |
| 23C4h  | QUEUE_60_B |               | Section 16.5.7.276 |
| 23C8h  | QUEUE_60_C |               | Section 16.5.7.277 |
| 23CCh  | QUEUE_60_D |               | Section 16.5.7.278 |
| 23D0h  | QUEUE_61_A |               | Section 16.5.7.279 |
| 23D4h  | QUEUE_61_B |               | Section 16.5.7.280 |
| 23D8h  | QUEUE_61_C |               | Section 16.5.7.281 |
| 23DCh  | QUEUE_61_D |               | Section 16.5.7.282 |
| 23E0h  | QUEUE_62_A |               | Section 16.5.7.283 |
| 23E4h  | QUEUE_62_B |               | Section 16.5.7.284 |
| 23E8h  | QUEUE_62_C |               | Section 16.5.7.285 |
| 23ECh  | QUEUE_62_D |               | Section 16.5.7.286 |
| 23F0h  | QUEUE_63_A |               | Section 16.5.7.287 |
| 23F4h  | QUEUE_63_B |               | Section 16.5.7.288 |
| 23F8h  | QUEUE_63_C |               | Section 16.5.7.289 |
| 23FCh  | QUEUE_63_D |               | Section 16.5.7.290 |
| 2400h  | QUEUE_64_A |               | Section 16.5.7.291 |
| 2404h  | QUEUE_64_B |               | Section 16.5.7.292 |
| 2408h  | QUEUE_64_C |               | Section 16.5.7.293 |
| 240Ch  | QUEUE_64_D |               | Section 16.5.7.294 |
| 2410h  | QUEUE_65_A |               | Section 16.5.7.295 |
| 2414h  | QUEUE_65_B |               | Section 16.5.7.296 |
| 2418h  | QUEUE_65_C |               | Section 16.5.7.297 |
| 241Ch  | QUEUE_65_D |               | Section 16.5.7.298 |
| 2420h  | QUEUE_66_A |               | Section 16.5.7.299 |
| 2424h  | QUEUE_66_B |               | Section 16.5.7.300 |
| 2428h  | QUEUE_66_C |               | Section 16.5.7.301 |
| 242Ch  | QUEUE_66_D |               | Section 16.5.7.302 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 2430h  | QUEUE_67_A |               | Section 16.5.7.303 |
| 2434h  | QUEUE_67_B |               | Section 16.5.7.304 |
| 2438h  | QUEUE_67_C |               | Section 16.5.7.305 |
| 243Ch  | QUEUE_67_D |               | Section 16.5.7.306 |
| 2440h  | QUEUE_68_A |               | Section 16.5.7.307 |
| 2444h  | QUEUE_68_B |               | Section 16.5.7.308 |
| 2448h  | QUEUE_68_C |               | Section 16.5.7.309 |
| 244Ch  | QUEUE_68_D |               | Section 16.5.7.310 |
| 2450h  | QUEUE_69_A |               | Section 16.5.7.311 |
| 2454h  | QUEUE_69_B |               | Section 16.5.7.312 |
| 2458h  | QUEUE_69_C |               | Section 16.5.7.313 |
| 245Ch  | QUEUE_69_D |               | Section 16.5.7.314 |
| 2460h  | QUEUE_70_A |               | Section 16.5.7.315 |
| 2464h  | QUEUE_70_B |               | Section 16.5.7.316 |
| 2468h  | QUEUE_70_C |               | Section 16.5.7.317 |
| 246Ch  | QUEUE_70_D |               | Section 16.5.7.318 |
| 2470h  | QUEUE_71_A |               | Section 16.5.7.319 |
| 2474h  | QUEUE_71_B |               | Section 16.5.7.320 |
| 2478h  | QUEUE_71_C |               | Section 16.5.7.321 |
| 247Ch  | QUEUE_71_D |               | Section 16.5.7.322 |
| 2480h  | QUEUE_72_A |               | Section 16.5.7.323 |
| 2484h  | QUEUE_72_B |               | Section 16.5.7.324 |
| 2488h  | QUEUE_72_C |               | Section 16.5.7.325 |
| 248Ch  | QUEUE_72_D |               | Section 16.5.7.326 |
| 2490h  | QUEUE_73_A |               | Section 16.5.7.327 |
| 2494h  | QUEUE_73_B |               | Section 16.5.7.328 |
| 2498h  | QUEUE_73_C |               | Section 16.5.7.329 |
| 249Ch  | QUEUE_73_D |               | Section 16.5.7.330 |
| 24A0h  | QUEUE_74_A |               | Section 16.5.7.331 |
| 24A4h  | QUEUE_74_B |               | Section 16.5.7.332 |
| 24A8h  | QUEUE_74_C |               | Section 16.5.7.333 |
| 24ACh  | QUEUE_74_D |               | Section 16.5.7.334 |
| 24B0h  | QUEUE_75_A |               | Section 16.5.7.335 |
| 24B4h  | QUEUE_75_B |               | Section 16.5.7.336 |
| 24B8h  | QUEUE_75_C |               | Section 16.5.7.337 |
| 24BCh  | QUEUE_75_D |               | Section 16.5.7.338 |
| 24C0h  | QUEUE_76_A |               | Section 16.5.7.339 |
| 24C4h  | QUEUE_76_B |               | Section 16.5.7.340 |
| 24C8h  | QUEUE_76_C |               | Section 16.5.7.341 |
| 24CCh  | QUEUE_76_D |               | Section 16.5.7.342 |
| 24D0h  | QUEUE_77_A |               | Section 16.5.7.343 |
| 24D4h  | QUEUE_77_B |               | Section 16.5.7.344 |
| 24D8h  | QUEUE_77_C |               | Section 16.5.7.345 |
| 24DCh  | QUEUE_77_D |               | Section 16.5.7.346 |
| 24E0h  | QUEUE_78_A |               | Section 16.5.7.347 |
| 24E4h  | QUEUE_78_B |               | Section 16.5.7.348 |
| 24E8h  | QUEUE_78_C |               | Section 16.5.7.349 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym    | Register Name | Section            |
|--------|------------|---------------|--------------------|
| 24ECh  | QUEUE_78_D |               | Section 16.5.7.350 |
| 24F0h  | QUEUE_79_A |               | Section 16.5.7.351 |
| 24F4h  | QUEUE_79_B |               | Section 16.5.7.352 |
| 24F8h  | QUEUE_79_C |               | Section 16.5.7.353 |
| 24FCh  | QUEUE_79_D |               | Section 16.5.7.354 |
| 2500h  | QUEUE_80_A |               | Section 16.5.7.355 |
| 2504h  | QUEUE_80_B |               | Section 16.5.7.356 |
| 2508h  | QUEUE_80_C |               | Section 16.5.7.357 |
| 250Ch  | QUEUE_80_D |               | Section 16.5.7.358 |
| 2510h  | QUEUE_81_A |               | Section 16.5.7.359 |
| 2514h  | QUEUE_81_B |               | Section 16.5.7.360 |
| 2518h  | QUEUE_81_C |               | Section 16.5.7.361 |
| 251Ch  | QUEUE_81_D |               | Section 16.5.7.362 |
| 2520h  | QUEUE_82_A |               | Section 16.5.7.363 |
| 2524h  | QUEUE_82_B |               | Section 16.5.7.364 |
| 2528h  | QUEUE_82_C |               | Section 16.5.7.365 |
| 252Ch  | QUEUE_82_D |               | Section 16.5.7.366 |
| 2530h  | QUEUE_83_A |               | Section 16.5.7.367 |
| 2534h  | QUEUE_83_B |               | Section 16.5.7.368 |
| 2538h  | QUEUE_83_C |               | Section 16.5.7.369 |
| 253Ch  | QUEUE_83_D |               | Section 16.5.7.370 |
| 2540h  | QUEUE_84_A |               | Section 16.5.7.371 |
| 2544h  | QUEUE_84_B |               | Section 16.5.7.372 |
| 2548h  | QUEUE_84_C |               | Section 16.5.7.373 |
| 254Ch  | QUEUE_84_D |               | Section 16.5.7.374 |
| 2550h  | QUEUE_85_A |               | Section 16.5.7.375 |
| 2554h  | QUEUE_85_B |               | Section 16.5.7.376 |
| 2558h  | QUEUE_85_C |               | Section 16.5.7.377 |
| 255Ch  | QUEUE_85_D |               | Section 16.5.7.378 |
| 2560h  | QUEUE_86_A |               | Section 16.5.7.379 |
| 2564h  | QUEUE_86_B |               | Section 16.5.7.380 |
| 2568h  | QUEUE_86_C |               | Section 16.5.7.381 |
| 256Ch  | QUEUE_86_D |               | Section 16.5.7.382 |
| 2570h  | QUEUE_87_A |               | Section 16.5.7.383 |
| 2574h  | QUEUE_87_B |               | Section 16.5.7.384 |
| 2578h  | QUEUE_87_C |               | Section 16.5.7.385 |
| 257Ch  | QUEUE_87_D |               | Section 16.5.7.386 |
| 2580h  | QUEUE_88_A |               | Section 16.5.7.387 |
| 2584h  | QUEUE_88_B |               | Section 16.5.7.388 |
| 2588h  | QUEUE_88_C |               | Section 16.5.7.389 |
| 258Ch  | QUEUE_88_D |               | Section 16.5.7.390 |
| 2590h  | QUEUE_89_A |               | Section 16.5.7.391 |
| 2594h  | QUEUE_89_B |               | Section 16.5.7.392 |
| 2598h  | QUEUE_89_C |               | Section 16.5.7.393 |
| 259Ch  | QUEUE_89_D |               | Section 16.5.7.394 |
| 25A0h  | QUEUE_90_A |               | Section 16.5.7.395 |
| 25A4h  | QUEUE_90_B |               | Section 16.5.7.396 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym     | Register Name | Section                            |
|--------|-------------|---------------|------------------------------------|
| 25A8h  | QUEUE_90_C  |               | <a href="#">Section 16.5.7.397</a> |
| 25ACh  | QUEUE_90_D  |               | <a href="#">Section 16.5.7.398</a> |
| 25B0h  | QUEUE_91_A  |               | <a href="#">Section 16.5.7.399</a> |
| 25B4h  | QUEUE_91_B  |               | <a href="#">Section 16.5.7.400</a> |
| 25B8h  | QUEUE_91_C  |               | <a href="#">Section 16.5.7.401</a> |
| 25BCh  | QUEUE_91_D  |               | <a href="#">Section 16.5.7.402</a> |
| 25C0h  | QUEUE_92_A  |               | <a href="#">Section 16.5.7.403</a> |
| 25C4h  | QUEUE_92_B  |               | <a href="#">Section 16.5.7.404</a> |
| 25C8h  | QUEUE_92_C  |               | <a href="#">Section 16.5.7.405</a> |
| 25CCh  | QUEUE_92_D  |               | <a href="#">Section 16.5.7.406</a> |
| 25D0h  | QUEUE_93_A  |               | <a href="#">Section 16.5.7.407</a> |
| 25D4h  | QUEUE_93_B  |               | <a href="#">Section 16.5.7.408</a> |
| 25D8h  | QUEUE_93_C  |               | <a href="#">Section 16.5.7.409</a> |
| 25DCh  | QUEUE_93_D  |               | <a href="#">Section 16.5.7.410</a> |
| 25E0h  | QUEUE_94_A  |               | <a href="#">Section 16.5.7.411</a> |
| 25E4h  | QUEUE_94_B  |               | <a href="#">Section 16.5.7.412</a> |
| 25E8h  | QUEUE_94_C  |               | <a href="#">Section 16.5.7.413</a> |
| 25ECh  | QUEUE_94_D  |               | <a href="#">Section 16.5.7.414</a> |
| 25F0h  | QUEUE_95_A  |               | <a href="#">Section 16.5.7.415</a> |
| 25F4h  | QUEUE_95_B  |               | <a href="#">Section 16.5.7.416</a> |
| 25F8h  | QUEUE_95_C  |               | <a href="#">Section 16.5.7.417</a> |
| 25FCCh | QUEUE_95_D  |               | <a href="#">Section 16.5.7.418</a> |
| 2600h  | QUEUE_96_A  |               | <a href="#">Section 16.5.7.419</a> |
| 2604h  | QUEUE_96_B  |               | <a href="#">Section 16.5.7.420</a> |
| 2608h  | QUEUE_96_C  |               | <a href="#">Section 16.5.7.421</a> |
| 260Ch  | QUEUE_96_D  |               | <a href="#">Section 16.5.7.422</a> |
| 2610h  | QUEUE_97_A  |               | <a href="#">Section 16.5.7.423</a> |
| 2614h  | QUEUE_97_B  |               | <a href="#">Section 16.5.7.424</a> |
| 2618h  | QUEUE_97_C  |               | <a href="#">Section 16.5.7.425</a> |
| 261Ch  | QUEUE_97_D  |               | <a href="#">Section 16.5.7.426</a> |
| 2620h  | QUEUE_98_A  |               | <a href="#">Section 16.5.7.427</a> |
| 2624h  | QUEUE_98_B  |               | <a href="#">Section 16.5.7.428</a> |
| 2628h  | QUEUE_98_C  |               | <a href="#">Section 16.5.7.429</a> |
| 262Ch  | QUEUE_98_D  |               | <a href="#">Section 16.5.7.430</a> |
| 2630h  | QUEUE_99_A  |               | <a href="#">Section 16.5.7.431</a> |
| 2634h  | QUEUE_99_B  |               | <a href="#">Section 16.5.7.432</a> |
| 2638h  | QUEUE_99_C  |               | <a href="#">Section 16.5.7.433</a> |
| 263Ch  | QUEUE_99_D  |               | <a href="#">Section 16.5.7.434</a> |
| 2640h  | QUEUE_100_A |               | <a href="#">Section 16.5.7.435</a> |
| 2644h  | QUEUE_100_B |               | <a href="#">Section 16.5.7.436</a> |
| 2648h  | QUEUE_100_C |               | <a href="#">Section 16.5.7.437</a> |
| 264Ch  | QUEUE_100_D |               | <a href="#">Section 16.5.7.438</a> |
| 2650h  | QUEUE_101_A |               | <a href="#">Section 16.5.7.439</a> |
| 2654h  | QUEUE_101_B |               | <a href="#">Section 16.5.7.440</a> |
| 2658h  | QUEUE_101_C |               | <a href="#">Section 16.5.7.441</a> |
| 265Ch  | QUEUE_101_D |               | <a href="#">Section 16.5.7.442</a> |
| 2660h  | QUEUE_102_A |               | <a href="#">Section 16.5.7.443</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym     | Register Name | Section            |
|--------|-------------|---------------|--------------------|
| 2664h  | QUEUE_102_B |               | Section 16.5.7.444 |
| 2668h  | QUEUE_102_C |               | Section 16.5.7.445 |
| 266Ch  | QUEUE_102_D |               | Section 16.5.7.446 |
| 2670h  | QUEUE_103_A |               | Section 16.5.7.447 |
| 2674h  | QUEUE_103_B |               | Section 16.5.7.448 |
| 2678h  | QUEUE_103_C |               | Section 16.5.7.449 |
| 267Ch  | QUEUE_103_D |               | Section 16.5.7.450 |
| 2680h  | QUEUE_104_A |               | Section 16.5.7.451 |
| 2684h  | QUEUE_104_B |               | Section 16.5.7.452 |
| 2688h  | QUEUE_104_C |               | Section 16.5.7.453 |
| 268Ch  | QUEUE_104_D |               | Section 16.5.7.454 |
| 2690h  | QUEUE_105_A |               | Section 16.5.7.455 |
| 2694h  | QUEUE_105_B |               | Section 16.5.7.456 |
| 2698h  | QUEUE_105_C |               | Section 16.5.7.457 |
| 269Ch  | QUEUE_105_D |               | Section 16.5.7.458 |
| 26A0h  | QUEUE_106_A |               | Section 16.5.7.459 |
| 26A4h  | QUEUE_106_B |               | Section 16.5.7.460 |
| 26A8h  | QUEUE_106_C |               | Section 16.5.7.461 |
| 26ACh  | QUEUE_106_D |               | Section 16.5.7.462 |
| 26B0h  | QUEUE_107_A |               | Section 16.5.7.463 |
| 26B4h  | QUEUE_107_B |               | Section 16.5.7.464 |
| 26B8h  | QUEUE_107_C |               | Section 16.5.7.465 |
| 26BCh  | QUEUE_107_D |               | Section 16.5.7.466 |
| 26C0h  | QUEUE_108_A |               | Section 16.5.7.467 |
| 26C4h  | QUEUE_108_B |               | Section 16.5.7.468 |
| 26C8h  | QUEUE_108_C |               | Section 16.5.7.469 |
| 26CCh  | QUEUE_108_D |               | Section 16.5.7.470 |
| 26D0h  | QUEUE_109_A |               | Section 16.5.7.471 |
| 26D4h  | QUEUE_109_B |               | Section 16.5.7.472 |
| 26D8h  | QUEUE_109_C |               | Section 16.5.7.473 |
| 26DCh  | QUEUE_109_D |               | Section 16.5.7.474 |
| 26E0h  | QUEUE_110_A |               | Section 16.5.7.475 |
| 26E4h  | QUEUE_110_B |               | Section 16.5.7.476 |
| 26E8h  | QUEUE_110_C |               | Section 16.5.7.477 |
| 26ECh  | QUEUE_110_D |               | Section 16.5.7.478 |
| 26F0h  | QUEUE_111_A |               | Section 16.5.7.479 |
| 26F4h  | QUEUE_111_B |               | Section 16.5.7.480 |
| 26F8h  | QUEUE_111_C |               | Section 16.5.7.481 |
| 26FCh  | QUEUE_111_D |               | Section 16.5.7.482 |
| 2700h  | QUEUE_112_A |               | Section 16.5.7.483 |
| 2704h  | QUEUE_112_B |               | Section 16.5.7.484 |
| 2708h  | QUEUE_112_C |               | Section 16.5.7.485 |
| 270Ch  | QUEUE_112_D |               | Section 16.5.7.486 |
| 2710h  | QUEUE_113_A |               | Section 16.5.7.487 |
| 2714h  | QUEUE_113_B |               | Section 16.5.7.488 |
| 2718h  | QUEUE_113_C |               | Section 16.5.7.489 |
| 271Ch  | QUEUE_113_D |               | Section 16.5.7.490 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym     | Register Name | Section                            |
|--------|-------------|---------------|------------------------------------|
| 2720h  | QUEUE_114_A |               | <a href="#">Section 16.5.7.491</a> |
| 2724h  | QUEUE_114_B |               | <a href="#">Section 16.5.7.492</a> |
| 2728h  | QUEUE_114_C |               | <a href="#">Section 16.5.7.493</a> |
| 272Ch  | QUEUE_114_D |               | <a href="#">Section 16.5.7.494</a> |
| 2730h  | QUEUE_115_A |               | <a href="#">Section 16.5.7.495</a> |
| 2734h  | QUEUE_115_B |               | <a href="#">Section 16.5.7.496</a> |
| 2738h  | QUEUE_115_C |               | <a href="#">Section 16.5.7.497</a> |
| 273Ch  | QUEUE_115_D |               | <a href="#">Section 16.5.7.498</a> |
| 2740h  | QUEUE_116_A |               | <a href="#">Section 16.5.7.499</a> |
| 2744h  | QUEUE_116_B |               | <a href="#">Section 16.5.7.500</a> |
| 2748h  | QUEUE_116_C |               | <a href="#">Section 16.5.7.501</a> |
| 274Ch  | QUEUE_116_D |               | <a href="#">Section 16.5.7.502</a> |
| 2750h  | QUEUE_117_A |               | <a href="#">Section 16.5.7.503</a> |
| 2754h  | QUEUE_117_B |               | <a href="#">Section 16.5.7.504</a> |
| 2758h  | QUEUE_117_C |               | <a href="#">Section 16.5.7.505</a> |
| 275Ch  | QUEUE_117_D |               | <a href="#">Section 16.5.7.506</a> |
| 2760h  | QUEUE_118_A |               | <a href="#">Section 16.5.7.507</a> |
| 2764h  | QUEUE_118_B |               | <a href="#">Section 16.5.7.508</a> |
| 2768h  | QUEUE_118_C |               | <a href="#">Section 16.5.7.509</a> |
| 276Ch  | QUEUE_118_D |               | <a href="#">Section 16.5.7.510</a> |
| 2770h  | QUEUE_119_A |               | <a href="#">Section 16.5.7.511</a> |
| 2774h  | QUEUE_119_B |               | <a href="#">Section 16.5.7.512</a> |
| 2778h  | QUEUE_119_C |               | <a href="#">Section 16.5.7.513</a> |
| 277Ch  | QUEUE_119_D |               | <a href="#">Section 16.5.7.514</a> |
| 2780h  | QUEUE_120_A |               | <a href="#">Section 16.5.7.515</a> |
| 2784h  | QUEUE_120_B |               | <a href="#">Section 16.5.7.516</a> |
| 2788h  | QUEUE_120_C |               | <a href="#">Section 16.5.7.517</a> |
| 278Ch  | QUEUE_120_D |               | <a href="#">Section 16.5.7.518</a> |
| 2790h  | QUEUE_121_A |               | <a href="#">Section 16.5.7.519</a> |
| 2794h  | QUEUE_121_B |               | <a href="#">Section 16.5.7.520</a> |
| 2798h  | QUEUE_121_C |               | <a href="#">Section 16.5.7.521</a> |
| 279Ch  | QUEUE_121_D |               | <a href="#">Section 16.5.7.522</a> |
| 27A0h  | QUEUE_122_A |               | <a href="#">Section 16.5.7.523</a> |
| 27A4h  | QUEUE_122_B |               | <a href="#">Section 16.5.7.524</a> |
| 27A8h  | QUEUE_122_C |               | <a href="#">Section 16.5.7.525</a> |
| 27ACh  | QUEUE_122_D |               | <a href="#">Section 16.5.7.526</a> |
| 27B0h  | QUEUE_123_A |               | <a href="#">Section 16.5.7.527</a> |
| 27B4h  | QUEUE_123_B |               | <a href="#">Section 16.5.7.528</a> |
| 27B8h  | QUEUE_123_C |               | <a href="#">Section 16.5.7.529</a> |
| 27BCh  | QUEUE_123_D |               | <a href="#">Section 16.5.7.530</a> |
| 27C0h  | QUEUE_124_A |               | <a href="#">Section 16.5.7.531</a> |
| 27C4h  | QUEUE_124_B |               | <a href="#">Section 16.5.7.532</a> |
| 27C8h  | QUEUE_124_C |               | <a href="#">Section 16.5.7.533</a> |
| 27CCh  | QUEUE_124_D |               | <a href="#">Section 16.5.7.534</a> |
| 27D0h  | QUEUE_125_A |               | <a href="#">Section 16.5.7.535</a> |
| 27D4h  | QUEUE_125_B |               | <a href="#">Section 16.5.7.536</a> |
| 27D8h  | QUEUE_125_C |               | <a href="#">Section 16.5.7.537</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b> | <b>Section</b>     |
|---------------|----------------|----------------------|--------------------|
| 27DCh         | QUEUE_125_D    |                      | Section 16.5.7.538 |
| 27E0h         | QUEUE_126_A    |                      | Section 16.5.7.539 |
| 27E4h         | QUEUE_126_B    |                      | Section 16.5.7.540 |
| 27E8h         | QUEUE_126_C    |                      | Section 16.5.7.541 |
| 27ECh         | QUEUE_126_D    |                      | Section 16.5.7.542 |
| 27F0h         | QUEUE_127_A    |                      | Section 16.5.7.543 |
| 27F4h         | QUEUE_127_B    |                      | Section 16.5.7.544 |
| 27F8h         | QUEUE_127_C    |                      | Section 16.5.7.545 |
| 27FCh         | QUEUE_127_D    |                      | Section 16.5.7.546 |
| 2800h         | QUEUE_128_A    |                      | Section 16.5.7.547 |
| 2804h         | QUEUE_128_B    |                      | Section 16.5.7.548 |
| 2808h         | QUEUE_128_C    |                      | Section 16.5.7.549 |
| 280Ch         | QUEUE_128_D    |                      | Section 16.5.7.550 |
| 2810h         | QUEUE_129_A    |                      | Section 16.5.7.551 |
| 2814h         | QUEUE_129_B    |                      | Section 16.5.7.552 |
| 2818h         | QUEUE_129_C    |                      | Section 16.5.7.553 |
| 281Ch         | QUEUE_129_D    |                      | Section 16.5.7.554 |
| 2820h         | QUEUE_130_A    |                      | Section 16.5.7.555 |
| 2824h         | QUEUE_130_B    |                      | Section 16.5.7.556 |
| 2828h         | QUEUE_130_C    |                      | Section 16.5.7.557 |
| 282Ch         | QUEUE_130_D    |                      | Section 16.5.7.558 |
| 2830h         | QUEUE_131_A    |                      | Section 16.5.7.559 |
| 2834h         | QUEUE_131_B    |                      | Section 16.5.7.560 |
| 2838h         | QUEUE_131_C    |                      | Section 16.5.7.561 |
| 283Ch         | QUEUE_131_D    |                      | Section 16.5.7.562 |
| 2840h         | QUEUE_132_A    |                      | Section 16.5.7.563 |
| 2844h         | QUEUE_132_B    |                      | Section 16.5.7.564 |
| 2848h         | QUEUE_132_C    |                      | Section 16.5.7.565 |
| 284Ch         | QUEUE_132_D    |                      | Section 16.5.7.566 |
| 2850h         | QUEUE_133_A    |                      | Section 16.5.7.567 |
| 2854h         | QUEUE_133_B    |                      | Section 16.5.7.568 |
| 2858h         | QUEUE_133_C    |                      | Section 16.5.7.569 |
| 285Ch         | QUEUE_133_D    |                      | Section 16.5.7.570 |
| 2860h         | QUEUE_134_A    |                      | Section 16.5.7.571 |
| 2864h         | QUEUE_134_B    |                      | Section 16.5.7.572 |
| 2868h         | QUEUE_134_C    |                      | Section 16.5.7.573 |
| 286Ch         | QUEUE_134_D    |                      | Section 16.5.7.574 |
| 2870h         | QUEUE_135_A    |                      | Section 16.5.7.575 |
| 2874h         | QUEUE_135_B    |                      | Section 16.5.7.576 |
| 2878h         | QUEUE_135_C    |                      | Section 16.5.7.577 |
| 287Ch         | QUEUE_135_D    |                      | Section 16.5.7.578 |
| 2880h         | QUEUE_136_A    |                      | Section 16.5.7.579 |
| 2884h         | QUEUE_136_B    |                      | Section 16.5.7.580 |
| 2888h         | QUEUE_136_C    |                      | Section 16.5.7.581 |
| 288Ch         | QUEUE_136_D    |                      | Section 16.5.7.582 |
| 2890h         | QUEUE_137_A    |                      | Section 16.5.7.583 |
| 2894h         | QUEUE_137_B    |                      | Section 16.5.7.584 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym     | Register Name | Section            |
|--------|-------------|---------------|--------------------|
| 2898h  | QUEUE_137_C |               | Section 16.5.7.585 |
| 289Ch  | QUEUE_137_D |               | Section 16.5.7.586 |
| 28A0h  | QUEUE_138_A |               | Section 16.5.7.587 |
| 28A4h  | QUEUE_138_B |               | Section 16.5.7.588 |
| 28A8h  | QUEUE_138_C |               | Section 16.5.7.589 |
| 28ACh  | QUEUE_138_D |               | Section 16.5.7.590 |
| 28B0h  | QUEUE_139_A |               | Section 16.5.7.591 |
| 28B4h  | QUEUE_139_B |               | Section 16.5.7.592 |
| 28B8h  | QUEUE_139_C |               | Section 16.5.7.593 |
| 28BCh  | QUEUE_139_D |               | Section 16.5.7.594 |
| 28C0h  | QUEUE_140_A |               | Section 16.5.7.595 |
| 28C4h  | QUEUE_140_B |               | Section 16.5.7.596 |
| 28C8h  | QUEUE_140_C |               | Section 16.5.7.597 |
| 28CCh  | QUEUE_140_D |               | Section 16.5.7.598 |
| 28D0h  | QUEUE_141_A |               | Section 16.5.7.599 |
| 28D4h  | QUEUE_141_B |               | Section 16.5.7.600 |
| 28D8h  | QUEUE_141_C |               | Section 16.5.7.601 |
| 28DCh  | QUEUE_141_D |               | Section 16.5.7.602 |
| 28E0h  | QUEUE_142_A |               | Section 16.5.7.603 |
| 28E4h  | QUEUE_142_B |               | Section 16.5.7.604 |
| 28E8h  | QUEUE_142_C |               | Section 16.5.7.605 |
| 28ECh  | QUEUE_142_D |               | Section 16.5.7.606 |
| 28F0h  | QUEUE_143_A |               | Section 16.5.7.607 |
| 28F4h  | QUEUE_143_B |               | Section 16.5.7.608 |
| 28F8h  | QUEUE_143_C |               | Section 16.5.7.609 |
| 28FCh  | QUEUE_143_D |               | Section 16.5.7.610 |
| 2900h  | QUEUE_144_A |               | Section 16.5.7.611 |
| 2904h  | QUEUE_144_B |               | Section 16.5.7.612 |
| 2908h  | QUEUE_144_C |               | Section 16.5.7.613 |
| 290Ch  | QUEUE_144_D |               | Section 16.5.7.614 |
| 2910h  | QUEUE_145_A |               | Section 16.5.7.615 |
| 2914h  | QUEUE_145_B |               | Section 16.5.7.616 |
| 2918h  | QUEUE_145_C |               | Section 16.5.7.617 |
| 291Ch  | QUEUE_145_D |               | Section 16.5.7.618 |
| 2920h  | QUEUE_146_A |               | Section 16.5.7.619 |
| 2924h  | QUEUE_146_B |               | Section 16.5.7.620 |
| 2928h  | QUEUE_146_C |               | Section 16.5.7.621 |
| 292Ch  | QUEUE_146_D |               | Section 16.5.7.622 |
| 2930h  | QUEUE_147_A |               | Section 16.5.7.623 |
| 2934h  | QUEUE_147_B |               | Section 16.5.7.624 |
| 2938h  | QUEUE_147_C |               | Section 16.5.7.625 |
| 293Ch  | QUEUE_147_D |               | Section 16.5.7.626 |
| 2940h  | QUEUE_148_A |               | Section 16.5.7.627 |
| 2944h  | QUEUE_148_B |               | Section 16.5.7.628 |
| 2948h  | QUEUE_148_C |               | Section 16.5.7.629 |
| 294Ch  | QUEUE_148_D |               | Section 16.5.7.630 |
| 2950h  | QUEUE_149_A |               | Section 16.5.7.631 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b>   | <b>Register Name</b> | <b>Section</b>     |
|---------------|------------------|----------------------|--------------------|
| 2954h         | QUEUE_149_B      |                      | Section 16.5.7.632 |
| 2958h         | QUEUE_149_C      |                      | Section 16.5.7.633 |
| 295Ch         | QUEUE_149_D      |                      | Section 16.5.7.634 |
| 2960h         | QUEUE_150_A      |                      | Section 16.5.7.635 |
| 2964h         | QUEUE_150_B      |                      | Section 16.5.7.636 |
| 2968h         | QUEUE_150_C      |                      | Section 16.5.7.637 |
| 296Ch         | QUEUE_150_D      |                      | Section 16.5.7.638 |
| 2970h         | QUEUE_151_A      |                      | Section 16.5.7.639 |
| 2974h         | QUEUE_151_B      |                      | Section 16.5.7.640 |
| 2978h         | QUEUE_151_C      |                      | Section 16.5.7.641 |
| 297Ch         | QUEUE_151_D      |                      | Section 16.5.7.642 |
| 2980h         | QUEUE_152_A      |                      | Section 16.5.7.643 |
| 2984h         | QUEUE_152_B      |                      | Section 16.5.7.644 |
| 2988h         | QUEUE_152_C      |                      | Section 16.5.7.645 |
| 298Ch         | QUEUE_152_D      |                      | Section 16.5.7.646 |
| 2990h         | QUEUE_153_A      |                      | Section 16.5.7.647 |
| 2994h         | QUEUE_153_B      |                      | Section 16.5.7.648 |
| 2998h         | QUEUE_153_C      |                      | Section 16.5.7.649 |
| 299Ch         | QUEUE_153_D      |                      | Section 16.5.7.650 |
| 29A0h         | QUEUE_154_A      |                      | Section 16.5.7.651 |
| 29A4h         | QUEUE_154_B      |                      | Section 16.5.7.652 |
| 29A8h         | QUEUE_154_C      |                      | Section 16.5.7.653 |
| 29ACh         | QUEUE_154_D      |                      | Section 16.5.7.654 |
| 29B0h         | QUEUE_155_A      |                      | Section 16.5.7.655 |
| 29B4h         | QUEUE_155_B      |                      | Section 16.5.7.656 |
| 29B8h         | QUEUE_155_C      |                      | Section 16.5.7.657 |
| 29BCh         | QUEUE_155_D      |                      | Section 16.5.7.658 |
| 3000h         | QUEUE_0_STATUS_A |                      | Section 16.5.7.659 |
| 3004h         | QUEUE_0_STATUS_B |                      | Section 16.5.7.660 |
| 3008h         | QUEUE_0_STATUS_C |                      | Section 16.5.7.661 |
| 3010h         | QUEUE_1_STATUS_A |                      | Section 16.5.7.662 |
| 3014h         | QUEUE_1_STATUS_B |                      | Section 16.5.7.663 |
| 3018h         | QUEUE_1_STATUS_C |                      | Section 16.5.7.664 |
| 3020h         | QUEUE_2_STATUS_A |                      | Section 16.5.7.665 |
| 3024h         | QUEUE_2_STATUS_B |                      | Section 16.5.7.666 |
| 3028h         | QUEUE_2_STATUS_C |                      | Section 16.5.7.667 |
| 3030h         | QUEUE_3_STATUS_A |                      | Section 16.5.7.668 |
| 3034h         | QUEUE_3_STATUS_B |                      | Section 16.5.7.669 |
| 3038h         | QUEUE_3_STATUS_C |                      | Section 16.5.7.670 |
| 3040h         | QUEUE_4_STATUS_A |                      | Section 16.5.7.671 |
| 3044h         | QUEUE_4_STATUS_B |                      | Section 16.5.7.672 |
| 3048h         | QUEUE_4_STATUS_C |                      | Section 16.5.7.673 |
| 3050h         | QUEUE_5_STATUS_A |                      | Section 16.5.7.674 |
| 3054h         | QUEUE_5_STATUS_B |                      | Section 16.5.7.675 |
| 3058h         | QUEUE_5_STATUS_C |                      | Section 16.5.7.676 |
| 3060h         | QUEUE_6_STATUS_A |                      | Section 16.5.7.677 |
| 3064h         | QUEUE_6_STATUS_B |                      | Section 16.5.7.678 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section            |
|--------|-------------------|---------------|--------------------|
| 3068h  | QUEUE_6_STATUS_C  |               | Section 16.5.7.679 |
| 3070h  | QUEUE_7_STATUS_A  |               | Section 16.5.7.680 |
| 3074h  | QUEUE_7_STATUS_B  |               | Section 16.5.7.681 |
| 3078h  | QUEUE_7_STATUS_C  |               | Section 16.5.7.682 |
| 3080h  | QUEUE_8_STATUS_A  |               | Section 16.5.7.683 |
| 3084h  | QUEUE_8_STATUS_B  |               | Section 16.5.7.684 |
| 3088h  | QUEUE_8_STATUS_C  |               | Section 16.5.7.685 |
| 3090h  | QUEUE_9_STATUS_A  |               | Section 16.5.7.686 |
| 3094h  | QUEUE_9_STATUS_B  |               | Section 16.5.7.687 |
| 3098h  | QUEUE_9_STATUS_C  |               | Section 16.5.7.688 |
| 30A0h  | QUEUE_10_STATUS_A |               | Section 16.5.7.689 |
| 30A4h  | QUEUE_10_STATUS_B |               | Section 16.5.7.690 |
| 30A8h  | QUEUE_10_STATUS_C |               | Section 16.5.7.691 |
| 30B0h  | QUEUE_11_STATUS_A |               | Section 16.5.7.692 |
| 30B4h  | QUEUE_11_STATUS_B |               | Section 16.5.7.693 |
| 30B8h  | QUEUE_11_STATUS_C |               | Section 16.5.7.694 |
| 30C0h  | QUEUE_12_STATUS_A |               | Section 16.5.7.695 |
| 30C4h  | QUEUE_12_STATUS_B |               | Section 16.5.7.696 |
| 30C8h  | QUEUE_12_STATUS_C |               | Section 16.5.7.697 |
| 30D0h  | QUEUE_13_STATUS_A |               | Section 16.5.7.698 |
| 30D4h  | QUEUE_13_STATUS_B |               | Section 16.5.7.699 |
| 30D8h  | QUEUE_13_STATUS_C |               | Section 16.5.7.700 |
| 30E0h  | QUEUE_14_STATUS_A |               | Section 16.5.7.701 |
| 30E4h  | QUEUE_14_STATUS_B |               | Section 16.5.7.702 |
| 30E8h  | QUEUE_14_STATUS_C |               | Section 16.5.7.703 |
| 30F0h  | QUEUE_15_STATUS_A |               | Section 16.5.7.704 |
| 30F4h  | QUEUE_15_STATUS_B |               | Section 16.5.7.705 |
| 30F8h  | QUEUE_15_STATUS_C |               | Section 16.5.7.706 |
| 3100h  | QUEUE_16_STATUS_A |               | Section 16.5.7.707 |
| 3104h  | QUEUE_16_STATUS_B |               | Section 16.5.7.708 |
| 3108h  | QUEUE_16_STATUS_C |               | Section 16.5.7.709 |
| 3110h  | QUEUE_17_STATUS_A |               | Section 16.5.7.710 |
| 3114h  | QUEUE_17_STATUS_B |               | Section 16.5.7.711 |
| 3118h  | QUEUE_17_STATUS_C |               | Section 16.5.7.712 |
| 3120h  | QUEUE_18_STATUS_A |               | Section 16.5.7.713 |
| 3124h  | QUEUE_18_STATUS_B |               | Section 16.5.7.714 |
| 3128h  | QUEUE_18_STATUS_C |               | Section 16.5.7.715 |
| 3130h  | QUEUE_19_STATUS_A |               | Section 16.5.7.716 |
| 3134h  | QUEUE_19_STATUS_B |               | Section 16.5.7.717 |
| 3138h  | QUEUE_19_STATUS_C |               | Section 16.5.7.718 |
| 3140h  | QUEUE_20_STATUS_A |               | Section 16.5.7.719 |
| 3144h  | QUEUE_20_STATUS_B |               | Section 16.5.7.720 |
| 3148h  | QUEUE_20_STATUS_C |               | Section 16.5.7.721 |
| 3150h  | QUEUE_21_STATUS_A |               | Section 16.5.7.722 |
| 3154h  | QUEUE_21_STATUS_B |               | Section 16.5.7.723 |
| 3158h  | QUEUE_21_STATUS_C |               | Section 16.5.7.724 |
| 3160h  | QUEUE_22_STATUS_A |               | Section 16.5.7.725 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section            |
|--------|-------------------|---------------|--------------------|
| 3164h  | QUEUE_22_STATUS_B |               | Section 16.5.7.726 |
| 3168h  | QUEUE_22_STATUS_C |               | Section 16.5.7.727 |
| 3170h  | QUEUE_23_STATUS_A |               | Section 16.5.7.728 |
| 3174h  | QUEUE_23_STATUS_B |               | Section 16.5.7.729 |
| 3178h  | QUEUE_23_STATUS_C |               | Section 16.5.7.730 |
| 3180h  | QUEUE_24_STATUS_A |               | Section 16.5.7.731 |
| 3184h  | QUEUE_24_STATUS_B |               | Section 16.5.7.732 |
| 3188h  | QUEUE_24_STATUS_C |               | Section 16.5.7.733 |
| 3190h  | QUEUE_25_STATUS_A |               | Section 16.5.7.734 |
| 3194h  | QUEUE_25_STATUS_B |               | Section 16.5.7.735 |
| 3198h  | QUEUE_25_STATUS_C |               | Section 16.5.7.736 |
| 31A0h  | QUEUE_26_STATUS_A |               | Section 16.5.7.737 |
| 31A4h  | QUEUE_26_STATUS_B |               | Section 16.5.7.738 |
| 31A8h  | QUEUE_26_STATUS_C |               | Section 16.5.7.739 |
| 31B0h  | QUEUE_27_STATUS_A |               | Section 16.5.7.740 |
| 31B4h  | QUEUE_27_STATUS_B |               | Section 16.5.7.741 |
| 31B8h  | QUEUE_27_STATUS_C |               | Section 16.5.7.742 |
| 31C0h  | QUEUE_28_STATUS_A |               | Section 16.5.7.743 |
| 31C4h  | QUEUE_28_STATUS_B |               | Section 16.5.7.744 |
| 31C8h  | QUEUE_28_STATUS_C |               | Section 16.5.7.745 |
| 31D0h  | QUEUE_29_STATUS_A |               | Section 16.5.7.746 |
| 31D4h  | QUEUE_29_STATUS_B |               | Section 16.5.7.747 |
| 31D8h  | QUEUE_29_STATUS_C |               | Section 16.5.7.748 |
| 31E0h  | QUEUE_30_STATUS_A |               | Section 16.5.7.749 |
| 31E4h  | QUEUE_30_STATUS_B |               | Section 16.5.7.750 |
| 31E8h  | QUEUE_30_STATUS_C |               | Section 16.5.7.751 |
| 31F0h  | QUEUE_31_STATUS_A |               | Section 16.5.7.752 |
| 31F4h  | QUEUE_31_STATUS_B |               | Section 16.5.7.753 |
| 31F8h  | QUEUE_31_STATUS_C |               | Section 16.5.7.754 |
| 3200h  | QUEUE_32_STATUS_A |               | Section 16.5.7.755 |
| 3204h  | QUEUE_32_STATUS_B |               | Section 16.5.7.756 |
| 3208h  | QUEUE_32_STATUS_C |               | Section 16.5.7.757 |
| 3210h  | QUEUE_33_STATUS_A |               | Section 16.5.7.758 |
| 3214h  | QUEUE_33_STATUS_B |               | Section 16.5.7.759 |
| 3218h  | QUEUE_33_STATUS_C |               | Section 16.5.7.760 |
| 3220h  | QUEUE_34_STATUS_A |               | Section 16.5.7.761 |
| 3224h  | QUEUE_34_STATUS_B |               | Section 16.5.7.762 |
| 3228h  | QUEUE_34_STATUS_C |               | Section 16.5.7.763 |
| 3230h  | QUEUE_35_STATUS_A |               | Section 16.5.7.764 |
| 3234h  | QUEUE_35_STATUS_B |               | Section 16.5.7.765 |
| 3238h  | QUEUE_35_STATUS_C |               | Section 16.5.7.766 |
| 3240h  | QUEUE_36_STATUS_A |               | Section 16.5.7.767 |
| 3244h  | QUEUE_36_STATUS_B |               | Section 16.5.7.768 |
| 3248h  | QUEUE_36_STATUS_C |               | Section 16.5.7.769 |
| 3250h  | QUEUE_37_STATUS_A |               | Section 16.5.7.770 |
| 3254h  | QUEUE_37_STATUS_B |               | Section 16.5.7.771 |
| 3258h  | QUEUE_37_STATUS_C |               | Section 16.5.7.772 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section            |
|--------|-------------------|---------------|--------------------|
| 3260h  | QUEUE_38_STATUS_A |               | Section 16.5.7.773 |
| 3264h  | QUEUE_38_STATUS_B |               | Section 16.5.7.774 |
| 3268h  | QUEUE_38_STATUS_C |               | Section 16.5.7.775 |
| 3270h  | QUEUE_39_STATUS_A |               | Section 16.5.7.776 |
| 3274h  | QUEUE_39_STATUS_B |               | Section 16.5.7.777 |
| 3278h  | QUEUE_39_STATUS_C |               | Section 16.5.7.778 |
| 3280h  | QUEUE_40_STATUS_A |               | Section 16.5.7.779 |
| 3284h  | QUEUE_40_STATUS_B |               | Section 16.5.7.780 |
| 3288h  | QUEUE_40_STATUS_C |               | Section 16.5.7.781 |
| 3290h  | QUEUE_41_STATUS_A |               | Section 16.5.7.782 |
| 3294h  | QUEUE_41_STATUS_B |               | Section 16.5.7.783 |
| 3298h  | QUEUE_41_STATUS_C |               | Section 16.5.7.784 |
| 32A0h  | QUEUE_42_STATUS_A |               | Section 16.5.7.785 |
| 32A4h  | QUEUE_42_STATUS_B |               | Section 16.5.7.786 |
| 32A8h  | QUEUE_42_STATUS_C |               | Section 16.5.7.787 |
| 32B0h  | QUEUE_43_STATUS_A |               | Section 16.5.7.788 |
| 32B4h  | QUEUE_43_STATUS_B |               | Section 16.5.7.789 |
| 32B8h  | QUEUE_43_STATUS_C |               | Section 16.5.7.790 |
| 32C0h  | QUEUE_44_STATUS_A |               | Section 16.5.7.791 |
| 32C4h  | QUEUE_44_STATUS_B |               | Section 16.5.7.792 |
| 32C8h  | QUEUE_44_STATUS_C |               | Section 16.5.7.793 |
| 32D0h  | QUEUE_45_STATUS_A |               | Section 16.5.7.794 |
| 32D4h  | QUEUE_45_STATUS_B |               | Section 16.5.7.795 |
| 32D8h  | QUEUE_45_STATUS_C |               | Section 16.5.7.796 |
| 32E0h  | QUEUE_46_STATUS_A |               | Section 16.5.7.797 |
| 32E4h  | QUEUE_46_STATUS_B |               | Section 16.5.7.798 |
| 32E8h  | QUEUE_46_STATUS_C |               | Section 16.5.7.799 |
| 32F0h  | QUEUE_47_STATUS_A |               | Section 16.5.7.800 |
| 32F4h  | QUEUE_47_STATUS_B |               | Section 16.5.7.801 |
| 32F8h  | QUEUE_47_STATUS_C |               | Section 16.5.7.802 |
| 3300h  | QUEUE_48_STATUS_A |               | Section 16.5.7.803 |
| 3304h  | QUEUE_48_STATUS_B |               | Section 16.5.7.804 |
| 3308h  | QUEUE_48_STATUS_C |               | Section 16.5.7.805 |
| 3310h  | QUEUE_49_STATUS_A |               | Section 16.5.7.806 |
| 3314h  | QUEUE_49_STATUS_B |               | Section 16.5.7.807 |
| 3318h  | QUEUE_49_STATUS_C |               | Section 16.5.7.808 |
| 3320h  | QUEUE_50_STATUS_A |               | Section 16.5.7.809 |
| 3324h  | QUEUE_50_STATUS_B |               | Section 16.5.7.810 |
| 3328h  | QUEUE_50_STATUS_C |               | Section 16.5.7.811 |
| 3330h  | QUEUE_51_STATUS_A |               | Section 16.5.7.812 |
| 3334h  | QUEUE_51_STATUS_B |               | Section 16.5.7.813 |
| 3338h  | QUEUE_51_STATUS_C |               | Section 16.5.7.814 |
| 3340h  | QUEUE_52_STATUS_A |               | Section 16.5.7.815 |
| 3344h  | QUEUE_52_STATUS_B |               | Section 16.5.7.816 |
| 3348h  | QUEUE_52_STATUS_C |               | Section 16.5.7.817 |
| 3350h  | QUEUE_53_STATUS_A |               | Section 16.5.7.818 |
| 3354h  | QUEUE_53_STATUS_B |               | Section 16.5.7.819 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section            |
|--------|-------------------|---------------|--------------------|
| 3358h  | QUEUE_53_STATUS_C |               | Section 16.5.7.820 |
| 3360h  | QUEUE_54_STATUS_A |               | Section 16.5.7.821 |
| 3364h  | QUEUE_54_STATUS_B |               | Section 16.5.7.822 |
| 3368h  | QUEUE_54_STATUS_C |               | Section 16.5.7.823 |
| 3370h  | QUEUE_55_STATUS_A |               | Section 16.5.7.824 |
| 3374h  | QUEUE_55_STATUS_B |               | Section 16.5.7.825 |
| 3378h  | QUEUE_55_STATUS_C |               | Section 16.5.7.826 |
| 3380h  | QUEUE_56_STATUS_A |               | Section 16.5.7.827 |
| 3384h  | QUEUE_56_STATUS_B |               | Section 16.5.7.828 |
| 3388h  | QUEUE_56_STATUS_C |               | Section 16.5.7.829 |
| 3390h  | QUEUE_57_STATUS_A |               | Section 16.5.7.830 |
| 3394h  | QUEUE_57_STATUS_B |               | Section 16.5.7.831 |
| 3398h  | QUEUE_57_STATUS_C |               | Section 16.5.7.832 |
| 33A0h  | QUEUE_58_STATUS_A |               | Section 16.5.7.833 |
| 33A4h  | QUEUE_58_STATUS_B |               | Section 16.5.7.834 |
| 33A8h  | QUEUE_58_STATUS_C |               | Section 16.5.7.835 |
| 33B0h  | QUEUE_59_STATUS_A |               | Section 16.5.7.836 |
| 33B4h  | QUEUE_59_STATUS_B |               | Section 16.5.7.837 |
| 33B8h  | QUEUE_59_STATUS_C |               | Section 16.5.7.838 |
| 33C0h  | QUEUE_60_STATUS_A |               | Section 16.5.7.839 |
| 33C4h  | QUEUE_60_STATUS_B |               | Section 16.5.7.840 |
| 33C8h  | QUEUE_60_STATUS_C |               | Section 16.5.7.841 |
| 33D0h  | QUEUE_61_STATUS_A |               | Section 16.5.7.842 |
| 33D4h  | QUEUE_61_STATUS_B |               | Section 16.5.7.843 |
| 33D8h  | QUEUE_61_STATUS_C |               | Section 16.5.7.844 |
| 33E0h  | QUEUE_62_STATUS_A |               | Section 16.5.7.845 |
| 33E4h  | QUEUE_62_STATUS_B |               | Section 16.5.7.846 |
| 33E8h  | QUEUE_62_STATUS_C |               | Section 16.5.7.847 |
| 33F0h  | QUEUE_63_STATUS_A |               | Section 16.5.7.848 |
| 33F4h  | QUEUE_63_STATUS_B |               | Section 16.5.7.849 |
| 33F8h  | QUEUE_63_STATUS_C |               | Section 16.5.7.850 |
| 3400h  | QUEUE_64_STATUS_A |               | Section 16.5.7.851 |
| 3404h  | QUEUE_64_STATUS_B |               | Section 16.5.7.852 |
| 3408h  | QUEUE_64_STATUS_C |               | Section 16.5.7.853 |
| 3410h  | QUEUE_65_STATUS_A |               | Section 16.5.7.854 |
| 3414h  | QUEUE_65_STATUS_B |               | Section 16.5.7.855 |
| 3418h  | QUEUE_65_STATUS_C |               | Section 16.5.7.856 |
| 3420h  | QUEUE_66_STATUS_A |               | Section 16.5.7.857 |
| 3424h  | QUEUE_66_STATUS_B |               | Section 16.5.7.858 |
| 3428h  | QUEUE_66_STATUS_C |               | Section 16.5.7.859 |
| 3430h  | QUEUE_67_STATUS_A |               | Section 16.5.7.860 |
| 3434h  | QUEUE_67_STATUS_B |               | Section 16.5.7.861 |
| 3438h  | QUEUE_67_STATUS_C |               | Section 16.5.7.862 |
| 3440h  | QUEUE_68_STATUS_A |               | Section 16.5.7.863 |
| 3444h  | QUEUE_68_STATUS_B |               | Section 16.5.7.864 |
| 3448h  | QUEUE_68_STATUS_C |               | Section 16.5.7.865 |
| 3450h  | QUEUE_69_STATUS_A |               | Section 16.5.7.866 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym           | Register Name | Section                            |
|--------|-------------------|---------------|------------------------------------|
| 3454h  | QUEUE_69_STATUS_B |               | <a href="#">Section 16.5.7.867</a> |
| 3458h  | QUEUE_69_STATUS_C |               | <a href="#">Section 16.5.7.868</a> |
| 3460h  | QUEUE_70_STATUS_A |               | <a href="#">Section 16.5.7.869</a> |
| 3464h  | QUEUE_70_STATUS_B |               | <a href="#">Section 16.5.7.870</a> |
| 3468h  | QUEUE_70_STATUS_C |               | <a href="#">Section 16.5.7.871</a> |
| 3470h  | QUEUE_71_STATUS_A |               | <a href="#">Section 16.5.7.872</a> |
| 3474h  | QUEUE_71_STATUS_B |               | <a href="#">Section 16.5.7.873</a> |
| 3478h  | QUEUE_71_STATUS_C |               | <a href="#">Section 16.5.7.874</a> |
| 3480h  | QUEUE_72_STATUS_A |               | <a href="#">Section 16.5.7.875</a> |
| 3484h  | QUEUE_72_STATUS_B |               | <a href="#">Section 16.5.7.876</a> |
| 3488h  | QUEUE_72_STATUS_C |               | <a href="#">Section 16.5.7.877</a> |
| 3490h  | QUEUE_73_STATUS_A |               | <a href="#">Section 16.5.7.878</a> |
| 3494h  | QUEUE_73_STATUS_B |               | <a href="#">Section 16.5.7.879</a> |
| 3498h  | QUEUE_73_STATUS_C |               | <a href="#">Section 16.5.7.880</a> |
| 34A0h  | QUEUE_74_STATUS_A |               | <a href="#">Section 16.5.7.881</a> |
| 34A4h  | QUEUE_74_STATUS_B |               | <a href="#">Section 16.5.7.882</a> |
| 34A8h  | QUEUE_74_STATUS_C |               | <a href="#">Section 16.5.7.883</a> |
| 34B0h  | QUEUE_75_STATUS_A |               | <a href="#">Section 16.5.7.884</a> |
| 34B4h  | QUEUE_75_STATUS_B |               | <a href="#">Section 16.5.7.885</a> |
| 34B8h  | QUEUE_75_STATUS_C |               | <a href="#">Section 16.5.7.886</a> |
| 34C0h  | QUEUE_76_STATUS_A |               | <a href="#">Section 16.5.7.887</a> |
| 34C4h  | QUEUE_76_STATUS_B |               | <a href="#">Section 16.5.7.888</a> |
| 34C8h  | QUEUE_76_STATUS_C |               | <a href="#">Section 16.5.7.889</a> |
| 34D0h  | QUEUE_77_STATUS_A |               | <a href="#">Section 16.5.7.890</a> |
| 34D4h  | QUEUE_77_STATUS_B |               | <a href="#">Section 16.5.7.891</a> |
| 34D8h  | QUEUE_77_STATUS_C |               | <a href="#">Section 16.5.7.892</a> |
| 34E0h  | QUEUE_78_STATUS_A |               | <a href="#">Section 16.5.7.893</a> |
| 34E4h  | QUEUE_78_STATUS_B |               | <a href="#">Section 16.5.7.894</a> |
| 34E8h  | QUEUE_78_STATUS_C |               | <a href="#">Section 16.5.7.895</a> |
| 34F0h  | QUEUE_79_STATUS_A |               | <a href="#">Section 16.5.7.896</a> |
| 34F4h  | QUEUE_79_STATUS_B |               | <a href="#">Section 16.5.7.897</a> |
| 34F8h  | QUEUE_79_STATUS_C |               | <a href="#">Section 16.5.7.898</a> |
| 3500h  | QUEUE_80_STATUS_A |               | <a href="#">Section 16.5.7.899</a> |
| 3504h  | QUEUE_80_STATUS_B |               | <a href="#">Section 16.5.7.900</a> |
| 3508h  | QUEUE_80_STATUS_C |               | <a href="#">Section 16.5.7.901</a> |
| 3510h  | QUEUE_81_STATUS_A |               | <a href="#">Section 16.5.7.902</a> |
| 3514h  | QUEUE_81_STATUS_B |               | <a href="#">Section 16.5.7.903</a> |
| 3518h  | QUEUE_81_STATUS_C |               | <a href="#">Section 16.5.7.904</a> |
| 3520h  | QUEUE_82_STATUS_A |               | <a href="#">Section 16.5.7.905</a> |
| 3524h  | QUEUE_82_STATUS_B |               | <a href="#">Section 16.5.7.906</a> |
| 3528h  | QUEUE_82_STATUS_C |               | <a href="#">Section 16.5.7.907</a> |
| 3530h  | QUEUE_83_STATUS_A |               | <a href="#">Section 16.5.7.908</a> |
| 3534h  | QUEUE_83_STATUS_B |               | <a href="#">Section 16.5.7.909</a> |
| 3538h  | QUEUE_83_STATUS_C |               | <a href="#">Section 16.5.7.910</a> |
| 3540h  | QUEUE_84_STATUS_A |               | <a href="#">Section 16.5.7.911</a> |
| 3544h  | QUEUE_84_STATUS_B |               | <a href="#">Section 16.5.7.912</a> |
| 3548h  | QUEUE_84_STATUS_C |               | <a href="#">Section 16.5.7.913</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym            | Register Name | Section            |
|--------|--------------------|---------------|--------------------|
| 3550h  | QUEUE_85_STATUS_A  |               | Section 16.5.7.914 |
| 3554h  | QUEUE_85_STATUS_B  |               | Section 16.5.7.915 |
| 3558h  | QUEUE_85_STATUS_C  |               | Section 16.5.7.916 |
| 3560h  | QUEUE_86_STATUS_A  |               | Section 16.5.7.917 |
| 3564h  | QUEUE_86_STATUS_B  |               | Section 16.5.7.918 |
| 3568h  | QUEUE_86_STATUS_C  |               | Section 16.5.7.919 |
| 3570h  | QUEUE_87_STATUS_A  |               | Section 16.5.7.920 |
| 3574h  | QUEUE_87_STATUS_B  |               | Section 16.5.7.921 |
| 3578h  | QUEUE_87_STATUS_C  |               | Section 16.5.7.922 |
| 3580h  | QUEUE_88_STATUS_A  |               | Section 16.5.7.923 |
| 3584h  | QUEUE_88_STATUS_B  |               | Section 16.5.7.924 |
| 3588h  | QUEUE_88_STATUS_C  |               | Section 16.5.7.925 |
| 3590h  | QUEUE_89_STATUS_A  |               | Section 16.5.7.926 |
| 3594h  | QUEUE_89_STATUS_B  |               | Section 16.5.7.927 |
| 3598h  | QUEUE_89_STATUS_C  |               | Section 16.5.7.928 |
| 35A0h  | QUEUE_90_STATUS_A  |               | Section 16.5.7.929 |
| 35A4h  | QUEUE_90_STATUS_B  |               | Section 16.5.7.930 |
| 35A8h  | QUEUE_90_STATUS_C  |               | Section 16.5.7.931 |
| 35B0h  | QUEUE_91_STATUS_A  |               | Section 16.5.7.932 |
| 35B4h  | QUEUE_91_STATUS_B  |               | Section 16.5.7.933 |
| 35B8h  | QUEUE_91_STATUS_C  |               | Section 16.5.7.934 |
| 35C0h  | QUEUE_92_STATUS_A  |               | Section 16.5.7.935 |
| 35C4h  | QUEUE_92_STATUS_B  |               | Section 16.5.7.936 |
| 35C8h  | QUEUE_92_STATUS_C  |               | Section 16.5.7.937 |
| 35D0h  | QUEUE_93_STATUS_A  |               | Section 16.5.7.938 |
| 35D4h  | QUEUE_93_STATUS_B  |               | Section 16.5.7.939 |
| 35D8h  | QUEUE_93_STATUS_C  |               | Section 16.5.7.940 |
| 35E0h  | QUEUE_94_STATUS_A  |               | Section 16.5.7.941 |
| 35E4h  | QUEUE_94_STATUS_B  |               | Section 16.5.7.942 |
| 35E8h  | QUEUE_94_STATUS_C  |               | Section 16.5.7.943 |
| 35F0h  | QUEUE_95_STATUS_A  |               | Section 16.5.7.944 |
| 35F4h  | QUEUE_95_STATUS_B  |               | Section 16.5.7.945 |
| 35F8h  | QUEUE_95_STATUS_C  |               | Section 16.5.7.946 |
| 3600h  | QUEUE_96_STATUS_A  |               | Section 16.5.7.947 |
| 3604h  | QUEUE_96_STATUS_B  |               | Section 16.5.7.948 |
| 3608h  | QUEUE_96_STATUS_C  |               | Section 16.5.7.949 |
| 3610h  | QUEUE_97_STATUS_A  |               | Section 16.5.7.950 |
| 3614h  | QUEUE_97_STATUS_B  |               | Section 16.5.7.951 |
| 3618h  | QUEUE_97_STATUS_C  |               | Section 16.5.7.952 |
| 3620h  | QUEUE_98_STATUS_A  |               | Section 16.5.7.953 |
| 3624h  | QUEUE_98_STATUS_B  |               | Section 16.5.7.954 |
| 3628h  | QUEUE_98_STATUS_C  |               | Section 16.5.7.955 |
| 3630h  | QUEUE_99_STATUS_A  |               | Section 16.5.7.956 |
| 3634h  | QUEUE_99_STATUS_B  |               | Section 16.5.7.957 |
| 3638h  | QUEUE_99_STATUS_C  |               | Section 16.5.7.958 |
| 3640h  | QUEUE_100_STATUS_A |               | Section 16.5.7.959 |
| 3644h  | QUEUE_100_STATUS_B |               | Section 16.5.7.960 |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym            | Register Name | Section                             |
|--------|--------------------|---------------|-------------------------------------|
| 3648h  | QUEUE_100_STATUS_C |               | <a href="#">Section 16.5.7.961</a>  |
| 3650h  | QUEUE_101_STATUS_A |               | <a href="#">Section 16.5.7.962</a>  |
| 3654h  | QUEUE_101_STATUS_B |               | <a href="#">Section 16.5.7.963</a>  |
| 3658h  | QUEUE_101_STATUS_C |               | <a href="#">Section 16.5.7.964</a>  |
| 3660h  | QUEUE_102_STATUS_A |               | <a href="#">Section 16.5.7.965</a>  |
| 3664h  | QUEUE_102_STATUS_B |               | <a href="#">Section 16.5.7.966</a>  |
| 3668h  | QUEUE_102_STATUS_C |               | <a href="#">Section 16.5.7.967</a>  |
| 3670h  | QUEUE_103_STATUS_A |               | <a href="#">Section 16.5.7.968</a>  |
| 3674h  | QUEUE_103_STATUS_B |               | <a href="#">Section 16.5.7.969</a>  |
| 3678h  | QUEUE_103_STATUS_C |               | <a href="#">Section 16.5.7.970</a>  |
| 3680h  | QUEUE_104_STATUS_A |               | <a href="#">Section 16.5.7.971</a>  |
| 3684h  | QUEUE_104_STATUS_B |               | <a href="#">Section 16.5.7.972</a>  |
| 3688h  | QUEUE_104_STATUS_C |               | <a href="#">Section 16.5.7.973</a>  |
| 3690h  | QUEUE_105_STATUS_A |               | <a href="#">Section 16.5.7.974</a>  |
| 3694h  | QUEUE_105_STATUS_B |               | <a href="#">Section 16.5.7.975</a>  |
| 3698h  | QUEUE_105_STATUS_C |               | <a href="#">Section 16.5.7.976</a>  |
| 36A0h  | QUEUE_106_STATUS_A |               | <a href="#">Section 16.5.7.977</a>  |
| 36A4h  | QUEUE_106_STATUS_B |               | <a href="#">Section 16.5.7.978</a>  |
| 36A8h  | QUEUE_106_STATUS_C |               | <a href="#">Section 16.5.7.979</a>  |
| 36B0h  | QUEUE_107_STATUS_A |               | <a href="#">Section 16.5.7.980</a>  |
| 36B4h  | QUEUE_107_STATUS_B |               | <a href="#">Section 16.5.7.981</a>  |
| 36B8h  | QUEUE_107_STATUS_C |               | <a href="#">Section 16.5.7.982</a>  |
| 36C0h  | QUEUE_108_STATUS_A |               | <a href="#">Section 16.5.7.983</a>  |
| 36C4h  | QUEUE_108_STATUS_B |               | <a href="#">Section 16.5.7.984</a>  |
| 36C8h  | QUEUE_108_STATUS_C |               | <a href="#">Section 16.5.7.985</a>  |
| 36D0h  | QUEUE_109_STATUS_A |               | <a href="#">Section 16.5.7.986</a>  |
| 36D4h  | QUEUE_109_STATUS_B |               | <a href="#">Section 16.5.7.987</a>  |
| 36D8h  | QUEUE_109_STATUS_C |               | <a href="#">Section 16.5.7.988</a>  |
| 36E0h  | QUEUE_110_STATUS_A |               | <a href="#">Section 16.5.7.989</a>  |
| 36E4h  | QUEUE_110_STATUS_B |               | <a href="#">Section 16.5.7.990</a>  |
| 36E8h  | QUEUE_110_STATUS_C |               | <a href="#">Section 16.5.7.991</a>  |
| 36F0h  | QUEUE_111_STATUS_A |               | <a href="#">Section 16.5.7.992</a>  |
| 36F4h  | QUEUE_111_STATUS_B |               | <a href="#">Section 16.5.7.993</a>  |
| 36F8h  | QUEUE_111_STATUS_C |               | <a href="#">Section 16.5.7.994</a>  |
| 3700h  | QUEUE_112_STATUS_A |               | <a href="#">Section 16.5.7.995</a>  |
| 3704h  | QUEUE_112_STATUS_B |               | <a href="#">Section 16.5.7.996</a>  |
| 3708h  | QUEUE_112_STATUS_C |               | <a href="#">Section 16.5.7.997</a>  |
| 3710h  | QUEUE_113_STATUS_A |               | <a href="#">Section 16.5.7.998</a>  |
| 3714h  | QUEUE_113_STATUS_B |               | <a href="#">Section 16.5.7.999</a>  |
| 3718h  | QUEUE_113_STATUS_C |               | <a href="#">Section 16.5.7.1000</a> |
| 3720h  | QUEUE_114_STATUS_A |               | <a href="#">Section 16.5.7.1001</a> |
| 3724h  | QUEUE_114_STATUS_B |               | <a href="#">Section 16.5.7.1002</a> |
| 3728h  | QUEUE_114_STATUS_C |               | <a href="#">Section 16.5.7.1003</a> |
| 3730h  | QUEUE_115_STATUS_A |               | <a href="#">Section 16.5.7.1004</a> |
| 3734h  | QUEUE_115_STATUS_B |               | <a href="#">Section 16.5.7.1005</a> |
| 3738h  | QUEUE_115_STATUS_C |               | <a href="#">Section 16.5.7.1006</a> |
| 3740h  | QUEUE_116_STATUS_A |               | <a href="#">Section 16.5.7.1007</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym            | Register Name | Section                             |
|--------|--------------------|---------------|-------------------------------------|
| 3744h  | QUEUE_116_STATUS_B |               | <a href="#">Section 16.5.7.1008</a> |
| 3748h  | QUEUE_116_STATUS_C |               | <a href="#">Section 16.5.7.1009</a> |
| 3750h  | QUEUE_117_STATUS_A |               | <a href="#">Section 16.5.7.1010</a> |
| 3754h  | QUEUE_117_STATUS_B |               | <a href="#">Section 16.5.7.1011</a> |
| 3758h  | QUEUE_117_STATUS_C |               | <a href="#">Section 16.5.7.1012</a> |
| 3760h  | QUEUE_118_STATUS_A |               | <a href="#">Section 16.5.7.1013</a> |
| 3764h  | QUEUE_118_STATUS_B |               | <a href="#">Section 16.5.7.1014</a> |
| 3768h  | QUEUE_118_STATUS_C |               | <a href="#">Section 16.5.7.1015</a> |
| 3770h  | QUEUE_119_STATUS_A |               | <a href="#">Section 16.5.7.1016</a> |
| 3774h  | QUEUE_119_STATUS_B |               | <a href="#">Section 16.5.7.1017</a> |
| 3778h  | QUEUE_119_STATUS_C |               | <a href="#">Section 16.5.7.1018</a> |
| 3780h  | QUEUE_120_STATUS_A |               | <a href="#">Section 16.5.7.1019</a> |
| 3784h  | QUEUE_120_STATUS_B |               | <a href="#">Section 16.5.7.1020</a> |
| 3788h  | QUEUE_120_STATUS_C |               | <a href="#">Section 16.5.7.1021</a> |
| 3790h  | QUEUE_121_STATUS_A |               | <a href="#">Section 16.5.7.1022</a> |
| 3794h  | QUEUE_121_STATUS_B |               | <a href="#">Section 16.5.7.1023</a> |
| 3798h  | QUEUE_121_STATUS_C |               | <a href="#">Section 16.5.7.1024</a> |
| 37A0h  | QUEUE_122_STATUS_A |               | <a href="#">Section 16.5.7.1025</a> |
| 37A4h  | QUEUE_122_STATUS_B |               | <a href="#">Section 16.5.7.1026</a> |
| 37A8h  | QUEUE_122_STATUS_C |               | <a href="#">Section 16.5.7.1027</a> |
| 37B0h  | QUEUE_123_STATUS_A |               | <a href="#">Section 16.5.7.1028</a> |
| 37B4h  | QUEUE_123_STATUS_B |               | <a href="#">Section 16.5.7.1029</a> |
| 37B8h  | QUEUE_123_STATUS_C |               | <a href="#">Section 16.5.7.1030</a> |
| 37C0h  | QUEUE_124_STATUS_A |               | <a href="#">Section 16.5.7.1031</a> |
| 37C4h  | QUEUE_124_STATUS_B |               | <a href="#">Section 16.5.7.1032</a> |
| 37C8h  | QUEUE_124_STATUS_C |               | <a href="#">Section 16.5.7.1033</a> |
| 37D0h  | QUEUE_125_STATUS_A |               | <a href="#">Section 16.5.7.1034</a> |
| 37D4h  | QUEUE_125_STATUS_B |               | <a href="#">Section 16.5.7.1035</a> |
| 37D8h  | QUEUE_125_STATUS_C |               | <a href="#">Section 16.5.7.1036</a> |
| 37E0h  | QUEUE_126_STATUS_A |               | <a href="#">Section 16.5.7.1037</a> |
| 37E4h  | QUEUE_126_STATUS_B |               | <a href="#">Section 16.5.7.1038</a> |
| 37E8h  | QUEUE_126_STATUS_C |               | <a href="#">Section 16.5.7.1039</a> |
| 37F0h  | QUEUE_127_STATUS_A |               | <a href="#">Section 16.5.7.1040</a> |
| 37F4h  | QUEUE_127_STATUS_B |               | <a href="#">Section 16.5.7.1041</a> |
| 37F8h  | QUEUE_127_STATUS_C |               | <a href="#">Section 16.5.7.1042</a> |
| 3800h  | QUEUE_128_STATUS_A |               | <a href="#">Section 16.5.7.1043</a> |
| 3804h  | QUEUE_128_STATUS_B |               | <a href="#">Section 16.5.7.1044</a> |
| 3808h  | QUEUE_128_STATUS_C |               | <a href="#">Section 16.5.7.1045</a> |
| 3810h  | QUEUE_129_STATUS_A |               | <a href="#">Section 16.5.7.1046</a> |
| 3814h  | QUEUE_129_STATUS_B |               | <a href="#">Section 16.5.7.1047</a> |
| 3818h  | QUEUE_129_STATUS_C |               | <a href="#">Section 16.5.7.1048</a> |
| 3820h  | QUEUE_130_STATUS_A |               | <a href="#">Section 16.5.7.1049</a> |
| 3824h  | QUEUE_130_STATUS_B |               | <a href="#">Section 16.5.7.1050</a> |
| 3828h  | QUEUE_130_STATUS_C |               | <a href="#">Section 16.5.7.1051</a> |
| 3830h  | QUEUE_131_STATUS_A |               | <a href="#">Section 16.5.7.1052</a> |
| 3834h  | QUEUE_131_STATUS_B |               | <a href="#">Section 16.5.7.1053</a> |
| 3838h  | QUEUE_131_STATUS_C |               | <a href="#">Section 16.5.7.1054</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

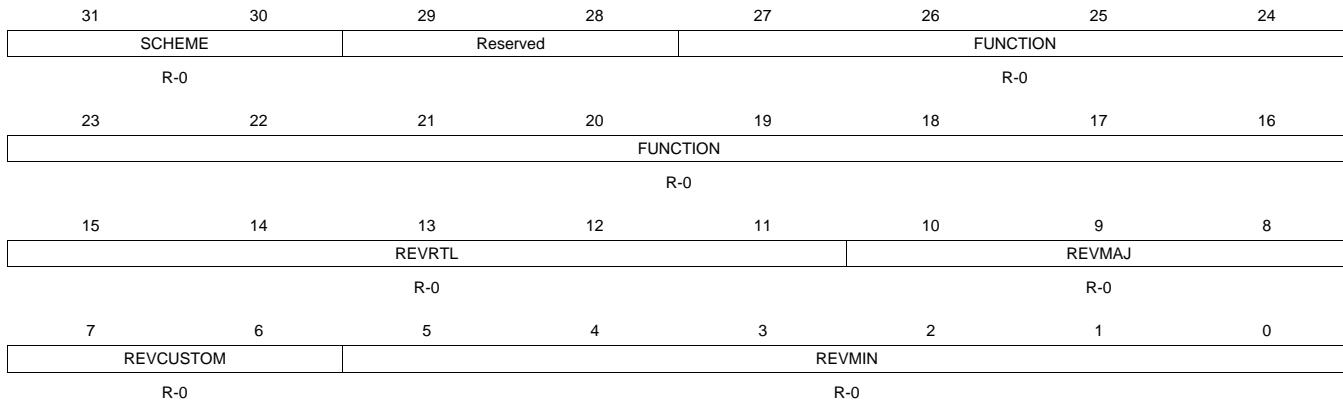
| Offset | Acronym            | Register Name | Section                             |
|--------|--------------------|---------------|-------------------------------------|
| 3840h  | QUEUE_132_STATUS_A |               | <a href="#">Section 16.5.7.1055</a> |
| 3844h  | QUEUE_132_STATUS_B |               | <a href="#">Section 16.5.7.1056</a> |
| 3848h  | QUEUE_132_STATUS_C |               | <a href="#">Section 16.5.7.1057</a> |
| 3850h  | QUEUE_133_STATUS_A |               | <a href="#">Section 16.5.7.1058</a> |
| 3854h  | QUEUE_133_STATUS_B |               | <a href="#">Section 16.5.7.1059</a> |
| 3858h  | QUEUE_133_STATUS_C |               | <a href="#">Section 16.5.7.1060</a> |
| 3860h  | QUEUE_134_STATUS_A |               | <a href="#">Section 16.5.7.1061</a> |
| 3864h  | QUEUE_134_STATUS_B |               | <a href="#">Section 16.5.7.1062</a> |
| 3868h  | QUEUE_134_STATUS_C |               | <a href="#">Section 16.5.7.1063</a> |
| 3870h  | QUEUE_135_STATUS_A |               | <a href="#">Section 16.5.7.1064</a> |
| 3874h  | QUEUE_135_STATUS_B |               | <a href="#">Section 16.5.7.1065</a> |
| 3878h  | QUEUE_135_STATUS_C |               | <a href="#">Section 16.5.7.1066</a> |
| 3880h  | QUEUE_136_STATUS_A |               | <a href="#">Section 16.5.7.1067</a> |
| 3884h  | QUEUE_136_STATUS_B |               | <a href="#">Section 16.5.7.1068</a> |
| 3888h  | QUEUE_136_STATUS_C |               | <a href="#">Section 16.5.7.1069</a> |
| 3890h  | QUEUE_137_STATUS_A |               | <a href="#">Section 16.5.7.1070</a> |
| 3894h  | QUEUE_137_STATUS_B |               | <a href="#">Section 16.5.7.1071</a> |
| 3898h  | QUEUE_137_STATUS_C |               | <a href="#">Section 16.5.7.1072</a> |
| 38A0h  | QUEUE_138_STATUS_A |               | <a href="#">Section 16.5.7.1073</a> |
| 38A4h  | QUEUE_138_STATUS_B |               | <a href="#">Section 16.5.7.1074</a> |
| 38A8h  | QUEUE_138_STATUS_C |               | <a href="#">Section 16.5.7.1075</a> |
| 38B0h  | QUEUE_139_STATUS_A |               | <a href="#">Section 16.5.7.1076</a> |
| 38B4h  | QUEUE_139_STATUS_B |               | <a href="#">Section 16.5.7.1077</a> |
| 38B8h  | QUEUE_139_STATUS_C |               | <a href="#">Section 16.5.7.1078</a> |
| 38C0h  | QUEUE_140_STATUS_A |               | <a href="#">Section 16.5.7.1079</a> |
| 38C4h  | QUEUE_140_STATUS_B |               | <a href="#">Section 16.5.7.1080</a> |
| 38C8h  | QUEUE_140_STATUS_C |               | <a href="#">Section 16.5.7.1081</a> |
| 38D0h  | QUEUE_141_STATUS_A |               | <a href="#">Section 16.5.7.1082</a> |
| 38D4h  | QUEUE_141_STATUS_B |               | <a href="#">Section 16.5.7.1083</a> |
| 38D8h  | QUEUE_141_STATUS_C |               | <a href="#">Section 16.5.7.1084</a> |
| 38E0h  | QUEUE_142_STATUS_A |               | <a href="#">Section 16.5.7.1085</a> |
| 38E4h  | QUEUE_142_STATUS_B |               | <a href="#">Section 16.5.7.1086</a> |
| 38E8h  | QUEUE_142_STATUS_C |               | <a href="#">Section 16.5.7.1087</a> |
| 38F0h  | QUEUE_143_STATUS_A |               | <a href="#">Section 16.5.7.1088</a> |
| 38F4h  | QUEUE_143_STATUS_B |               | <a href="#">Section 16.5.7.1089</a> |
| 38F8h  | QUEUE_143_STATUS_C |               | <a href="#">Section 16.5.7.1090</a> |
| 3900h  | QUEUE_144_STATUS_A |               | <a href="#">Section 16.5.7.1091</a> |
| 3904h  | QUEUE_144_STATUS_B |               | <a href="#">Section 16.5.7.1092</a> |
| 3908h  | QUEUE_144_STATUS_C |               | <a href="#">Section 16.5.7.1093</a> |
| 3910h  | QUEUE_145_STATUS_A |               | <a href="#">Section 16.5.7.1094</a> |
| 3914h  | QUEUE_145_STATUS_B |               | <a href="#">Section 16.5.7.1095</a> |
| 3918h  | QUEUE_145_STATUS_C |               | <a href="#">Section 16.5.7.1096</a> |
| 3920h  | QUEUE_146_STATUS_A |               | <a href="#">Section 16.5.7.1097</a> |
| 3924h  | QUEUE_146_STATUS_B |               | <a href="#">Section 16.5.7.1098</a> |
| 3928h  | QUEUE_146_STATUS_C |               | <a href="#">Section 16.5.7.1099</a> |
| 3930h  | QUEUE_147_STATUS_A |               | <a href="#">Section 16.5.7.1100</a> |
| 3934h  | QUEUE_147_STATUS_B |               | <a href="#">Section 16.5.7.1101</a> |

**Table 16-290. QUEUE\_MGR REGISTERS (continued)**

| Offset | Acronym            | Register Name | Section                             |
|--------|--------------------|---------------|-------------------------------------|
| 3938h  | QUEUE_147_STATUS_C |               | <a href="#">Section 16.5.7.1102</a> |
| 3940h  | QUEUE_148_STATUS_A |               | <a href="#">Section 16.5.7.1103</a> |
| 3944h  | QUEUE_148_STATUS_B |               | <a href="#">Section 16.5.7.1104</a> |
| 3948h  | QUEUE_148_STATUS_C |               | <a href="#">Section 16.5.7.1105</a> |
| 3950h  | QUEUE_149_STATUS_A |               | <a href="#">Section 16.5.7.1106</a> |
| 3954h  | QUEUE_149_STATUS_B |               | <a href="#">Section 16.5.7.1107</a> |
| 3958h  | QUEUE_149_STATUS_C |               | <a href="#">Section 16.5.7.1108</a> |
| 3960h  | QUEUE_150_STATUS_A |               | <a href="#">Section 16.5.7.1109</a> |
| 3964h  | QUEUE_150_STATUS_B |               | <a href="#">Section 16.5.7.1110</a> |
| 3968h  | QUEUE_150_STATUS_C |               | <a href="#">Section 16.5.7.1111</a> |
| 3970h  | QUEUE_151_STATUS_A |               | <a href="#">Section 16.5.7.1112</a> |
| 3974h  | QUEUE_151_STATUS_B |               | <a href="#">Section 16.5.7.1113</a> |
| 3978h  | QUEUE_151_STATUS_C |               | <a href="#">Section 16.5.7.1114</a> |
| 3980h  | QUEUE_152_STATUS_A |               | <a href="#">Section 16.5.7.1115</a> |
| 3984h  | QUEUE_152_STATUS_B |               | <a href="#">Section 16.5.7.1116</a> |
| 3988h  | QUEUE_152_STATUS_C |               | <a href="#">Section 16.5.7.1117</a> |
| 3990h  | QUEUE_153_STATUS_A |               | <a href="#">Section 16.5.7.1118</a> |
| 3994h  | QUEUE_153_STATUS_B |               | <a href="#">Section 16.5.7.1119</a> |
| 3998h  | QUEUE_153_STATUS_C |               | <a href="#">Section 16.5.7.1120</a> |
| 39A0h  | QUEUE_154_STATUS_A |               | <a href="#">Section 16.5.7.1121</a> |
| 39A4h  | QUEUE_154_STATUS_B |               | <a href="#">Section 16.5.7.1122</a> |
| 39A8h  | QUEUE_154_STATUS_C |               | <a href="#">Section 16.5.7.1123</a> |
| 39B0h  | QUEUE_155_STATUS_A |               | <a href="#">Section 16.5.7.1124</a> |
| 39B4h  | QUEUE_155_STATUS_B |               | <a href="#">Section 16.5.7.1125</a> |
| 39B8h  | QUEUE_155_STATUS_C |               | <a href="#">Section 16.5.7.1126</a> |

**16.5.7.1 QMGRREVID Register (offset = 0h) [reset = 4E530800h]**

QMGRREVID is shown in [Figure 16-277](#) and described in [Table 16-291](#).

**Figure 16-277. QMGRREVID Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

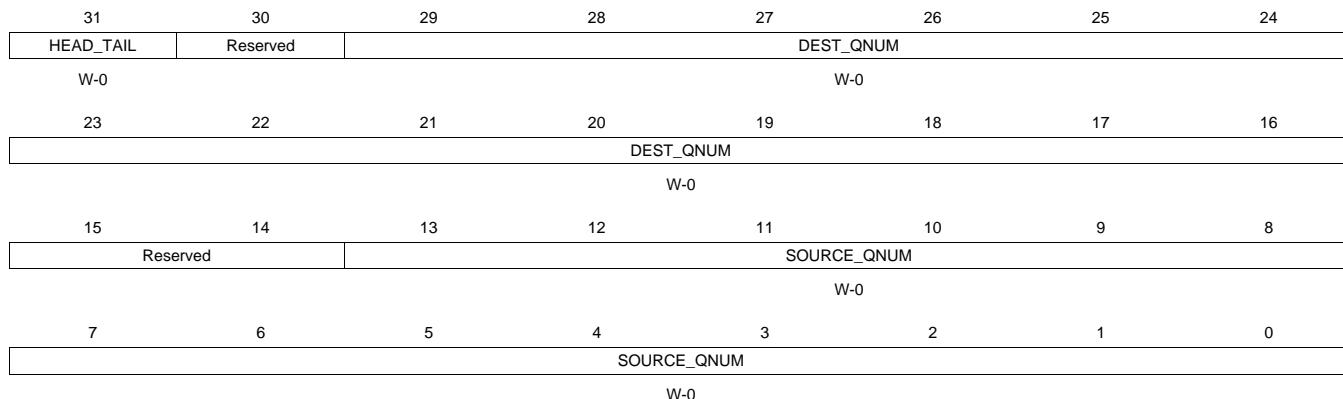
**Table 16-291. QMGRREVID Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description                                    |
|-------|-----------|------|-------|--|
| 31-30 | SCHEME    | R-0  | 0     | Scheme that this register is compliant with    |
| 27-16 | FUNCTION  | R-0  | 0     | Function                                       |
| 15-11 | REVRTL    | R-0  | 0     | RTL revision                                   |
| 10-8  | REVMAJ    | R-0  | 0     | Major revision                                 |
| 7-6   | REVCUSTOM | R-0  | 0     | Custom revision                                |
| 5-0   | REVMIN    | R-0  | 0     | Minor revision Queue Manager Revision Register |

### 16.5.7.2 QMGRRST Register (offset = 8h) [reset = 0h]

QMGRRST is shown in [Figure 16-278](#) and described in [Table 16-292](#).

**Figure 16-278. QMGRRST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

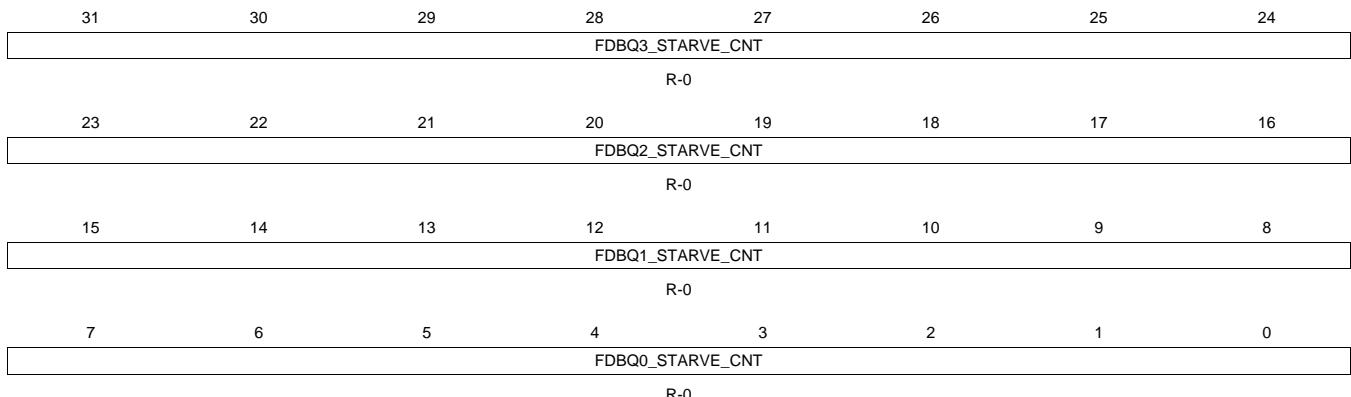
**Table 16-292. QMGRRST Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31    | HEAD_TAIL   | W-0  | 0     | Indicates whether queue contents should be merged on to head or tail of destination queue.<br>Clear this field for head and set for tail.   |
| 29-16 | DEST_QNUM   | W-0  | 0     | Destination Queue Number  |
| 13-0  | SOURCE_QNUM | W-0  | 0     | Source Queue Number Queue Manager Queue Diversion Register<br>Note : CBA write transactions to this register cause the QMGR to start processing an internal state machine.<br>This disables CBA read transactions while it is busy processing the state machine.<br>Once the state machine is back in the idle state CBA read transactions are available again. |

### 16.5.7.3 FDBSC0 Register (offset = 20h) [reset = 0h]

FDBSC0 is shown in [Figure 16-279](#) and described in [Table 16-293](#).

**Figure 16-279. FDBSC0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

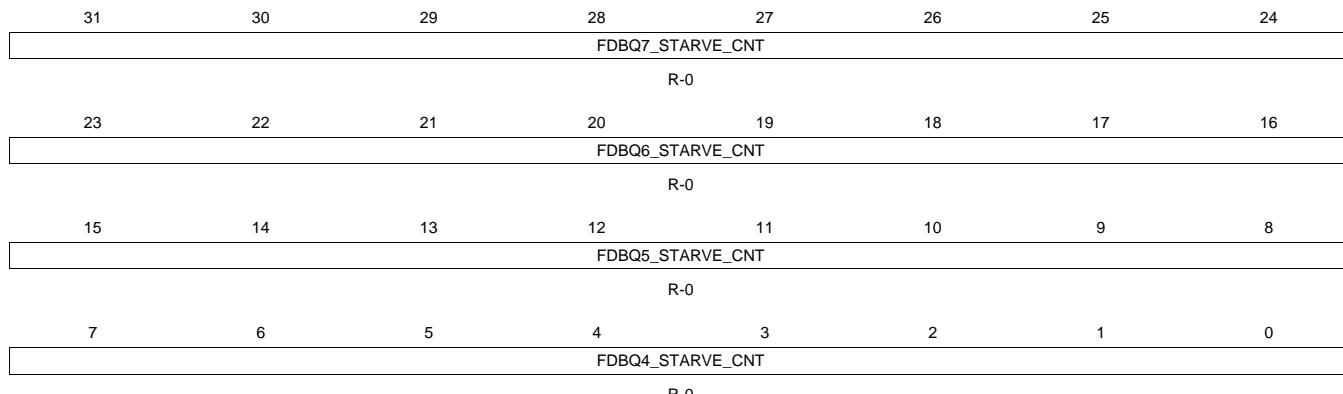
**Table 16-293. FDBSC0 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | FDBQ3_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 3 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ2_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 2 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ1_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 1 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ0_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 0 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 0 |

#### 16.5.7.4 FDBSC1 Register (offset = 24h) [reset = 0h]

FDBSC1 is shown in [Figure 16-280](#) and described in [Table 16-294](#).

**Figure 16-280. FDBSC1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

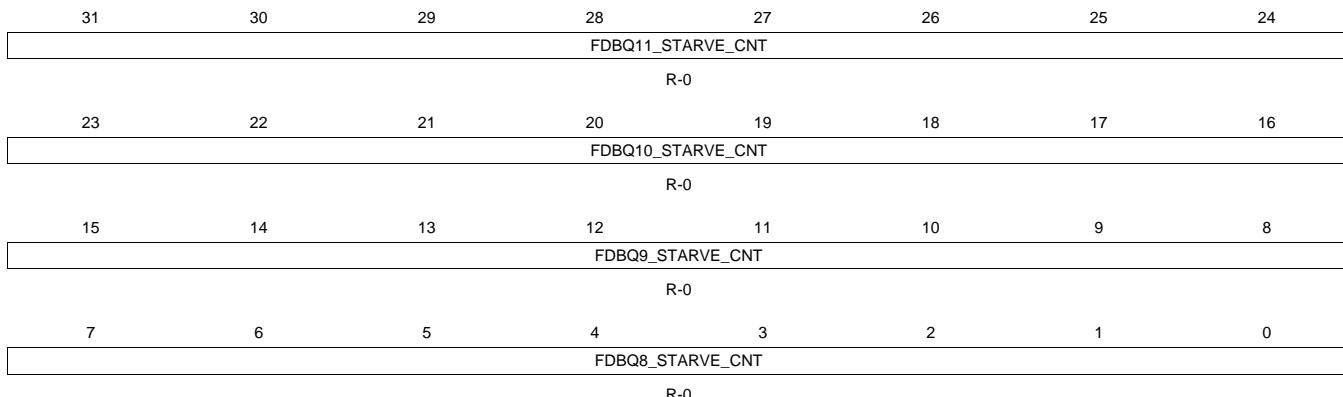
**Table 16-294. FDBSC1 Register Field Descriptions**

| Bit   | Field            | Type | Reset | Description  |
|-------|------------------|------|-------|--|
| 31-24 | FDBQ7_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 7 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ6_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 6 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ5_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 5 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ4_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 4 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 1 |

### 16.5.7.5 FDBSC2 Register (offset = 28h) [reset = 0h]

FDBSC2 is shown in [Figure 16-281](#) and described in [Table 16-295](#).

**Figure 16-281. FDBSC2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

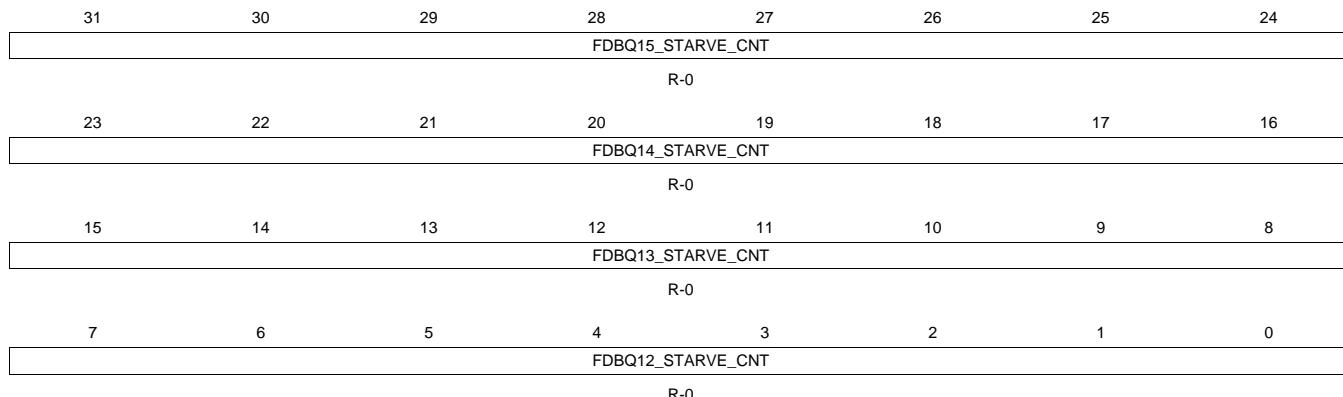
**Table 16-295. FDBSC2 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 31-24 | FDBQ11_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 11 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.  |
| 23-16 | FDBQ10_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 10 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.  |
| 15-8  | FDBQ9_STARVE_CNT  | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 9 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ8_STARVE_CNT  | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 8 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 2 |

### 16.5.7.6 FDBSC3 Register (offset = 2Ch) [reset = 0h]

FDBSC3 is shown in [Figure 16-282](#) and described in [Table 16-296](#).

**Figure 16-282. FDBSC3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

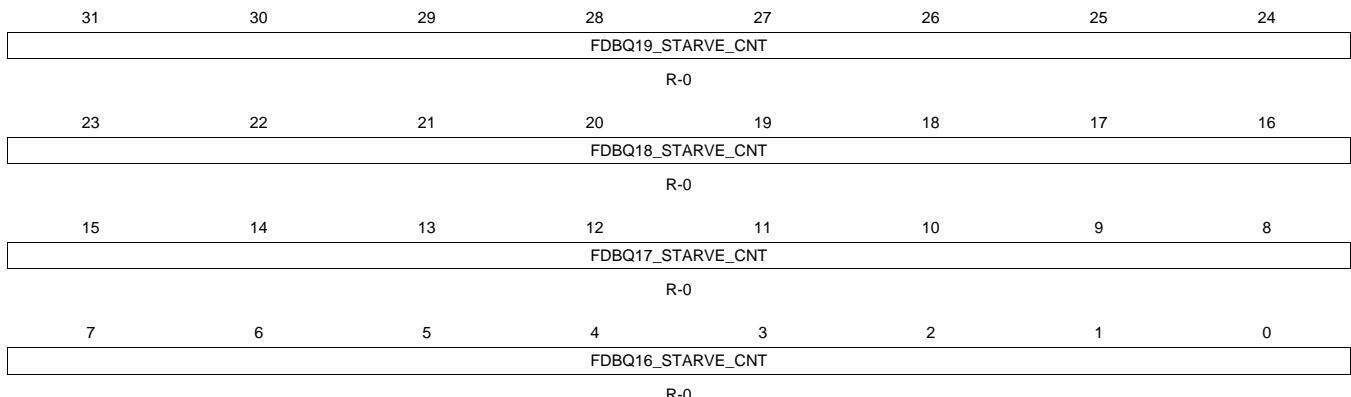
**Table 16-296. FDBSC3 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FDBQ15_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 15 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ14_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 14 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ13_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 13 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ12_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 12 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Table<br>109 - QMGR_Free_Descriptor_Buffer_Starvation_Count Register 3 |

### 16.5.7.7 FDBSC4 Register (offset = 30h) [reset = 0h]

FDBSC4 is shown in [Figure 16-283](#) and described in [Table 16-297](#).

**Figure 16-283. FDBSC4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

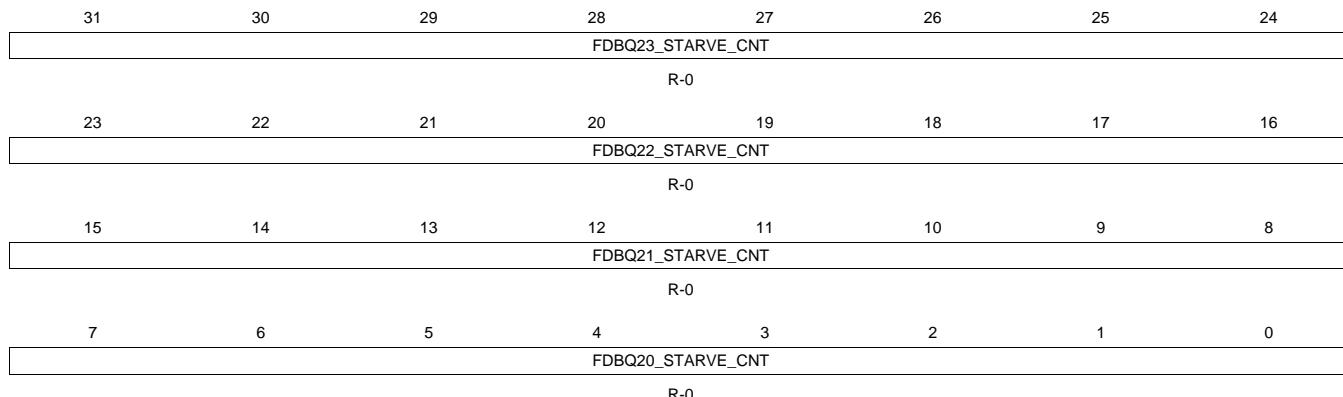
**Table 16-297. FDBSC4 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FDBQ19_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 19 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ18_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 18 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ17_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 17 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ16_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 16 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 4 |

### 16.5.7.8 FDBSC5 Register (offset = 34h) [reset = 0h]

FDBSC5 is shown in [Figure 16-284](#) and described in [Table 16-298](#).

**Figure 16-284. FDBSC5 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

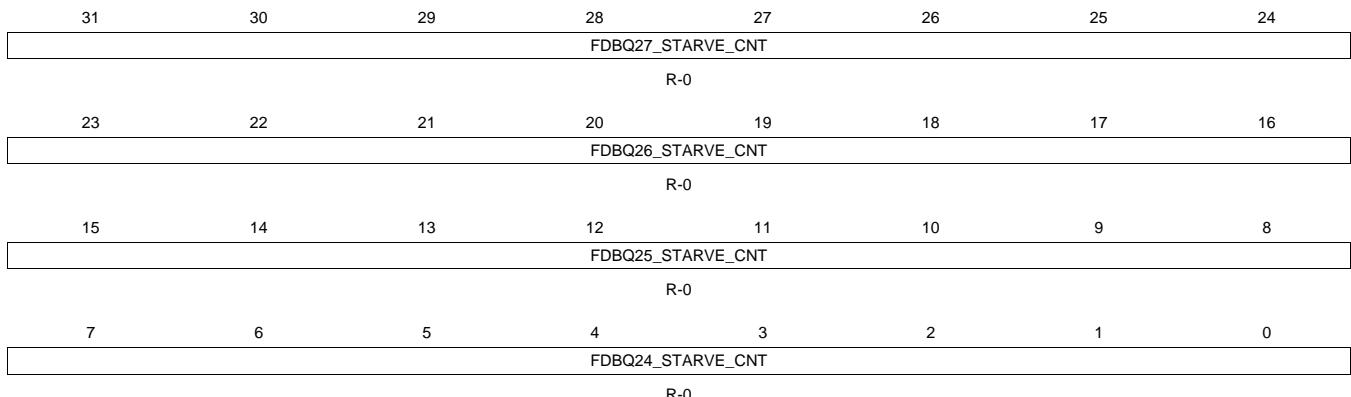
**Table 16-298. FDBSC5 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FDBQ23_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 23 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ22_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 22 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ21_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 21 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ20_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 20 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 5 |

### 16.5.7.9 FDBSC6 Register (offset = 38h) [reset = 0h]

FDBSC6 is shown in [Figure 16-285](#) and described in [Table 16-299](#).

**Figure 16-285. FDBSC6 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

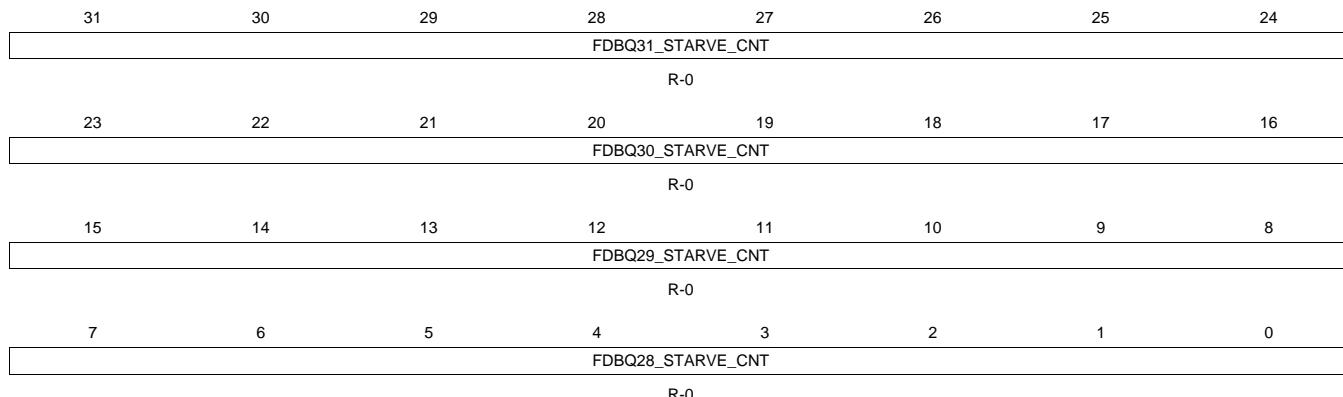
**Table 16-299. FDBSC6 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FDBQ27_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 27 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ26_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 26 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ25_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 25 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ24_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 24 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 6 |

### 16.5.7.10 FDBSC7 Register (offset = 3Ch) [reset = 0h]

FDBSC7 is shown in [Figure 16-286](#) and described in [Table 16-300](#).

**Figure 16-286. FDBSC7 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-300. FDBSC7 Register Field Descriptions**

| Bit   | Field             | Type | Reset | Description   |
|-------|-------------------|------|-------|---|
| 31-24 | FDBQ31_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 31 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 23-16 | FDBQ30_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 30 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 15-8  | FDBQ29_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 29 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.   |
| 7-0   | FDBQ28_STARVE_CNT | R-0  | 0     | This field increments each time the Free Descriptor/Buffer Queue 28 is read while it is empty via the CPPI DMA.<br>This field is cleared when read via the cpu.<br>Queue_Manager_Free_Descriptor_Buffer_Starvation_Count Register 7 |

### 16.5.7.11 LRAM0BASE Register (offset = 80h) [reset = 0h]

LRAM0BASE is shown in [Figure 16-287](#) and described in [Table 16-301](#).

**Figure 16-287. LRAM0BASE Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--|
| REGION0_BASE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |  |
| R/W-0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-301. LRAM0BASE Register Field Descriptions**

| Bit  | Field        | Type  | Reset | Description   |
|------|--------------|-------|-------|---|
| 31-2 | REGION0_BASE | R/W-0 | 0     | This field stores the base address for the first region of the linking RAM.<br>This may be anywhere in 32-bit address space but would be typically located in on-chip memory. |

**16.5.7.12 LRAM0SIZE Register (offset = 84h) [reset = 0h]**

 LRAM0SIZE is shown in [Figure 16-288](#) and described in [Table 16-302](#).

**Figure 16-288. LRAM0SIZE Register**

| 31   | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16           | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0 |
|--|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-------|---|
| Reserved   |    |    |    |    |    |    |    |    |    |    |    |    |    |    | REGION0_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R/W-0 |   |
| LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset |    |    |    |    |    |    |    |    |    |    |    |    |    |    |              |    |    |    |    |    |    |   |   |   |   |   |   |   |   |       |   |

**Table 16-302. LRAM0SIZE Register Field Descriptions**

| Bit  | Field        | Type  | Reset | Description  |
|------|--------------|-------|-------|--|
| 13-0 | REGION0_SIZE | R/W-0 | 0     | This field indicates the number of entries that are contained in the linking RAM region 0.<br>A descriptor with index less than region0_size value has its linking location in region 0.<br>A descriptor with index greater than region0_size has its linking location in region 1.<br>The queue manager will add the index (left shifted by 2 bits) to the appropriate regionX_base_addr to get the absolute 32-bit address to the linking location for a descriptor.<br>Queue Manager Linking Ram Region 0 Size Register |

### 16.5.7.13 LRAM1BASE Register (offset = 88h) [reset = 0h]

LRAM1BASE is shown in [Figure 16-289](#) and described in [Table 16-303](#).

**Figure 16-289. LRAM1BASE Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--|
| REGION1_BASE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |  |
| R/W-0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |  |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-303. LRAM1BASE Register Field Descriptions**

| Bit  | Field        | Type  | Reset | Description   |
|------|--------------|-------|-------|---|
| 31-2 | REGION1_BASE | R/W-0 | 0     | This field stores the base address for the second region of the linking RAM.<br>This may be anywhere in 32- bit address space but would be typically located in off- chip memory. |

**16.5.7.14 PEND0 Register (offset = 90h) [reset = 0h]**

 PEND0 is shown in [Figure 16-290](#) and described in [Table 16-304](#).

**Figure 16-290. PEND0 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QPEND0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-304. PEND0 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | QPEND0 | R-0  | 0     | This field indicates the queue pending status for queues[31:0]. |

### 16.5.7.15 PEND1 Register (offset = 94h) [reset = 0h]

PEND1 is shown in Figure 16-291 and described in Table 16-305.

**Figure 16-291. PEND1 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QPEND1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-305. PEND1 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | QPEND1 | R-0  | 0     | This field indicates the queue pending status for queues[63:32].<br>Table 118 - QMGR_Queue_Pending_1 Register 1 |

**16.5.7.16 PEND2 Register (offset = 98h) [reset = 0h]**

PEND2 is shown in [Figure 16-292](#) and described in [Table 16-306](#).

**Figure 16-292. PEND2 Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QPEND2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-306. PEND2 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description  |
|------|--------|------|-------|--|
| 31-0 | QPEND2 | R-0  | 0     | This field indicates the queue pending status for queues[95:64].<br>Queue_Manager_Queue_Pending_2 Register 2 |

**16.5.7.17 PEND3 Register (offset = 9Ch) [reset = 0h]**

PEND3 is shown in [Figure 16-293](#) and described in [Table 16-307](#).

**Figure 16-293. PEND3 Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| QPEND3 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-307. PEND3 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description   |
|------|--------|------|-------|---|
| 31-0 | QPEND3 | R-0  | 0     | This field indicates the queue pending status for queues[127:96].<br>Queue_Manager_Queue_Pending_3 Register 3 |

**16.5.7.18 PEND4 Register (offset = A0h) [reset = 0h]**

PEND4 is shown in [Figure 16-294](#) and described in [Table 16-308](#).

**Figure 16-294. PEND4 Register**

| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| QPEND4 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-308. PEND4 Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description  |
|------|--------|------|-------|--|
| 31-0 | QPEND4 | R-0  | 0     | This field indicates the queue pending status for queues[159:128].<br>Queue_Manager_Queue_Pending_4 Register 4 |

### 16.5.7.19 QMEMRBASE0 Register (offset = 1000h) [reset = 0h]

QMEMRBASE0 is shown in [Figure 16-295](#) and described in [Table 16-309](#).

**Figure 16-295. QMEMRBASE0 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-309. QMEMRBASE0 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.20 QMEMCTRL0 Register (offset = 1004h) [reset = 0h]

QMEMCTRL0 is shown in [Figure 16-296](#) and described in [Table 16-310](#).

**Figure 16-296. QMEMCTRL0 Register**

|          |          |    |             |             |          |    |    |
|----------|----------|----|-------------|-------------|----------|----|----|
| 31       | 30       | 29 | 28          | 27          | 26       | 25 | 24 |
| Reserved |          |    |             | START_INDEX |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 23       | 22       | 21 | 20          | 19          | 18       | 17 | 16 |
|          |          |    | START_INDEX |             |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 15       | 14       | 13 | 12          | 11          | 10       | 9  | 8  |
| Reserved |          |    |             | DESC_SIZE   |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 7        | 6        | 5  | 4           | 3           | 2        | 1  | 0  |
|          | Reserved |    |             |             | REG_SIZE |    |    |
|          |          |    |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-310. QMEMCTRL0 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes. The settings of desc_size from 9-15 are reserved.  |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors. Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region. For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index. Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation. The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes. Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

**16.5.7.21 QMEMRBASE1 Register (offset = 1010h) [reset = 0h]**

QMEMRBASE1 is shown in [Figure 16-297](#) and described in [Table 16-311](#).

**Figure 16-297. QMEMRBASE1 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-311. QMEMRBASE1 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.22 QMEMCTRL1 Register (offset = 1014h) [reset = 0h]

QMEMCTRL1 is shown in [Figure 16-298](#) and described in [Table 16-312](#).

**Figure 16-298. QMEMCTRL1 Register**

| 31       | 30       | 29 | 28          | 27          | 26       | 25 | 24 |
|----------|----------|----|-------------|-------------|----------|----|----|
| Reserved |          |    |             | START_INDEX |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 23       | 22       | 21 | 20          | 19          | 18       | 17 | 16 |
|          |          |    | START_INDEX |             |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 15       | 14       | 13 | 12          | 11          | 10       | 9  | 8  |
| Reserved |          |    |             | DESC_SIZE   |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 7        | 6        | 5  | 4           | 3           | 2        | 1  | 0  |
|          | Reserved |    |             |             | REG_SIZE |    |    |
|          |          |    |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-312. QMEMCTRL1 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes. The settings of desc_size from 9-15 are reserved.  |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors. Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region. For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index. Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation. The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes. Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

**16.5.7.23 QMEMRBASE2 Register (offset = 1020h) [reset = 0h]**

QMEMRBASE2 is shown in [Figure 16-299](#) and described in [Table 16-313](#).

**Figure 16-299. QMEMRBASE2 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-313. QMEMRBASE2 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.24 QMEMCTRL2 Register (offset = 1024h) [reset = 0h]

QMEMCTRL2 is shown in [Figure 16-300](#) and described in [Table 16-314](#).

**Figure 16-300. QMEMCTRL2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-314. QMEMCTRL2 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes.<br>The settings of desc_size from 9-15 are reserved.   |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors).<br>It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors.<br>Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region.<br>For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index.<br>Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation.<br>The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes.<br>Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

**16.5.7.25 QMEMRBASE3 Register (offset = 1030h) [reset = 0h]**

QMEMRBASE3 is shown in [Figure 16-301](#) and described in [Table 16-315](#).

**Figure 16-301. QMEMRBASE3 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-315. QMEMRBASE3 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.26 QMEMCTRL3 Register (offset = 1034h) [reset = 0h]

QMEMCTRL3 is shown in [Figure 16-302](#) and described in [Table 16-316](#).

**Figure 16-302. QMEMCTRL3 Register**

|          |    |          |             |             |          |    |    |
|----------|----|----------|-------------|-------------|----------|----|----|
| 31       | 30 | 29       | 28          | 27          | 26       | 25 | 24 |
| Reserved |    |          |             | START_INDEX |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 23       | 22 | 21       | 20          | 19          | 18       | 17 | 16 |
|          |    |          | START_INDEX |             |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 15       | 14 | 13       | 12          | 11          | 10       | 9  | 8  |
| Reserved |    |          |             | DESC_SIZE   |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 7        | 6  | 5        | 4           | 3           | 2        | 1  | 0  |
|          |    | Reserved |             |             | REG_SIZE |    |    |
|          |    |          |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-316. QMEMCTRL3 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes.<br>The settings of desc_size from 9-15 are reserved.   |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors).<br>It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors.<br>Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region.<br>For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index.<br>Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation.<br>The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes.<br>Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

### 16.5.7.27 QMEMRBASE4 Register (offset = 1040h) [reset = 0h]

QMEMRBASE4 is shown in [Figure 16-303](#) and described in [Table 16-317](#).

**Figure 16-303. QMEMRBASE4 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-317. QMEMRBASE4 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.28 QMEMCTRL4 Register (offset = 1044h) [reset = 0h]

QMEMCTRL4 is shown in [Figure 16-304](#) and described in [Table 16-318](#).

**Figure 16-304. QMEMCTRL4 Register**

| 31       | 30       | 29 | 28          | 27          | 26       | 25 | 24 |
|----------|----------|----|-------------|-------------|----------|----|----|
| Reserved |          |    |             | START_INDEX |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 23       | 22       | 21 | 20          | 19          | 18       | 17 | 16 |
|          |          |    | START_INDEX |             |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 15       | 14       | 13 | 12          | 11          | 10       | 9  | 8  |
| Reserved |          |    |             | DESC_SIZE   |          |    |    |
|          |          |    |             | R/W-0       |          |    |    |
| 7        | 6        | 5  | 4           | 3           | 2        | 1  | 0  |
|          | Reserved |    |             |             | REG_SIZE |    |    |
|          |          |    |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-318. QMEMCTRL4 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes. The settings of desc_size from 9-15 are reserved.  |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors. Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region. For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index. Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation. The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes. Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

**16.5.7.29 QMEMRBASE5 Register (offset = 1050h) [reset = 0h]**

QMEMRBASE5 is shown in [Figure 16-305](#) and described in [Table 16-319](#).

**Figure 16-305. QMEMRBASE5 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-319. QMEMRBASE5 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.30 QMEMCTRL5 Register (offset = 1054h) [reset = 0h]

QMEMCTRL5 is shown in [Figure 16-306](#) and described in [Table 16-320](#).

**Figure 16-306. QMEMCTRL5 Register**

|          |    |          |             |             |          |    |    |
|----------|----|----------|-------------|-------------|----------|----|----|
| 31       | 30 | 29       | 28          | 27          | 26       | 25 | 24 |
| Reserved |    |          |             | START_INDEX |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 23       | 22 | 21       | 20          | 19          | 18       | 17 | 16 |
|          |    |          | START_INDEX |             |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 15       | 14 | 13       | 12          | 11          | 10       | 9  | 8  |
| Reserved |    |          |             | DESC_SIZE   |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 7        | 6  | 5        | 4           | 3           | 2        | 1  | 0  |
|          |    | Reserved |             |             | REG_SIZE |    |    |
|          |    |          |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-320. QMEMCTRL5 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes. The settings of desc_size from 9-15 are reserved.  |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors). It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors. Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region. For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index. Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation. The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes. Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

### 16.5.7.31 QMEMRBASE6 Register (offset = 1060h) [reset = 0h]

QMEMRBASE6 is shown in [Figure 16-307](#) and described in [Table 16-321](#).

**Figure 16-307. QMEMRBASE6 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

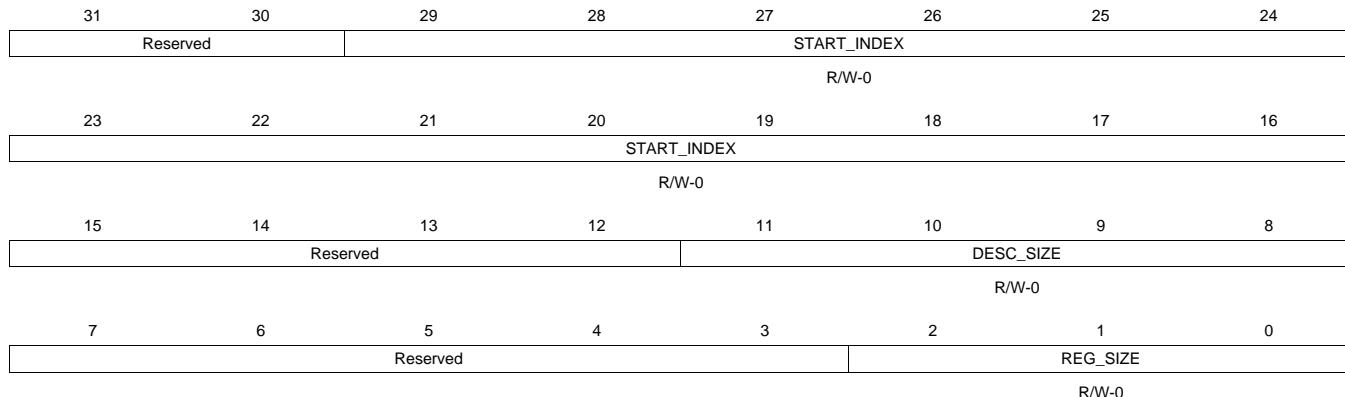
**Table 16-321. QMEMRBASE6 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.32 QMEMCTRL6 Register (offset = 1064h) [reset = 0h]

QMEMCTRL6 is shown in [Figure 16-308](#) and described in [Table 16-322](#).

**Figure 16-308. QMEMCTRL6 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-322. QMEMCTRL6 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes.<br>The settings of desc_size from 9-15 are reserved.   |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors).<br>It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors.<br>Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region.<br>For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index.<br>Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation.<br>The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes.<br>Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

### 16.5.7.33 QMEMRBASE7 Register (offset = 1070h) [reset = 0h]

QMEMRBASE7 is shown in [Figure 16-309](#) and described in [Table 16-323](#).

**Figure 16-309. QMEMRBASE7 Register**

| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5        | 4 | 3 | 2 | 1 | 0 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|----------|---|---|---|---|---|
| REG   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   | Reserved |   |   |   |   |   |
| R/W-0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |          |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-323. QMEMRBASE7 Register Field Descriptions**

| Bit  | Field | Type  | Reset | Description  |
|------|-------|-------|-------|--|
| 31-5 | REG   | R/W-0 | 0     | This field contains the base address of the memory region R. |

### 16.5.7.34 QMEMCTRL7 Register (offset = 1074h) [reset = 0h]

QMEMCTRL7 is shown in [Figure 16-310](#) and described in [Table 16-324](#).

**Figure 16-310. QMEMCTRL7 Register**

|          |    |          |             |             |          |    |    |
|----------|----|----------|-------------|-------------|----------|----|----|
| 31       | 30 | 29       | 28          | 27          | 26       | 25 | 24 |
| Reserved |    |          |             | START_INDEX |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 23       | 22 | 21       | 20          | 19          | 18       | 17 | 16 |
|          |    |          | START_INDEX |             |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 15       | 14 | 13       | 12          | 11          | 10       | 9  | 8  |
| Reserved |    |          |             | DESC_SIZE   |          |    |    |
|          |    |          |             | R/W-0       |          |    |    |
| 7        | 6  | 5        | 4           | 3           | 2        | 1  | 0  |
|          |    | Reserved |             |             | REG_SIZE |    |    |
|          |    |          |             |             | R/W-0    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-324. QMEMCTRL7 Register Field Descriptions**

| Bit   | Field       | Type  | Reset | Description   |
|-------|-------------|-------|-------|---|
| 29-16 | START_INDEX | R/W-0 | 0     | This field indicates where in linking RAM does the descriptor linking information corresponding to memory region R starts.  |
| 11-8  | DESC_SIZE   | R/W-0 | 0     | This field indicates the size of each descriptor in this memory region. It is an encoded value that specifies descriptor size as $2^{(5+desc\_size)}$ number of bytes.<br>The settings of desc_size from 9-15 are reserved.   |
| 2-0   | REG_SIZE    | R/W-0 | 0     | This field indicates the size of the memory region (in terms of number of descriptors).<br>It is an encoded value that specifies region size as $2^{(5+reg\_size)}$ number of descriptors.<br>Queue Manager Memory Region R Control Registers The following sections describe each of the four register locations that may be present for each queue in the queues region.<br>For reasons of implementation and area efficiency, these registers are not actually implemented as a huge array of flip flops but are instead implemented as a single set of mailbox registers which use the LSBs of the provided address as a queue index.<br>Due to this implementation all accesses to these registers need to be performed as a single burst write for each packet push operation or a single burst read for each packet pop operation.<br>The length of a burst to push or pop a packet will vary depending on the optional features that the queue supports which may be 4, 8, 12 or 16 bytes.<br>Queue N Register D must always be written / read in the burst but the preceding words are optional depending on the required queue functionality. |

### 16.5.7.35 QUEUE\_0\_A Register (offset = 2000h) [reset = 0h]

QUEUE\_0\_A is shown in [Figure 16-311](#) and described in [Table 16-325](#).

**Figure 16-311. QUEUE\_0\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-325. QUEUE\_0\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.36 QUEUE\_0\_B Register (offset = 2004h) [reset = 0h]**

QUEUE\_0\_B is shown in [Figure 16-312](#) and described in [Table 16-326](#).

**Figure 16-312. QUEUE\_0\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

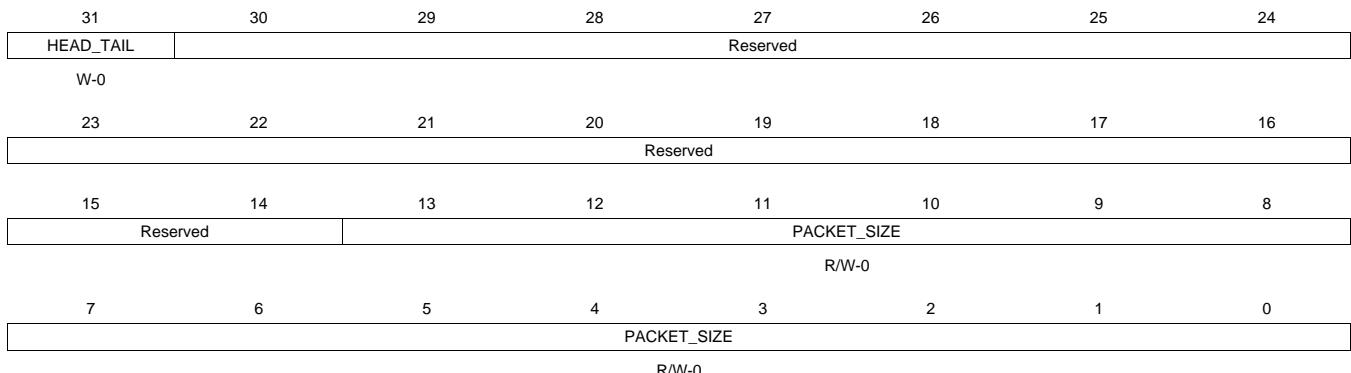
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-326. QUEUE\_0\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.37 QUEUE\_0\_C Register (offset = 2008h) [reset = 0h]**

QUEUE\_0\_C is shown in [Figure 16-313](#) and described in [Table 16-327](#).

**Figure 16-313. QUEUE\_0\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

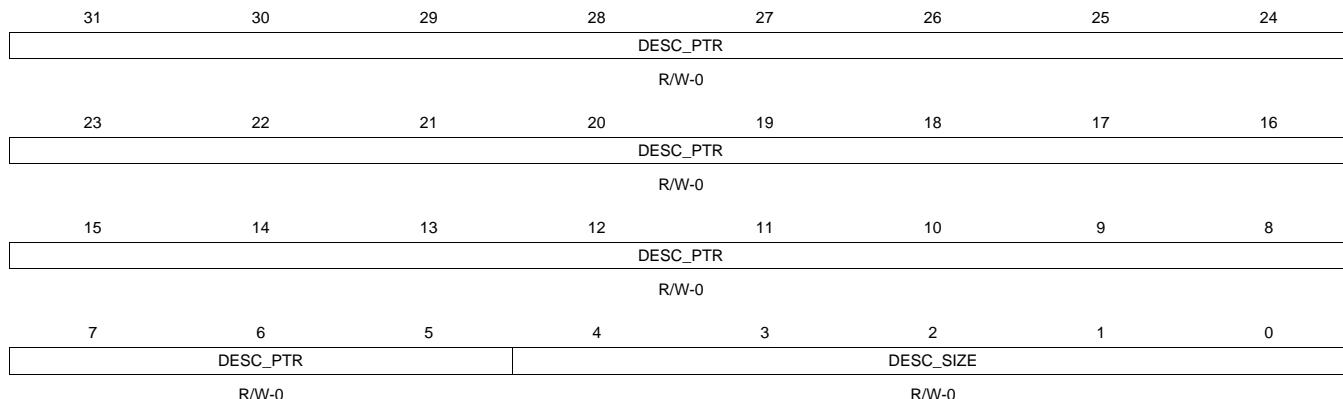
**Table 16-327. QUEUE\_0\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.38 QUEUE\_0\_D Register (offset = 200Ch) [reset = 0h]

QUEUE\_0\_D is shown in [Figure 16-314](#) and described in [Table 16-328](#).

**Figure 16-314. QUEUE\_0\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-328. QUEUE\_0\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.39 QUEUE\_1\_A Register (offset = 2010h) [reset = 0h]**

QUEUE\_1\_A is shown in [Figure 16-315](#) and described in [Table 16-329](#).

**Figure 16-315. QUEUE\_1\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-329. QUEUE\_1\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.40 QUEUE\_1\_B Register (offset = 2014h) [reset = 0h]**

QUEUE\_1\_B is shown in [Figure 16-316](#) and described in [Table 16-330](#).

**Figure 16-316. QUEUE\_1\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

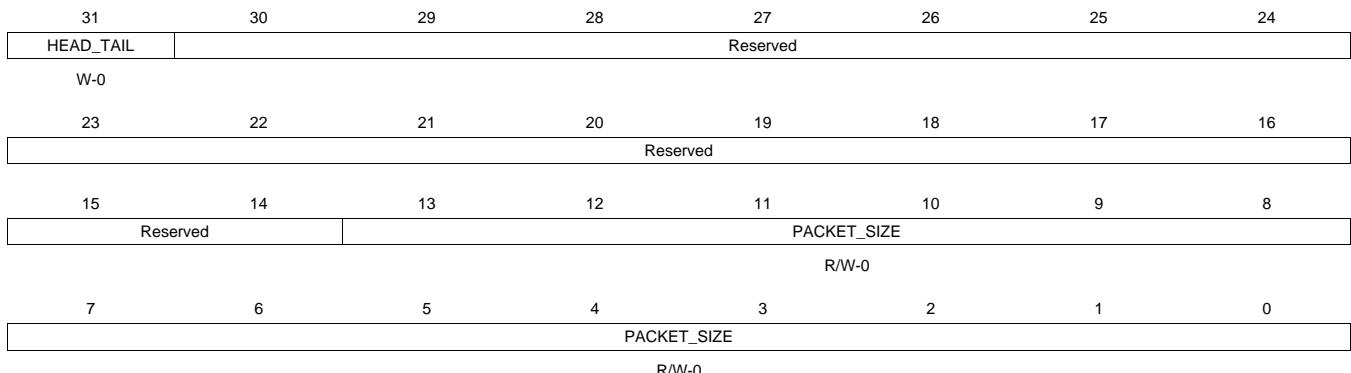
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-330. QUEUE\_1\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.41 QUEUE\_1\_C Register (offset = 2018h) [reset = 0h]**

QUEUE\_1\_C is shown in [Figure 16-317](#) and described in [Table 16-331](#).

**Figure 16-317. QUEUE\_1\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

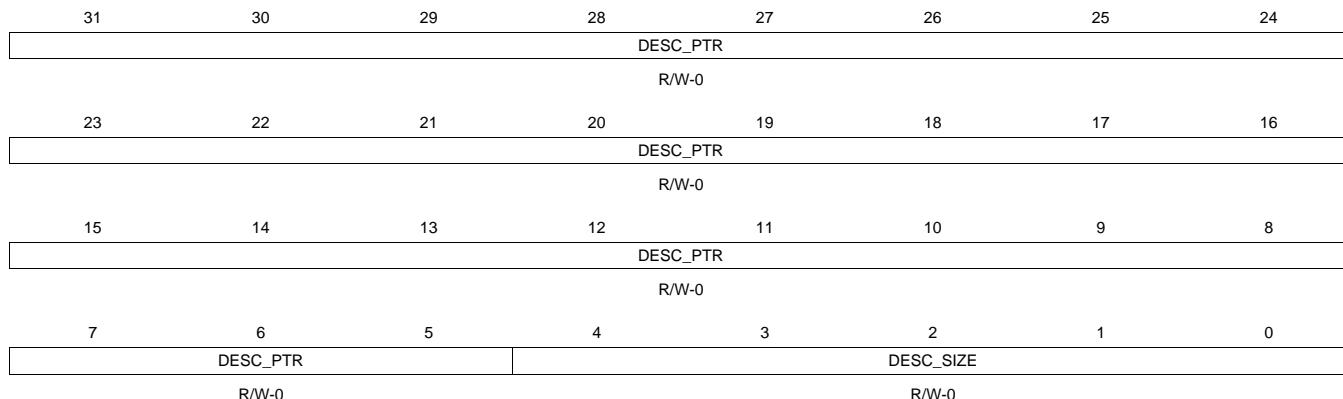
**Table 16-331. QUEUE\_1\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

#### 16.5.7.42 QUEUE\_1\_D Register (offset = 201Ch) [reset = 0h]

QUEUE\_1\_D is shown in [Figure 16-318](#) and described in [Table 16-332](#).

**Figure 16-318. QUEUE\_1\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-332. QUEUE\_1\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.43 QUEUE\_2\_A Register (offset = 2020h) [reset = 0h]**

QUEUE\_2\_A is shown in [Figure 16-319](#) and described in [Table 16-333](#).

**Figure 16-319. QUEUE\_2\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-333. QUEUE\_2\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.44 QUEUE\_2\_B Register (offset = 2024h) [reset = 0h]**

QUEUE\_2\_B is shown in [Figure 16-320](#) and described in [Table 16-334](#).

**Figure 16-320. QUEUE\_2\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

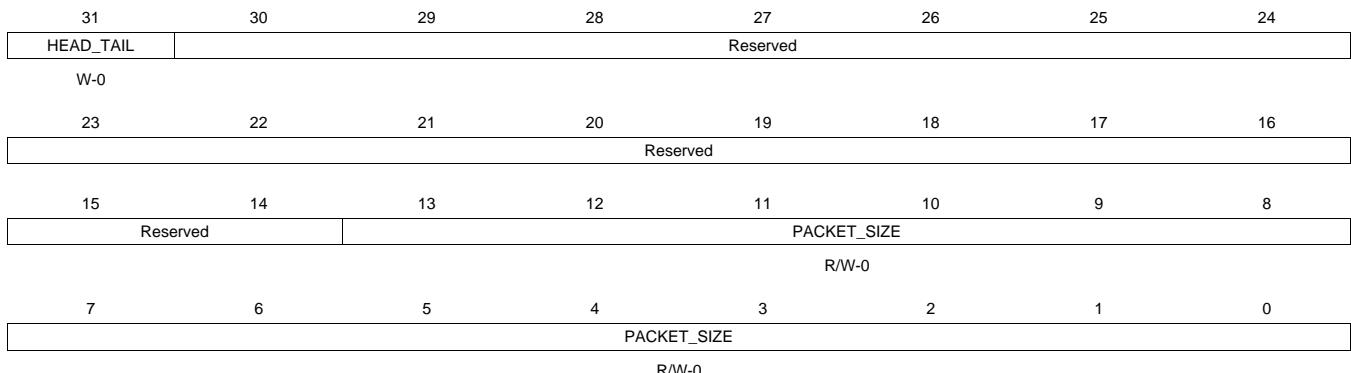
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-334. QUEUE\_2\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.45 QUEUE\_2\_C Register (offset = 2028h) [reset = 0h]**

QUEUE\_2\_C is shown in [Figure 16-321](#) and described in [Table 16-335](#).

**Figure 16-321. QUEUE\_2\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

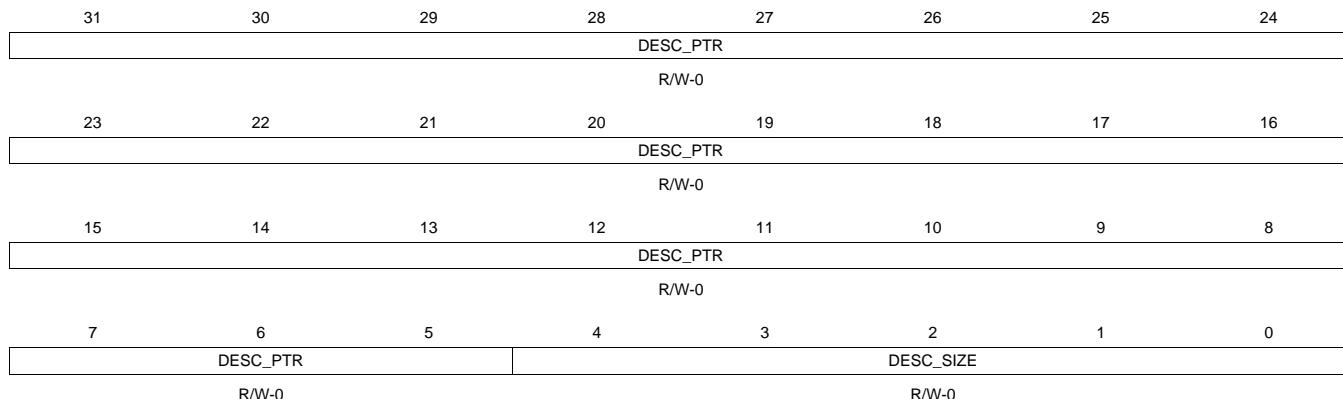
**Table 16-335. QUEUE\_2\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

#### 16.5.7.46 QUEUE\_2\_D Register (offset = 202Ch) [reset = 0h]

QUEUE\_2\_D is shown in [Figure 16-322](#) and described in [Table 16-336](#).

**Figure 16-322. QUEUE\_2\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-336. QUEUE\_2\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.47 QUEUE\_3\_A Register (offset = 2030h) [reset = 0h]**

QUEUE\_3\_A is shown in [Figure 16-323](#) and described in [Table 16-337](#).

**Figure 16-323. QUEUE\_3\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-337. QUEUE\_3\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.48 QUEUE\_3\_B Register (offset = 2034h) [reset = 0h]**

QUEUE\_3\_B is shown in [Figure 16-324](#) and described in [Table 16-338](#).

**Figure 16-324. QUEUE\_3\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

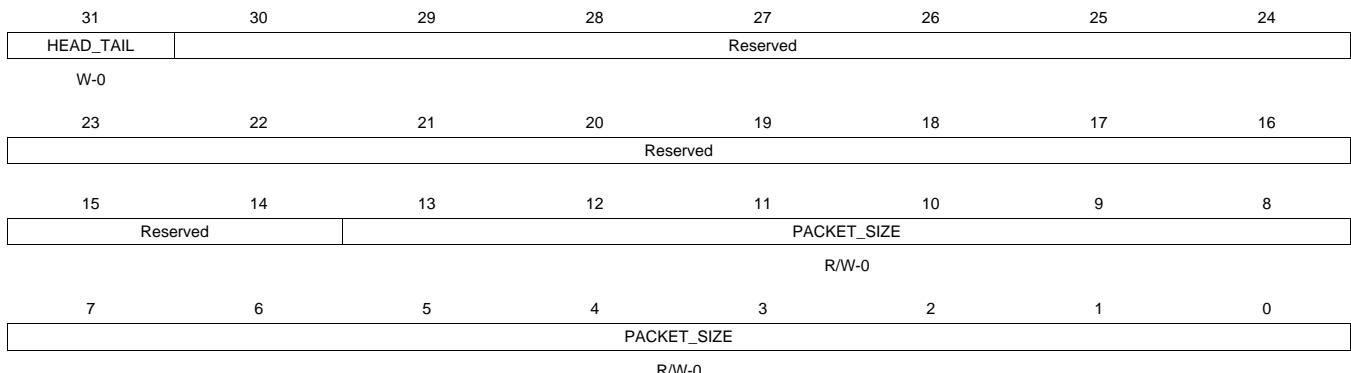
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-338. QUEUE\_3\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.49 QUEUE\_3\_C Register (offset = 2038h) [reset = 0h]**

QUEUE\_3\_C is shown in [Figure 16-325](#) and described in [Table 16-339](#).

**Figure 16-325. QUEUE\_3\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

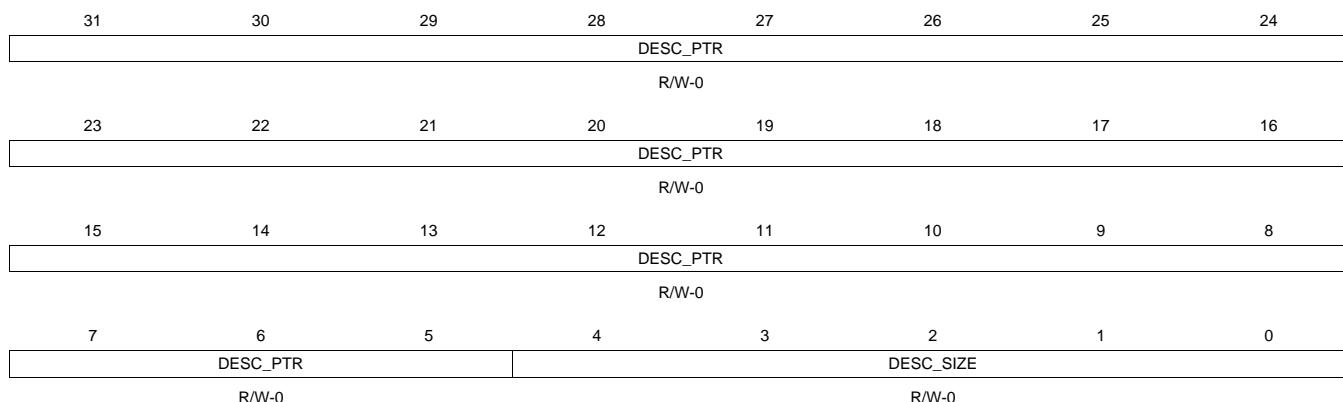
**Table 16-339. QUEUE\_3\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.50 QUEUE\_3\_D Register (offset = 203Ch) [reset = 0h]

QUEUE\_3\_D is shown in [Figure 16-326](#) and described in [Table 16-340](#).

**Figure 16-326. QUEUE\_3\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-340. QUEUE\_3\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.51 QUEUE\_4\_A Register (offset = 2040h) [reset = 0h]

QUEUE\_4\_A is shown in [Figure 16-327](#) and described in [Table 16-341](#).

**Figure 16-327. QUEUE\_4\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-341. QUEUE\_4\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.52 QUEUE\_4\_B Register (offset = 2044h) [reset = 0h]**

QUEUE\_4\_B is shown in [Figure 16-328](#) and described in [Table 16-342](#).

**Figure 16-328. QUEUE\_4\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

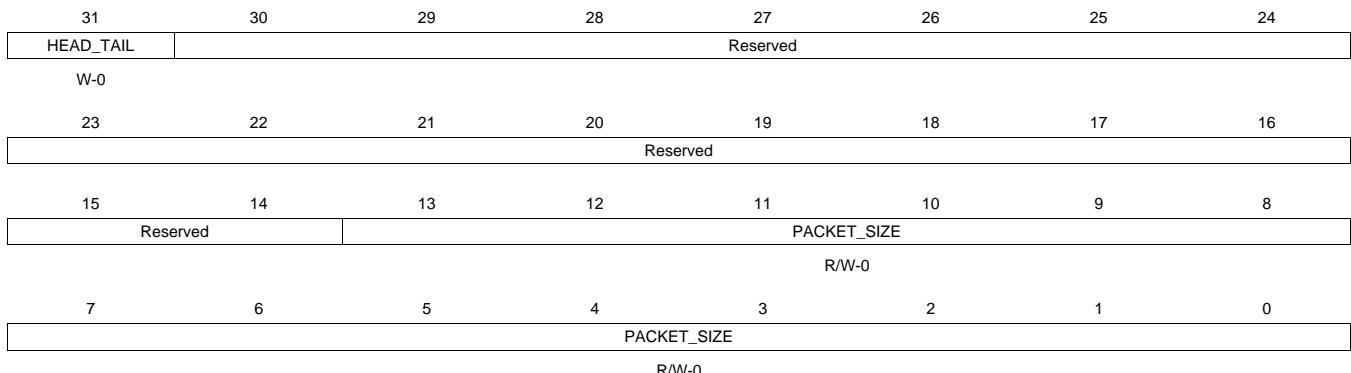
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-342. QUEUE\_4\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.53 QUEUE\_4\_C Register (offset = 2048h) [reset = 0h]**

QUEUE\_4\_C is shown in [Figure 16-329](#) and described in [Table 16-343](#).

**Figure 16-329. QUEUE\_4\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

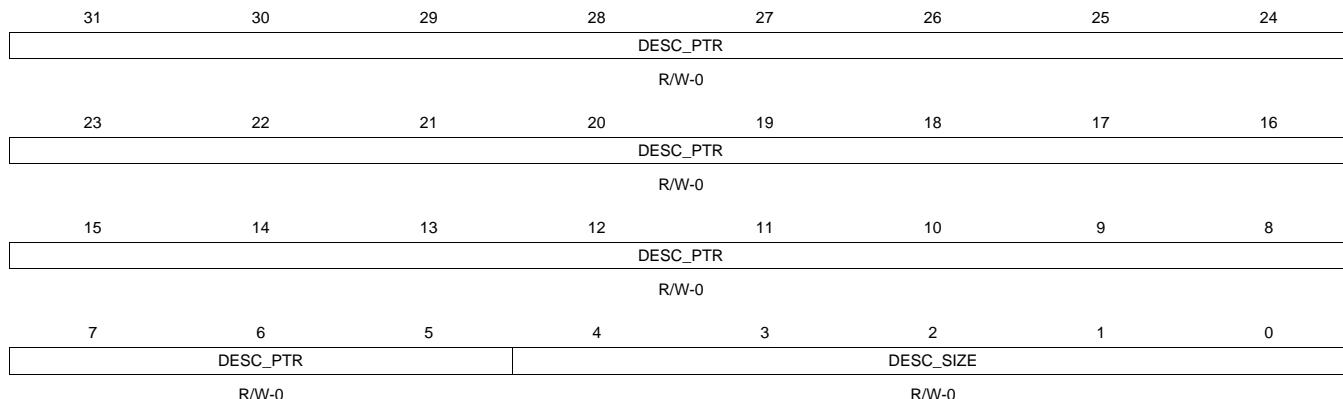
**Table 16-343. QUEUE\_4\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.54 QUEUE\_4\_D Register (offset = 204Ch) [reset = 0h]

QUEUE\_4\_D is shown in [Figure 16-330](#) and described in [Table 16-344](#).

**Figure 16-330. QUEUE\_4\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-344. QUEUE\_4\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.55 QUEUE\_5\_A Register (offset = 2050h) [reset = 0h]**

QUEUE\_5\_A is shown in [Figure 16-331](#) and described in [Table 16-345](#).

**Figure 16-331. QUEUE\_5\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-345. QUEUE\_5\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.56 QUEUE\_5\_B Register (offset = 2054h) [reset = 0h]**

QUEUE\_5\_B is shown in [Figure 16-332](#) and described in [Table 16-346](#).

**Figure 16-332. QUEUE\_5\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

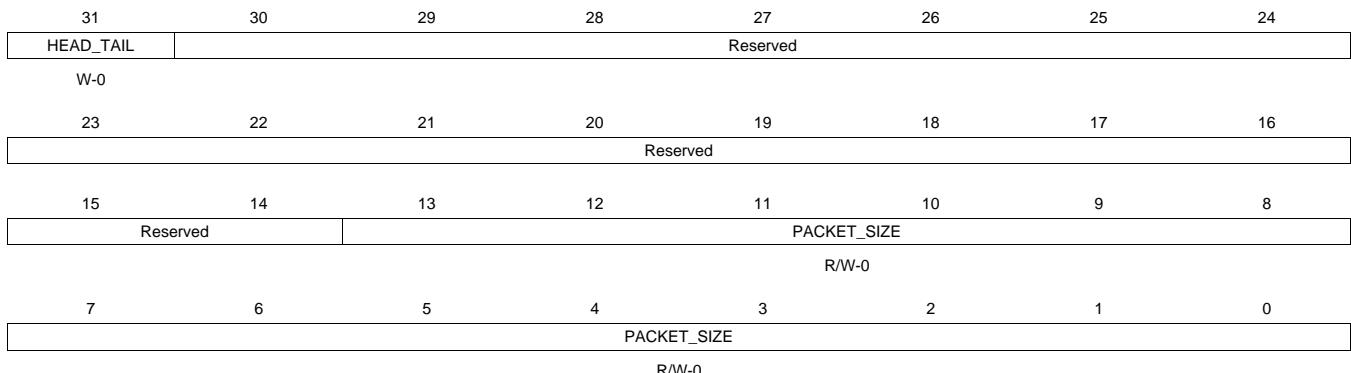
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-346. QUEUE\_5\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.57 QUEUE\_5\_C Register (offset = 2058h) [reset = 0h]**

QUEUE\_5\_C is shown in [Figure 16-333](#) and described in [Table 16-347](#).

**Figure 16-333. QUEUE\_5\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

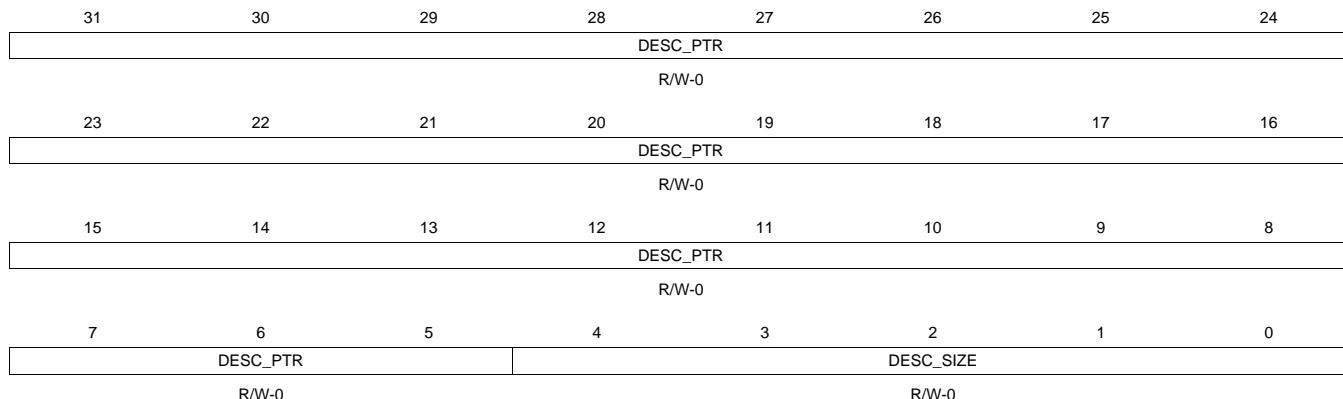
**Table 16-347. QUEUE\_5\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.58 QUEUE\_5\_D Register (offset = 205Ch) [reset = 0h]

QUEUE\_5\_D is shown in [Figure 16-334](#) and described in [Table 16-348](#).

**Figure 16-334. QUEUE\_5\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-348. QUEUE\_5\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.59 QUEUE\_6\_A Register (offset = 2060h) [reset = 0h]**

QUEUE\_6\_A is shown in [Figure 16-335](#) and described in [Table 16-349](#).

**Figure 16-335. QUEUE\_6\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-349. QUEUE\_6\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.60 QUEUE\_6\_B Register (offset = 2064h) [reset = 0h]**

QUEUE\_6\_B is shown in [Figure 16-336](#) and described in [Table 16-350](#).

**Figure 16-336. QUEUE\_6\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

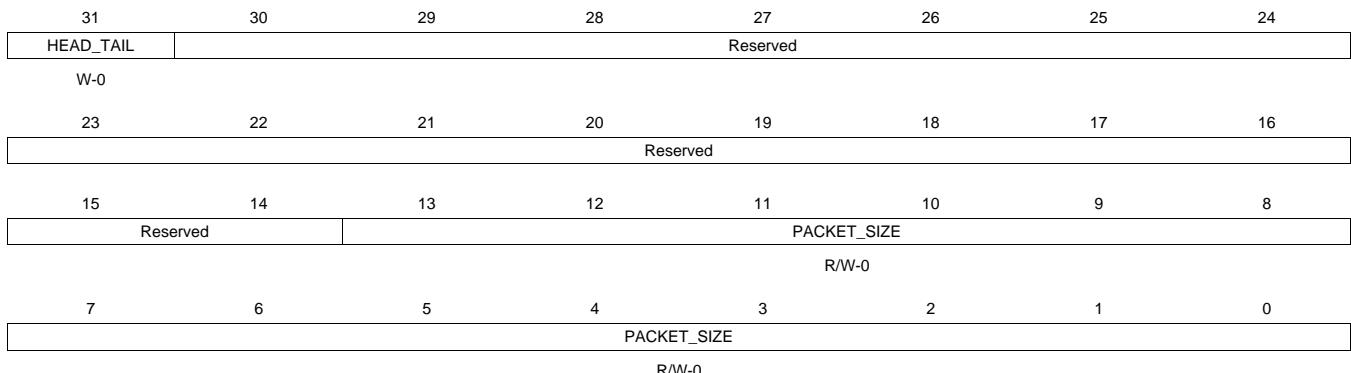
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-350. QUEUE\_6\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.61 QUEUE\_6\_C Register (offset = 2068h) [reset = 0h]**

QUEUE\_6\_C is shown in [Figure 16-337](#) and described in [Table 16-351](#).

**Figure 16-337. QUEUE\_6\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

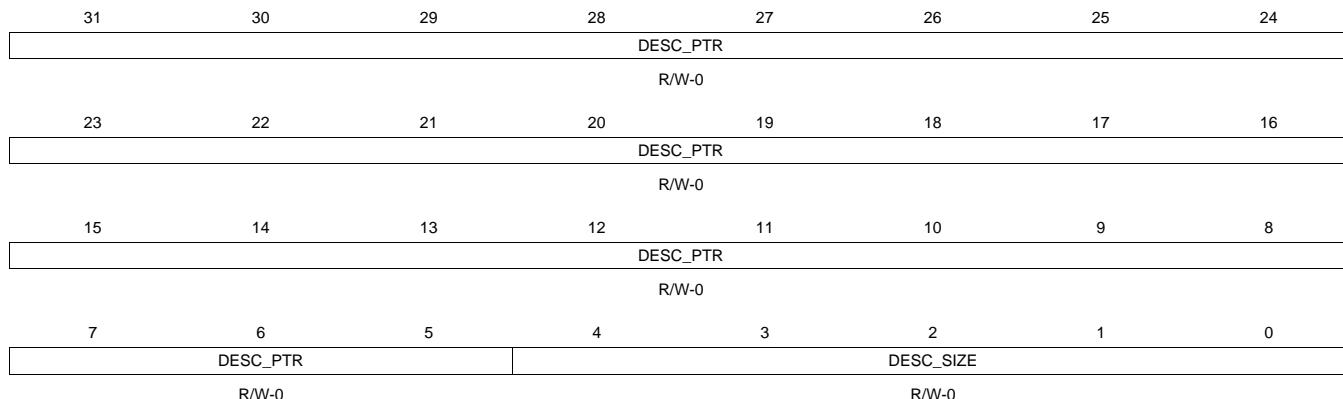
**Table 16-351. QUEUE\_6\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.62 QUEUE\_6\_D Register (offset = 206Ch) [reset = 0h]

QUEUE\_6\_D is shown in [Figure 16-338](#) and described in [Table 16-352](#).

**Figure 16-338. QUEUE\_6\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-352. QUEUE\_6\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.63 QUEUE\_7\_A Register (offset = 2070h) [reset = 0h]**

QUEUE\_7\_A is shown in [Figure 16-339](#) and described in [Table 16-353](#).

**Figure 16-339. QUEUE\_7\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-353. QUEUE\_7\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.64 QUEUE\_7\_B Register (offset = 2074h) [reset = 0h]**

QUEUE\_7\_B is shown in [Figure 16-340](#) and described in [Table 16-354](#).

**Figure 16-340. QUEUE\_7\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

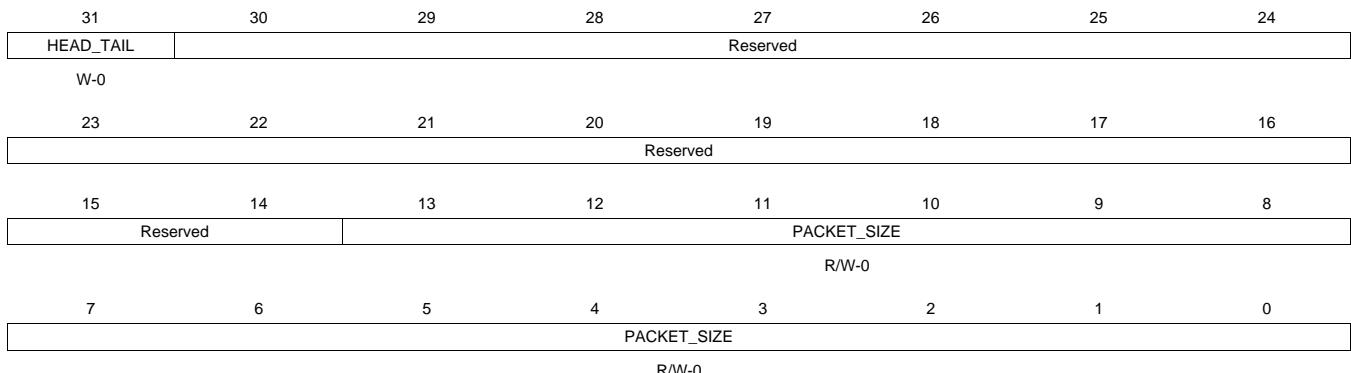
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-354. QUEUE\_7\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.65 QUEUE\_7\_C Register (offset = 2078h) [reset = 0h]**

QUEUE\_7\_C is shown in [Figure 16-341](#) and described in [Table 16-355](#).

**Figure 16-341. QUEUE\_7\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

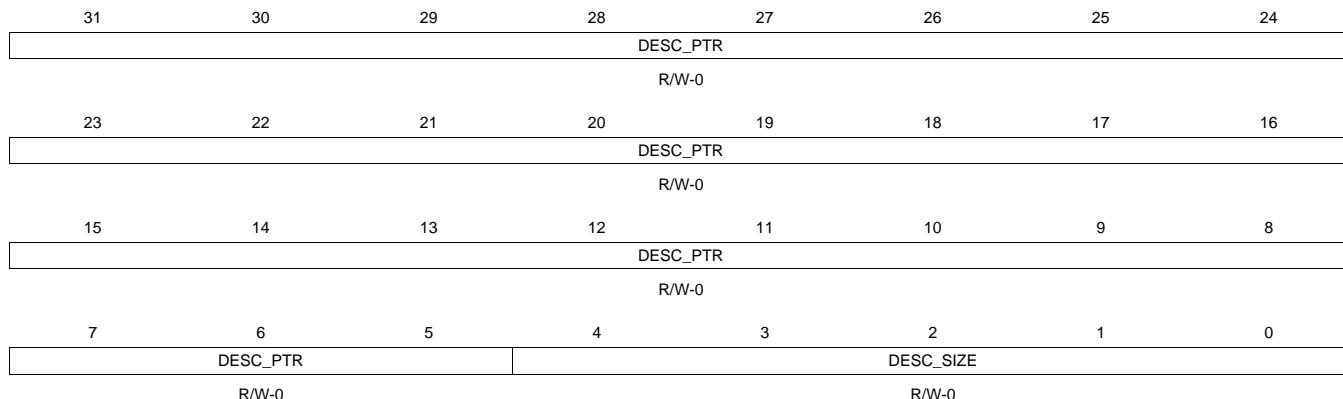
**Table 16-355. QUEUE\_7\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.66 QUEUE\_7\_D Register (offset = 207Ch) [reset = 0h]

QUEUE\_7\_D is shown in [Figure 16-342](#) and described in [Table 16-356](#).

**Figure 16-342. QUEUE\_7\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-356. QUEUE\_7\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.67 QUEUE\_8\_A Register (offset = 2080h) [reset = 0h]

QUEUE\_8\_A is shown in [Figure 16-343](#) and described in [Table 16-357](#).

**Figure 16-343. QUEUE\_8\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-357. QUEUE\_8\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.68 QUEUE\_8\_B Register (offset = 2084h) [reset = 0h]**

 QUEUE\_8\_B is shown in [Figure 16-344](#) and described in [Table 16-358](#).

**Figure 16-344. QUEUE\_8\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

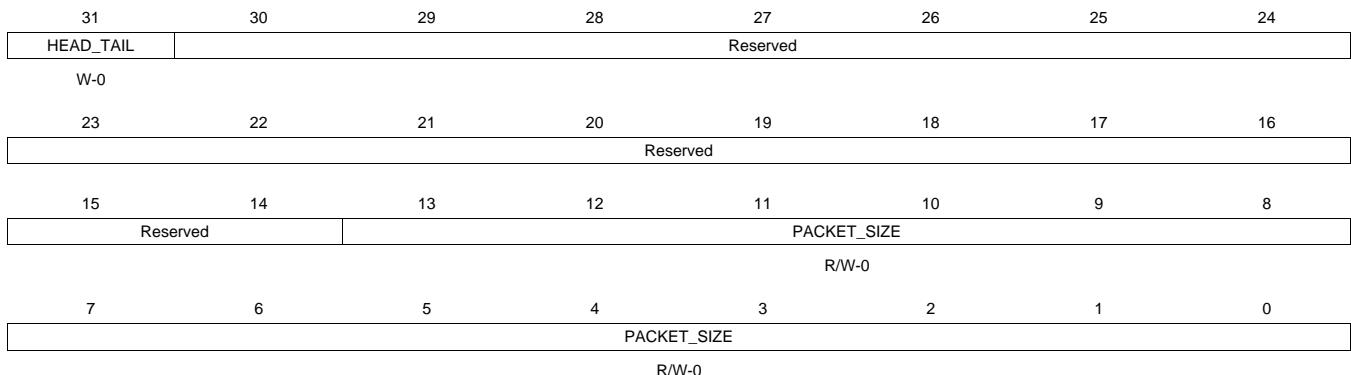
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-358. QUEUE\_8\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.69 QUEUE\_8\_C Register (offset = 2088h) [reset = 0h]**

QUEUE\_8\_C is shown in [Figure 16-345](#) and described in [Table 16-359](#).

**Figure 16-345. QUEUE\_8\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

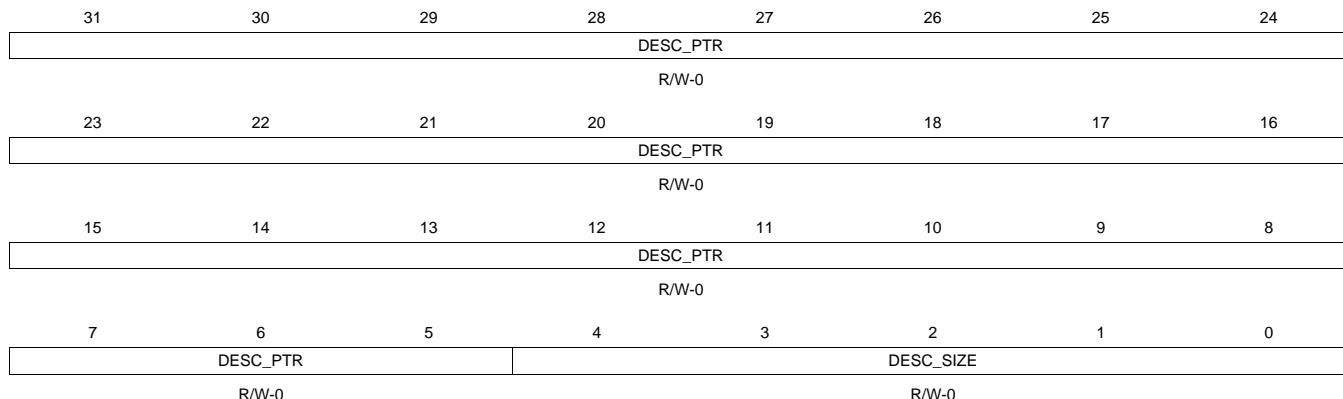
**Table 16-359. QUEUE\_8\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.70 QUEUE\_8\_D Register (offset = 208Ch) [reset = 0h]

QUEUE\_8\_D is shown in [Figure 16-346](#) and described in [Table 16-360](#).

**Figure 16-346. QUEUE\_8\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-360. QUEUE\_8\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.71 QUEUE\_9\_A Register (offset = 2090h) [reset = 0h]

QUEUE\_9\_A is shown in [Figure 16-347](#) and described in [Table 16-361](#).

**Figure 16-347. QUEUE\_9\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-361. QUEUE\_9\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.72 QUEUE\_9\_B Register (offset = 2094h) [reset = 0h]**

QUEUE\_9\_B is shown in [Figure 16-348](#) and described in [Table 16-362](#).

**Figure 16-348. QUEUE\_9\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

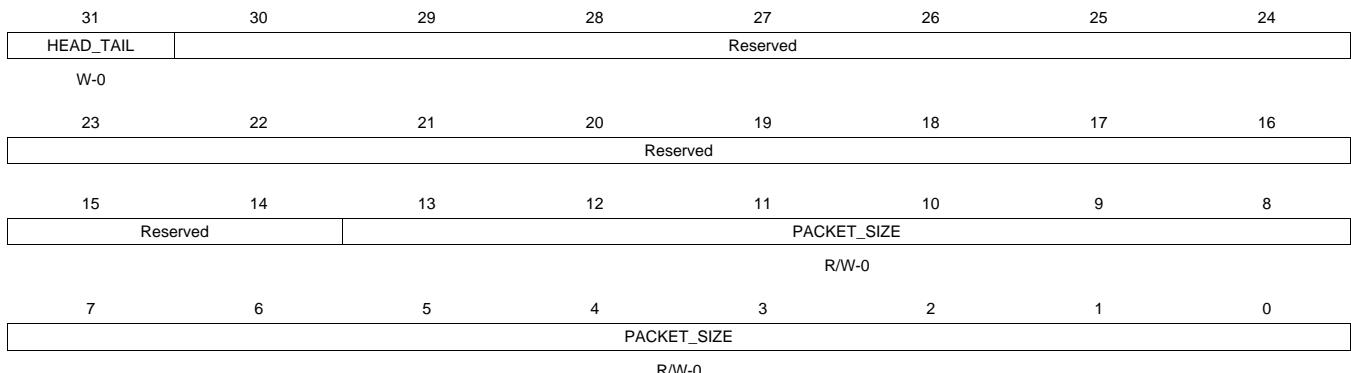
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-362. QUEUE\_9\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.73 QUEUE\_9\_C Register (offset = 2098h) [reset = 0h]**

QUEUE\_9\_C is shown in [Figure 16-349](#) and described in [Table 16-363](#).

**Figure 16-349. QUEUE\_9\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

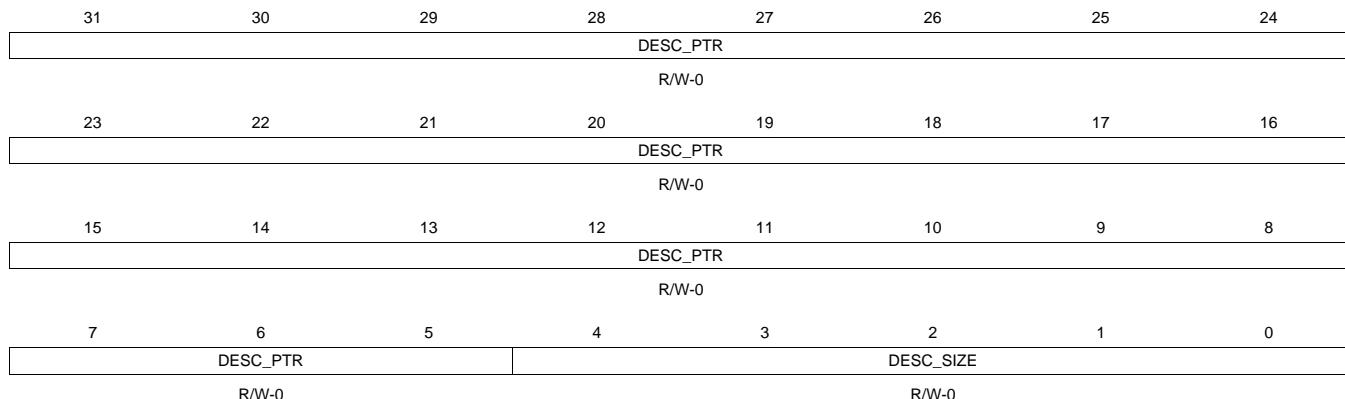
**Table 16-363. QUEUE\_9\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.74 QUEUE\_9\_D Register (offset = 209Ch) [reset = 0h]

QUEUE\_9\_D is shown in [Figure 16-350](#) and described in [Table 16-364](#).

**Figure 16-350. QUEUE\_9\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-364. QUEUE\_9\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.75 QUEUE\_10\_A Register (offset = 20A0h) [reset = 0h]

QUEUE\_10\_A is shown in [Figure 16-351](#) and described in [Table 16-365](#).

**Figure 16-351. QUEUE\_10\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-365. QUEUE\_10\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.76 QUEUE\_10\_B Register (offset = 20A4h) [reset = 0h]**

QUEUE\_10\_B is shown in [Figure 16-352](#) and described in [Table 16-366](#).

**Figure 16-352. QUEUE\_10\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

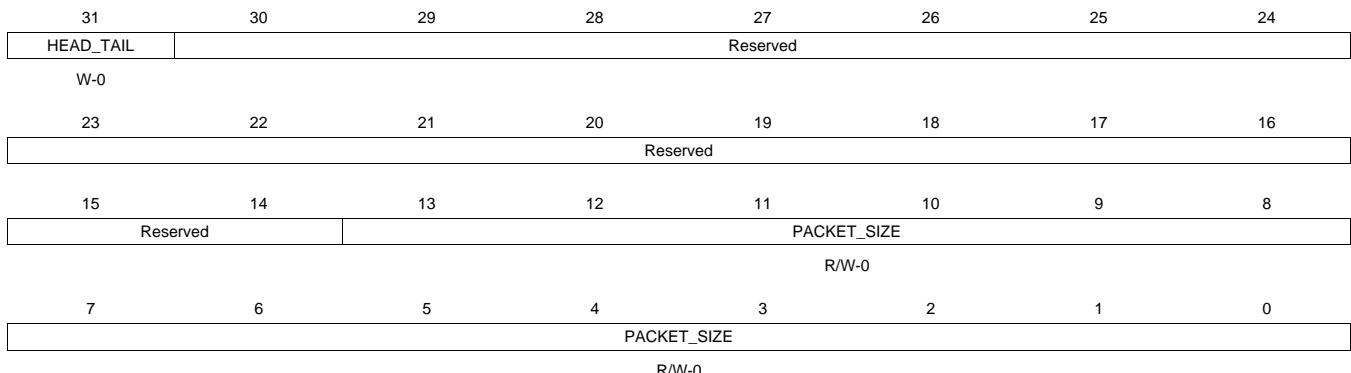
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-366. QUEUE\_10\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.77 QUEUE\_10\_C Register (offset = 20A8h) [reset = 0h]**

QUEUE\_10\_C is shown in [Figure 16-353](#) and described in [Table 16-367](#).

**Figure 16-353. QUEUE\_10\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

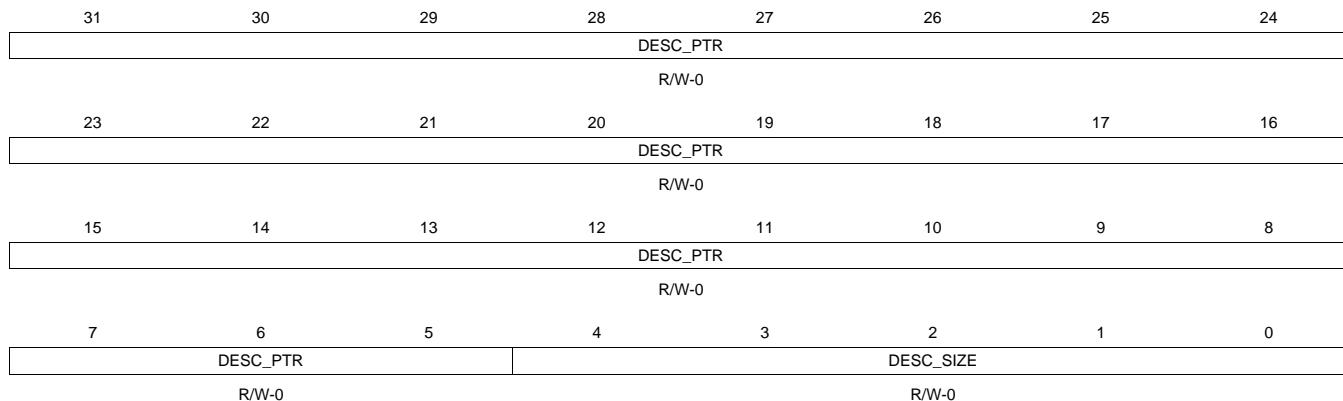
**Table 16-367. QUEUE\_10\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.78 QUEUE\_10\_D Register (offset = 20ACh) [reset = 0h]

QUEUE\_10\_D is shown in [Figure 16-354](#) and described in [Table 16-368](#).

**Figure 16-354. QUEUE\_10\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-368. QUEUE\_10\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.79 QUEUE\_11\_A Register (offset = 20B0h) [reset = 0h]

QUEUE\_11\_A is shown in [Figure 16-355](#) and described in [Table 16-369](#).

**Figure 16-355. QUEUE\_11\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-369. QUEUE\_11\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.80 QUEUE\_11\_B Register (offset = 20B4h) [reset = 0h]**

QUEUE\_11\_B is shown in [Figure 16-356](#) and described in [Table 16-370](#).

**Figure 16-356. QUEUE\_11\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

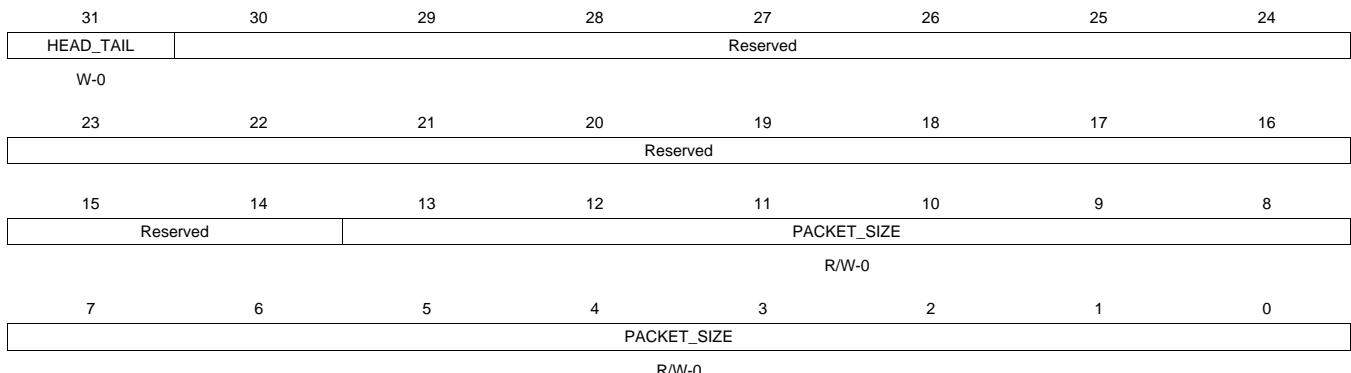
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-370. QUEUE\_11\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.81 QUEUE\_11\_C Register (offset = 20B8h) [reset = 0h]**

QUEUE\_11\_C is shown in [Figure 16-357](#) and described in [Table 16-371](#).

**Figure 16-357. QUEUE\_11\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

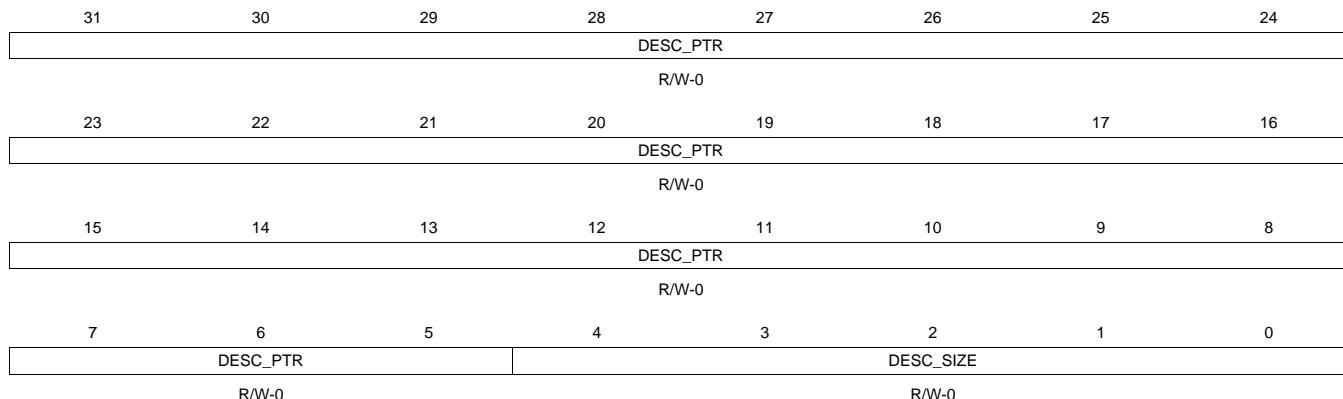
**Table 16-371. QUEUE\_11\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.82 QUEUE\_11\_D Register (offset = 20BCh) [reset = 0h]

QUEUE\_11\_D is shown in [Figure 16-358](#) and described in [Table 16-372](#).

**Figure 16-358. QUEUE\_11\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-372. QUEUE\_11\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.83 QUEUE\_12\_A Register (offset = 20C0h) [reset = 0h]

QUEUE\_12\_A is shown in [Figure 16-359](#) and described in [Table 16-373](#).

**Figure 16-359. QUEUE\_12\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-373. QUEUE\_12\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.84 QUEUE\_12\_B Register (offset = 20C4h) [reset = 0h]**

QUEUE\_12\_B is shown in [Figure 16-360](#) and described in [Table 16-374](#).

**Figure 16-360. QUEUE\_12\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

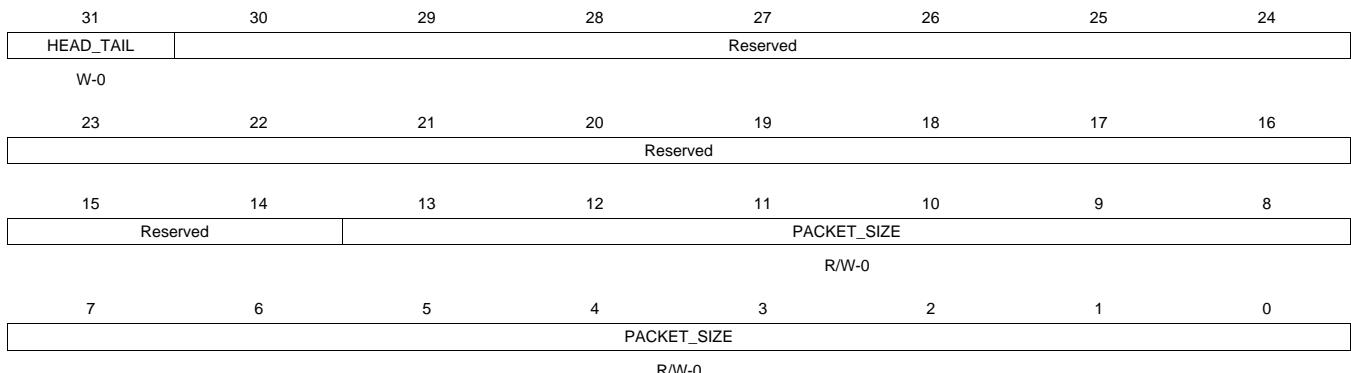
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-374. QUEUE\_12\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.85 QUEUE\_12\_C Register (offset = 20C8h) [reset = 0h]**

QUEUE\_12\_C is shown in [Figure 16-361](#) and described in [Table 16-375](#).

**Figure 16-361. QUEUE\_12\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

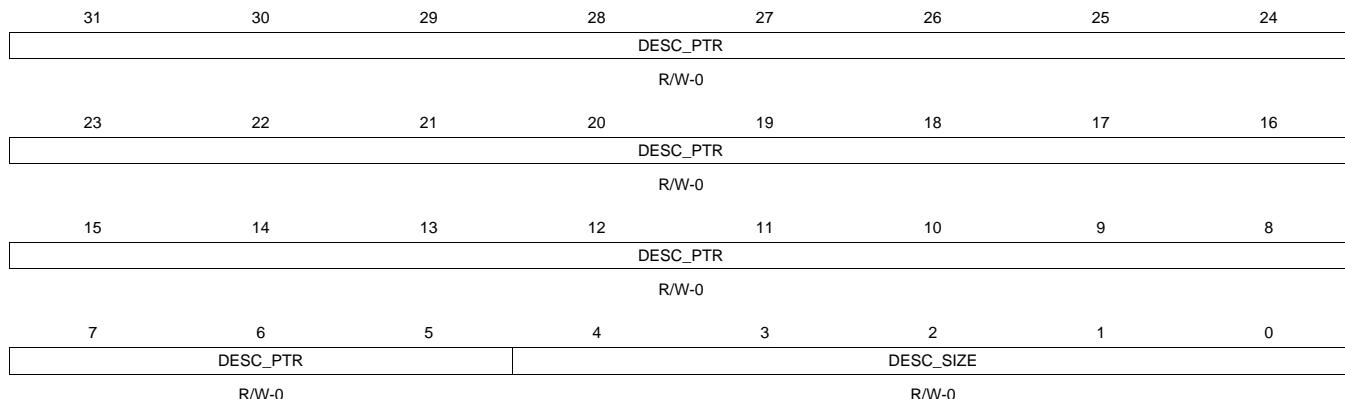
**Table 16-375. QUEUE\_12\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.86 QUEUE\_12\_D Register (offset = 20CCh) [reset = 0h]

QUEUE\_12\_D is shown in [Figure 16-362](#) and described in [Table 16-376](#).

**Figure 16-362. QUEUE\_12\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-376. QUEUE\_12\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.87 QUEUE\_13\_A Register (offset = 20D0h) [reset = 0h]**

QUEUE\_13\_A is shown in [Figure 16-363](#) and described in [Table 16-377](#).

**Figure 16-363. QUEUE\_13\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-377. QUEUE\_13\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.88 QUEUE\_13\_B Register (offset = 20D4h) [reset = 0h]**

QUEUE\_13\_B is shown in [Figure 16-364](#) and described in [Table 16-378](#).

**Figure 16-364. QUEUE\_13\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

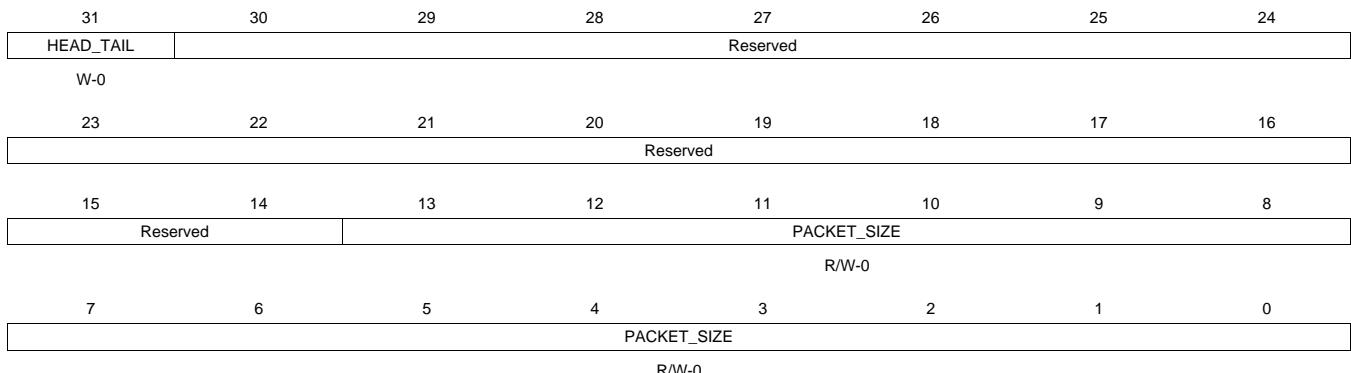
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-378. QUEUE\_13\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.89 QUEUE\_13\_C Register (offset = 20D8h) [reset = 0h]**

QUEUE\_13\_C is shown in [Figure 16-365](#) and described in [Table 16-379](#).

**Figure 16-365. QUEUE\_13\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

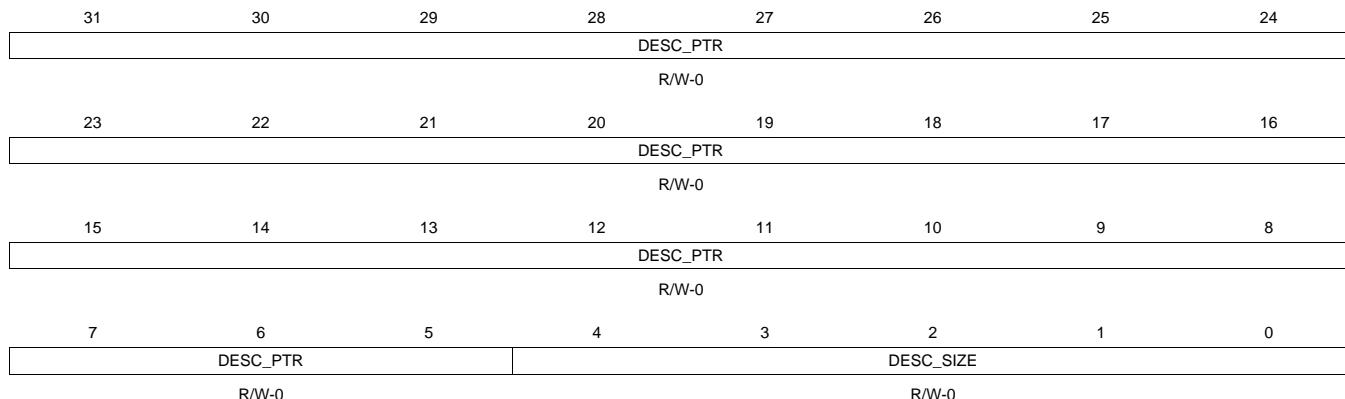
**Table 16-379. QUEUE\_13\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.90 QUEUE\_13\_D Register (offset = 20DCh) [reset = 0h]

QUEUE\_13\_D is shown in [Figure 16-366](#) and described in [Table 16-380](#).

**Figure 16-366. QUEUE\_13\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-380. QUEUE\_13\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.91 QUEUE\_14\_A Register (offset = 20E0h) [reset = 0h]**

QUEUE\_14\_A is shown in [Figure 16-367](#) and described in [Table 16-381](#).

**Figure 16-367. QUEUE\_14\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-381. QUEUE\_14\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.92 QUEUE\_14\_B Register (offset = 20E4h) [reset = 0h]**

QUEUE\_14\_B is shown in [Figure 16-368](#) and described in [Table 16-382](#).

**Figure 16-368. QUEUE\_14\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

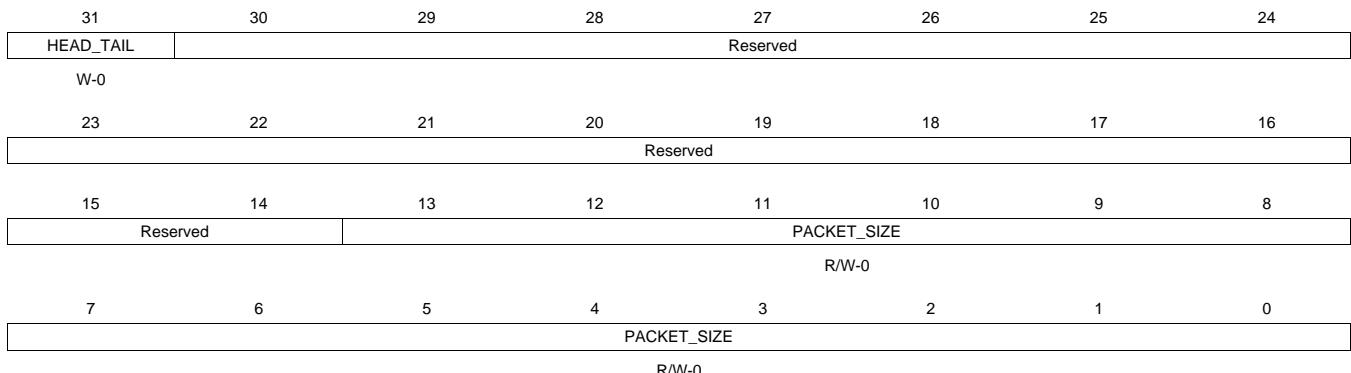
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-382. QUEUE\_14\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.93 QUEUE\_14\_C Register (offset = 20E8h) [reset = 0h]**

QUEUE\_14\_C is shown in [Figure 16-369](#) and described in [Table 16-383](#).

**Figure 16-369. QUEUE\_14\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

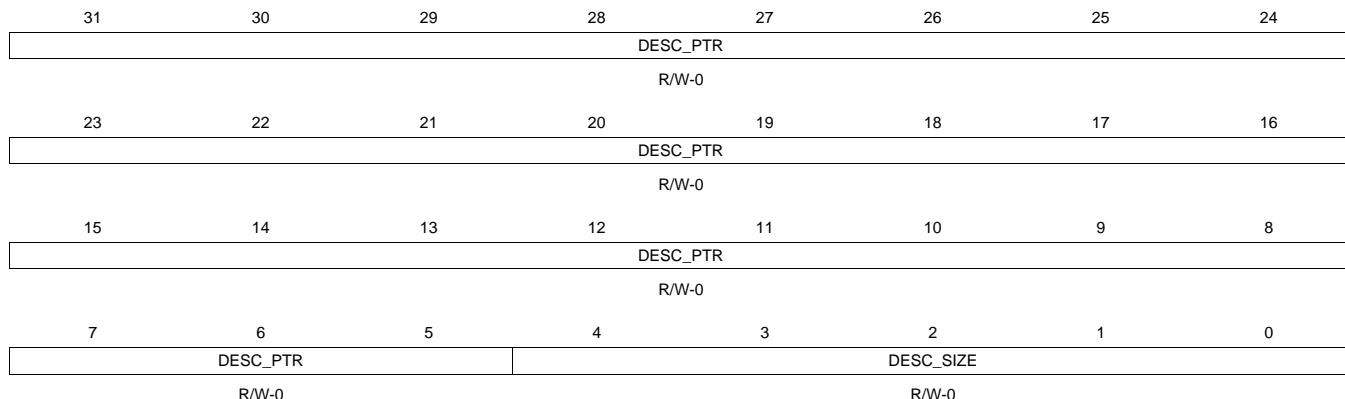
**Table 16-383. QUEUE\_14\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.94 QUEUE\_14\_D Register (offset = 20ECh) [reset = 0h]

QUEUE\_14\_D is shown in [Figure 16-370](#) and described in [Table 16-384](#).

**Figure 16-370. QUEUE\_14\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-384. QUEUE\_14\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.95 QUEUE\_15\_A Register (offset = 20F0h) [reset = 0h]

QUEUE\_15\_A is shown in [Figure 16-371](#) and described in [Table 16-385](#).

**Figure 16-371. QUEUE\_15\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-385. QUEUE\_15\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.96 QUEUE\_15\_B Register (offset = 20F4h) [reset = 0h]**

QUEUE\_15\_B is shown in [Figure 16-372](#) and described in [Table 16-386](#).

**Figure 16-372. QUEUE\_15\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

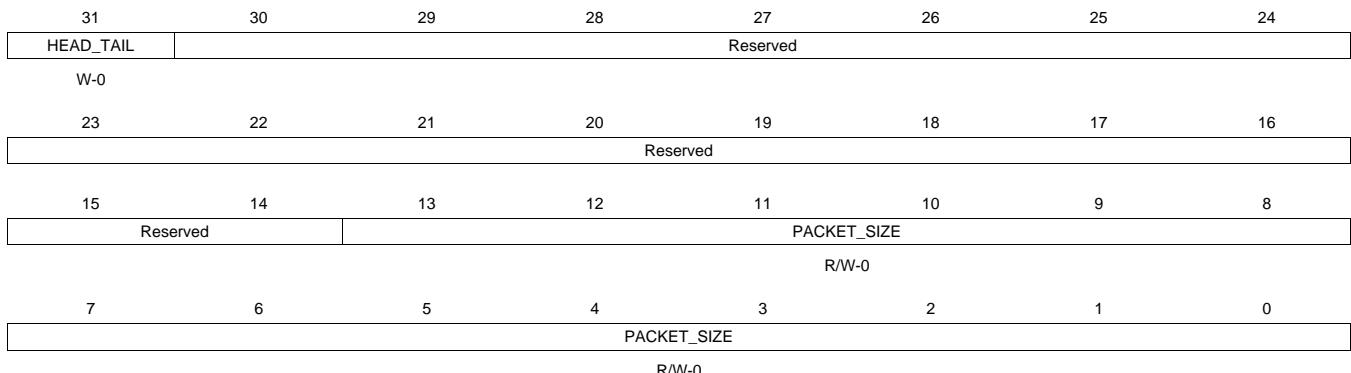
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-386. QUEUE\_15\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.97 QUEUE\_15\_C Register (offset = 20F8h) [reset = 0h]**

QUEUE\_15\_C is shown in [Figure 16-373](#) and described in [Table 16-387](#).

**Figure 16-373. QUEUE\_15\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

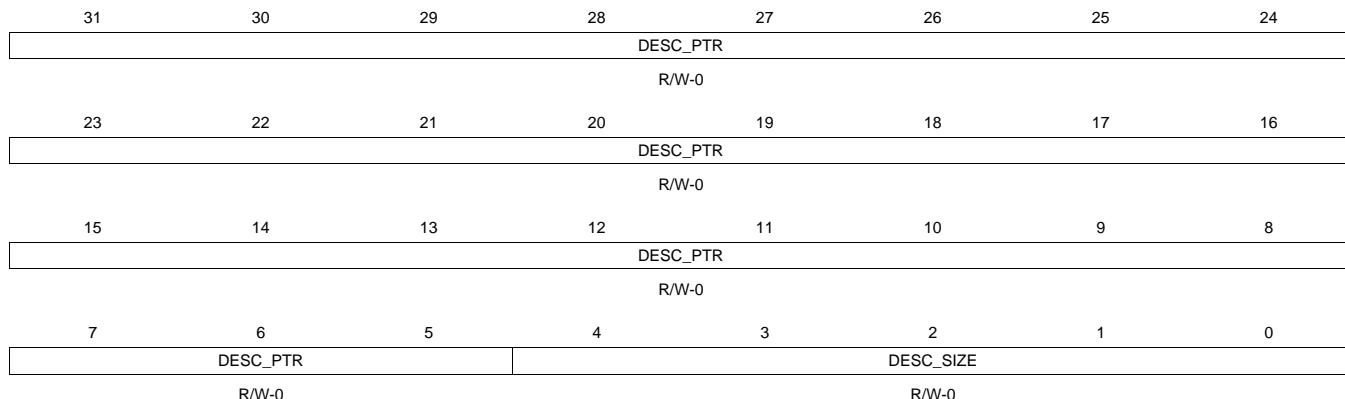
**Table 16-387. QUEUE\_15\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.98 QUEUE\_15\_D Register (offset = 20FCh) [reset = 0h]

QUEUE\_15\_D is shown in [Figure 16-374](#) and described in [Table 16-388](#).

**Figure 16-374. QUEUE\_15\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-388. QUEUE\_15\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.99 QUEUE\_16\_A Register (offset = 2100h) [reset = 0h]**

QUEUE\_16\_A is shown in [Figure 16-375](#) and described in [Table 16-389](#).

**Figure 16-375. QUEUE\_16\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-389. QUEUE\_16\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.100 QUEUE\_16\_B Register (offset = 2104h) [reset = 0h]**

QUEUE\_16\_B is shown in [Figure 16-376](#) and described in [Table 16-390](#).

**Figure 16-376. QUEUE\_16\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

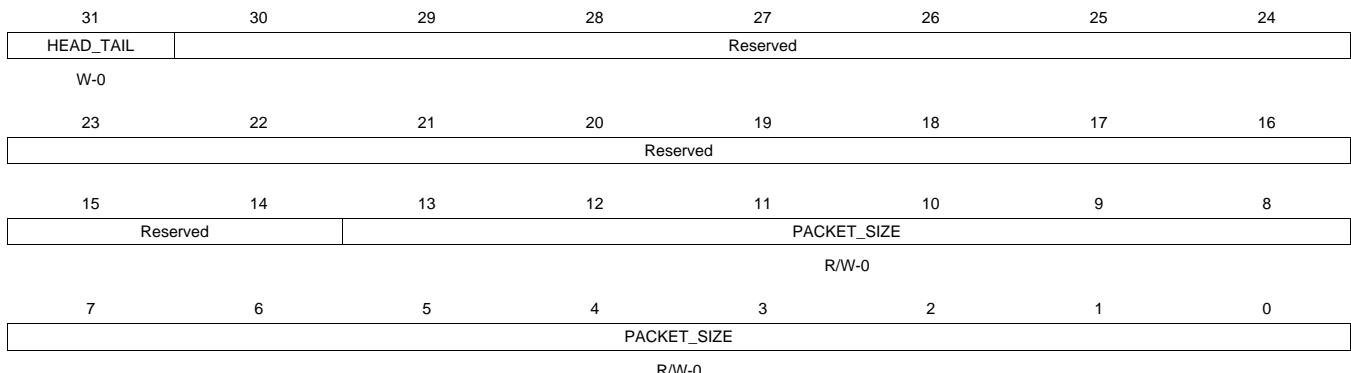
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-390. QUEUE\_16\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.101 QUEUE\_16\_C Register (offset = 2108h) [reset = 0h]**

QUEUE\_16\_C is shown in [Figure 16-377](#) and described in [Table 16-391](#).

**Figure 16-377. QUEUE\_16\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

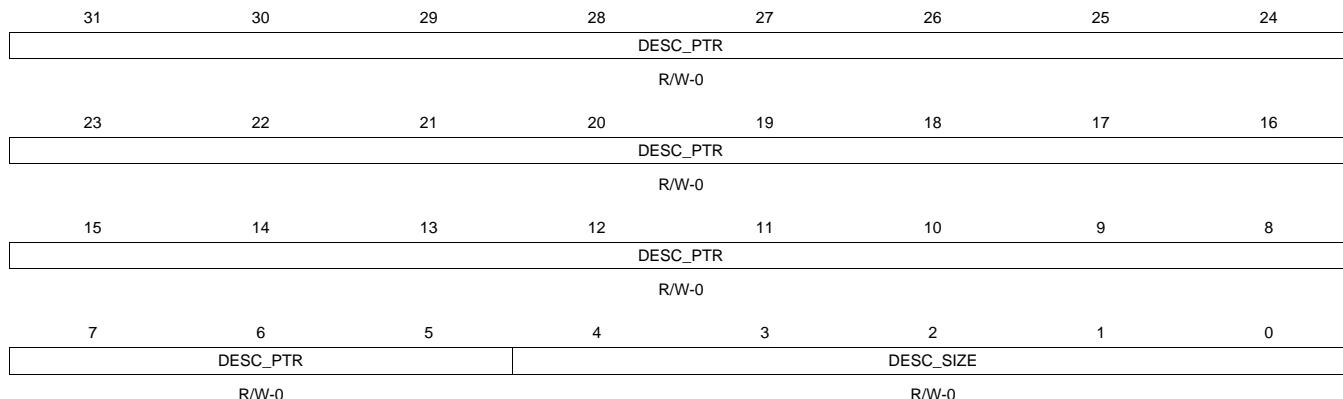
**Table 16-391. QUEUE\_16\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.102 QUEUE\_16\_D Register (offset = 210Ch) [reset = 0h]

QUEUE\_16\_D is shown in [Figure 16-378](#) and described in [Table 16-392](#).

**Figure 16-378. QUEUE\_16\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-392. QUEUE\_16\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.103 QUEUE\_17\_A Register (offset = 2110h) [reset = 0h]**

QUEUE\_17\_A is shown in [Figure 16-379](#) and described in [Table 16-393](#).

**Figure 16-379. QUEUE\_17\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-393. QUEUE\_17\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.104 QUEUE\_17\_B Register (offset = 2114h) [reset = 0h]**

QUEUE\_17\_B is shown in [Figure 16-380](#) and described in [Table 16-394](#).

**Figure 16-380. QUEUE\_17\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

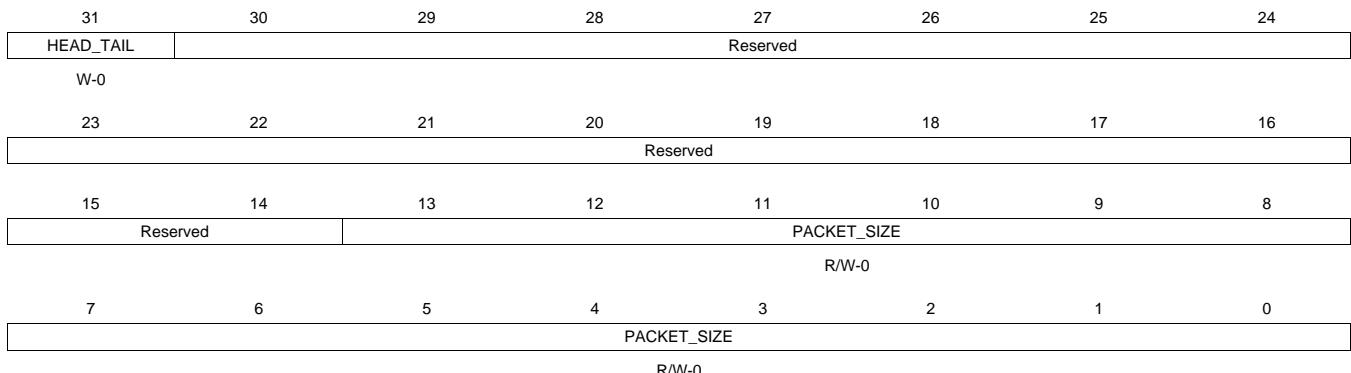
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-394. QUEUE\_17\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.105 QUEUE\_17\_C Register (offset = 2118h) [reset = 0h]**

QUEUE\_17\_C is shown in [Figure 16-381](#) and described in [Table 16-395](#).

**Figure 16-381. QUEUE\_17\_C Register**


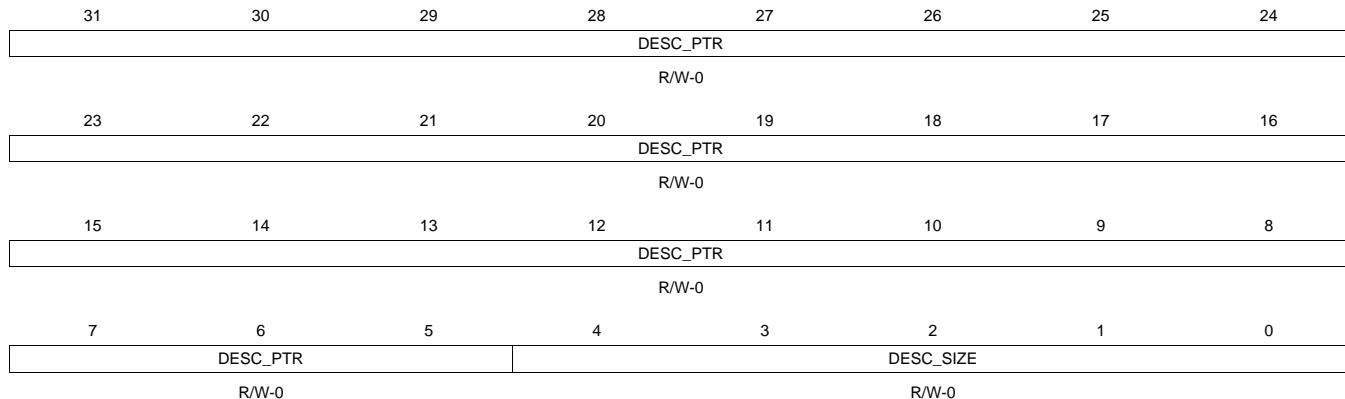
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-395. QUEUE\_17\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.106 QUEUE\_17\_D Register (offset = 211Ch) [reset = 0h]**

 QUEUE\_17\_D is shown in [Figure 16-382](#) and described in [Table 16-396](#).

**Figure 16-382. QUEUE\_17\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-396. QUEUE\_17\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.107 QUEUE\_18\_A Register (offset = 2120h) [reset = 0h]

QUEUE\_18\_A is shown in [Figure 16-383](#) and described in [Table 16-397](#).

**Figure 16-383. QUEUE\_18\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-397. QUEUE\_18\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.108 QUEUE\_18\_B Register (offset = 2124h) [reset = 0h]**

QUEUE\_18\_B is shown in [Figure 16-384](#) and described in [Table 16-398](#).

**Figure 16-384. QUEUE\_18\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

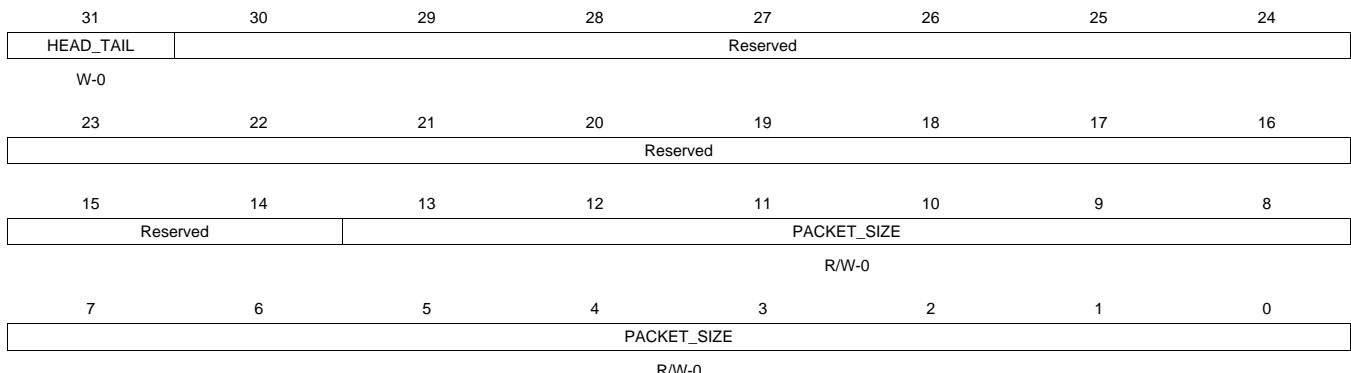
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-398. QUEUE\_18\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.109 QUEUE\_18\_C Register (offset = 2128h) [reset = 0h]**

QUEUE\_18\_C is shown in [Figure 16-385](#) and described in [Table 16-399](#).

**Figure 16-385. QUEUE\_18\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

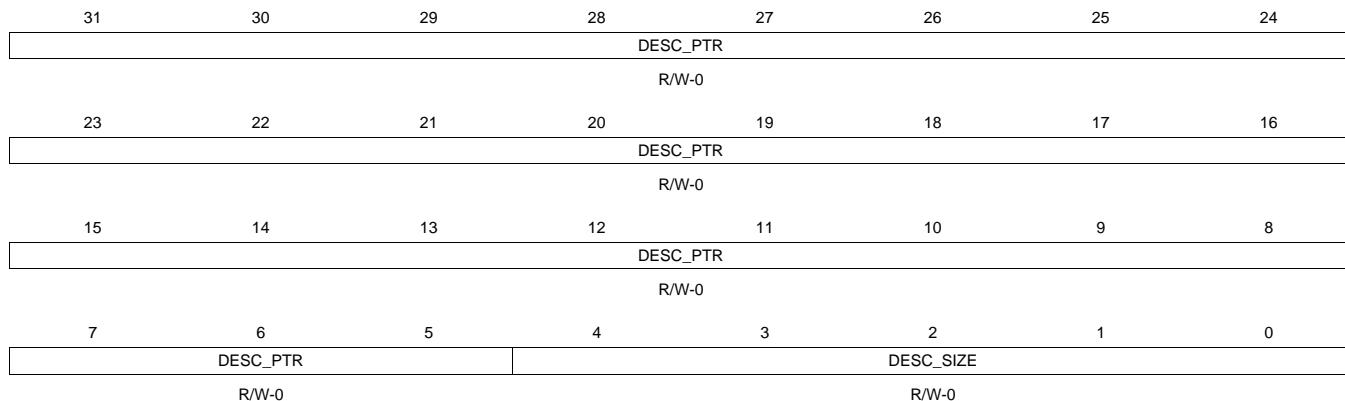
**Table 16-399. QUEUE\_18\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.110 QUEUE\_18\_D Register (offset = 212Ch) [reset = 0h]

QUEUE\_18\_D is shown in [Figure 16-386](#) and described in [Table 16-400](#).

**Figure 16-386. QUEUE\_18\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-400. QUEUE\_18\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.111 QUEUE\_19\_A Register (offset = 2130h) [reset = 0h]**

QUEUE\_19\_A is shown in [Figure 16-387](#) and described in [Table 16-401](#).

**Figure 16-387. QUEUE\_19\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-401. QUEUE\_19\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.112 QUEUE\_19\_B Register (offset = 2134h) [reset = 0h]**

QUEUE\_19\_B is shown in [Figure 16-388](#) and described in [Table 16-402](#).

**Figure 16-388. QUEUE\_19\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

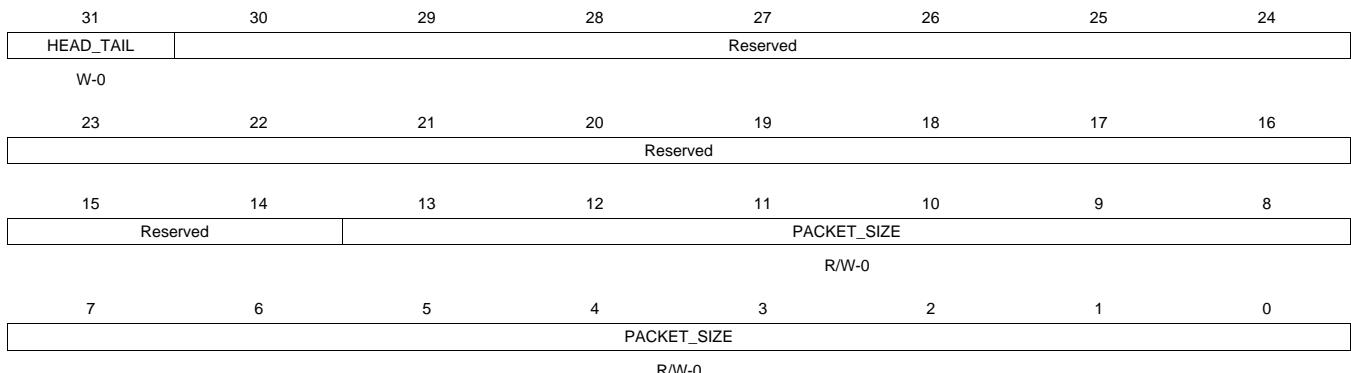
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-402. QUEUE\_19\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.113 QUEUE\_19\_C Register (offset = 2138h) [reset = 0h]**

QUEUE\_19\_C is shown in [Figure 16-389](#) and described in [Table 16-403](#).

**Figure 16-389. QUEUE\_19\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

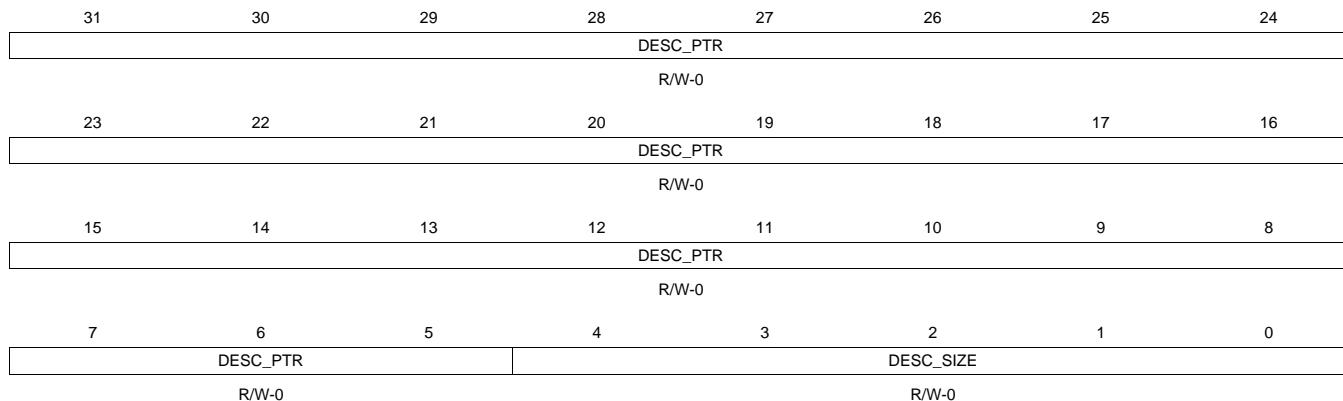
**Table 16-403. QUEUE\_19\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.114 QUEUE\_19\_D Register (offset = 213Ch) [reset = 0h]

QUEUE\_19\_D is shown in [Figure 16-390](#) and described in [Table 16-404](#).

**Figure 16-390. QUEUE\_19\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-404. QUEUE\_19\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.115 QUEUE\_20\_A Register (offset = 2140h) [reset = 0h]

QUEUE\_20\_A is shown in [Figure 16-391](#) and described in [Table 16-405](#).

**Figure 16-391. QUEUE\_20\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-405. QUEUE\_20\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.116 QUEUE\_20\_B Register (offset = 2144h) [reset = 0h]**

QUEUE\_20\_B is shown in [Figure 16-392](#) and described in [Table 16-406](#).

**Figure 16-392. QUEUE\_20\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

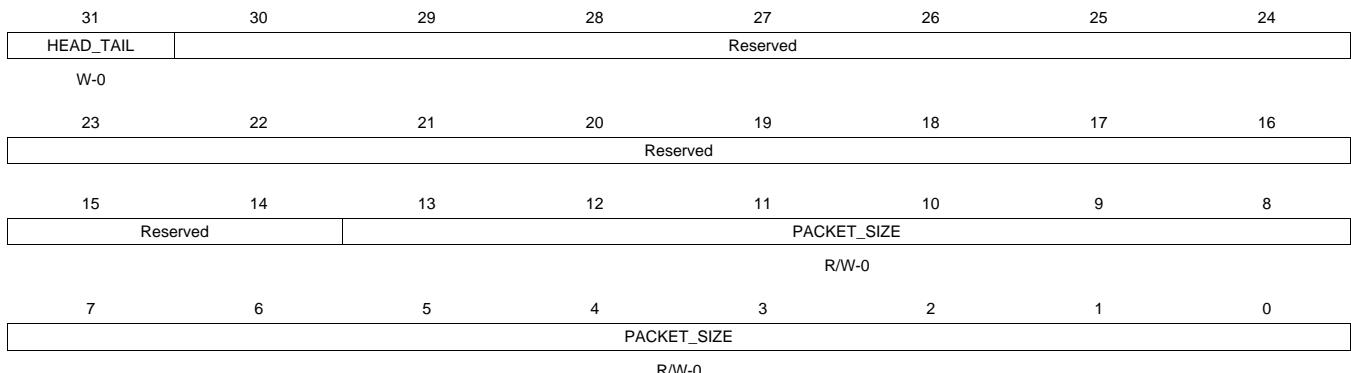
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-406. QUEUE\_20\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.117 QUEUE\_20\_C Register (offset = 2148h) [reset = 0h]**

QUEUE\_20\_C is shown in [Figure 16-393](#) and described in [Table 16-407](#).

**Figure 16-393. QUEUE\_20\_C Register**


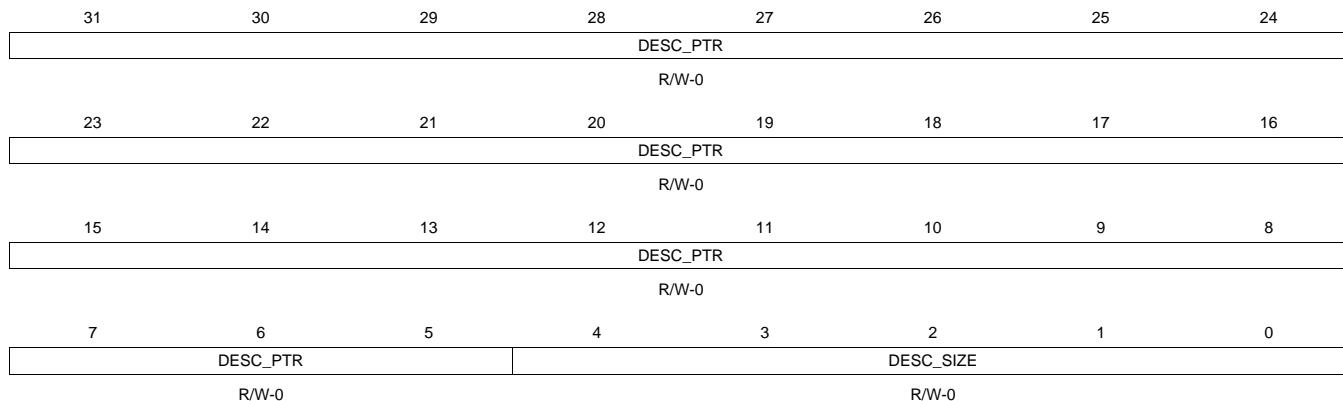
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-407. QUEUE\_20\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.118 QUEUE\_20\_D Register (offset = 214Ch) [reset = 0h]**

 QUEUE\_20\_D is shown in [Figure 16-394](#) and described in [Table 16-408](#).

**Figure 16-394. QUEUE\_20\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-408. QUEUE\_20\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.119 QUEUE\_21\_A Register (offset = 2150h) [reset = 0h]**

QUEUE\_21\_A is shown in [Figure 16-395](#) and described in [Table 16-409](#).

**Figure 16-395. QUEUE\_21\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-409. QUEUE\_21\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.120 QUEUE\_21\_B Register (offset = 2154h) [reset = 0h]**

QUEUE\_21\_B is shown in [Figure 16-396](#) and described in [Table 16-410](#).

**Figure 16-396. QUEUE\_21\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

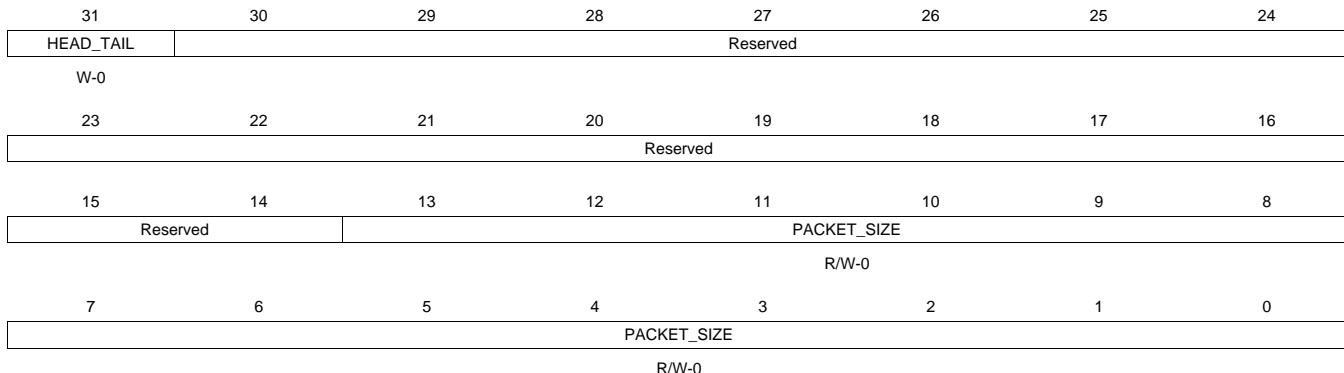
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-410. QUEUE\_21\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.121 QUEUE\_21\_C Register (offset = 2158h) [reset = 0h]**

QUEUE\_21\_C is shown in [Figure 16-397](#) and described in [Table 16-411](#).

**Figure 16-397. QUEUE\_21\_C Register**


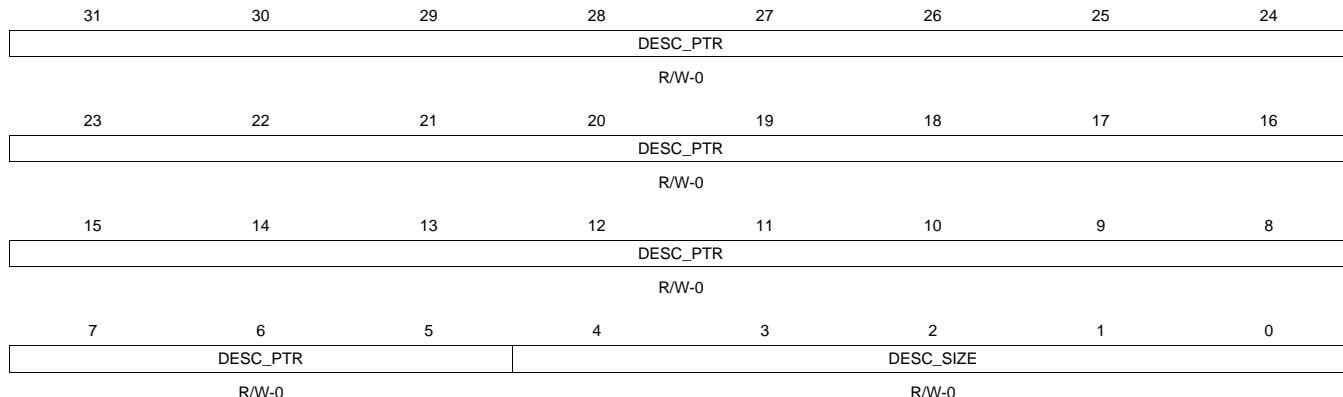
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-411. QUEUE\_21\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.122 QUEUE\_21\_D Register (offset = 215Ch) [reset = 0h]**

 QUEUE\_21\_D is shown in [Figure 16-398](#) and described in [Table 16-412](#).

**Figure 16-398. QUEUE\_21\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-412. QUEUE\_21\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.123 QUEUE\_22\_A Register (offset = 2160h) [reset = 0h]**

QUEUE\_22\_A is shown in [Figure 16-399](#) and described in [Table 16-413](#).

**Figure 16-399. QUEUE\_22\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-413. QUEUE\_22\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.124 QUEUE\_22\_B Register (offset = 2164h) [reset = 0h]**

QUEUE\_22\_B is shown in [Figure 16-400](#) and described in [Table 16-414](#).

**Figure 16-400. QUEUE\_22\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

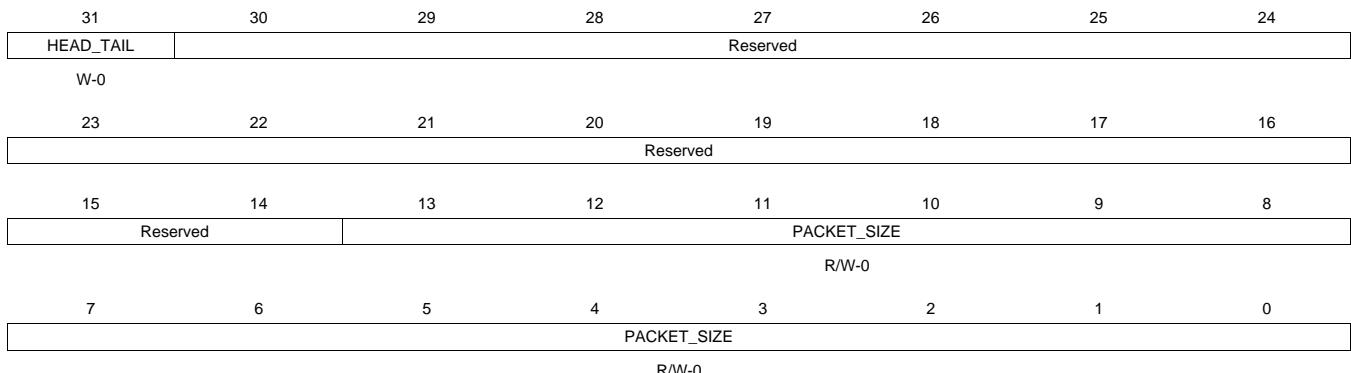
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-414. QUEUE\_22\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.125 QUEUE\_22\_C Register (offset = 2168h) [reset = 0h]**

QUEUE\_22\_C is shown in [Figure 16-401](#) and described in [Table 16-415](#).

**Figure 16-401. QUEUE\_22\_C Register**


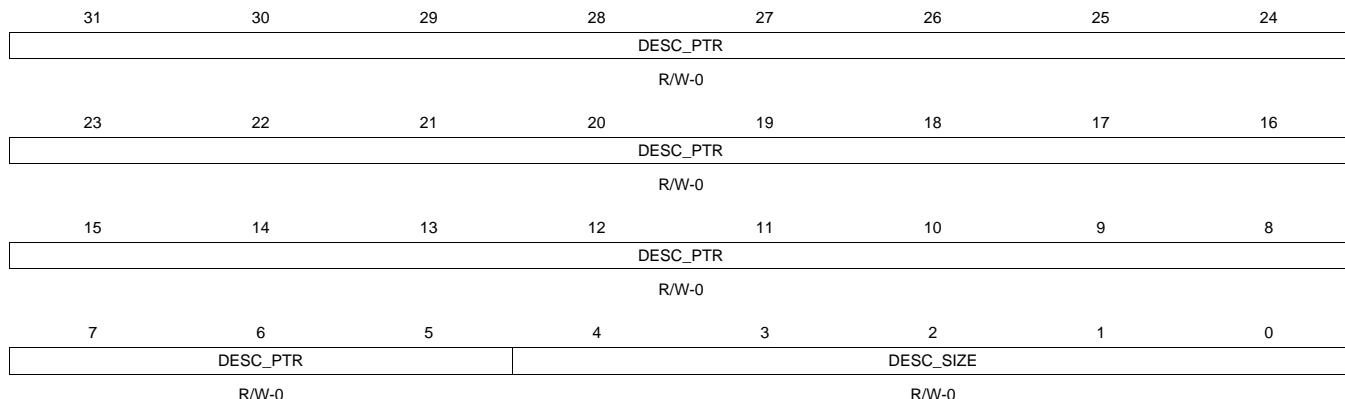
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-415. QUEUE\_22\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.126 QUEUE\_22\_D Register (offset = 216Ch) [reset = 0h]**

 QUEUE\_22\_D is shown in [Figure 16-402](#) and described in [Table 16-416](#).

**Figure 16-402. QUEUE\_22\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-416. QUEUE\_22\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.127 QUEUE\_23\_A Register (offset = 2170h) [reset = 0h]**

QUEUE\_23\_A is shown in [Figure 16-403](#) and described in [Table 16-417](#).

**Figure 16-403. QUEUE\_23\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-417. QUEUE\_23\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.128 QUEUE\_23\_B Register (offset = 2174h) [reset = 0h]**

QUEUE\_23\_B is shown in [Figure 16-404](#) and described in [Table 16-418](#).

**Figure 16-404. QUEUE\_23\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

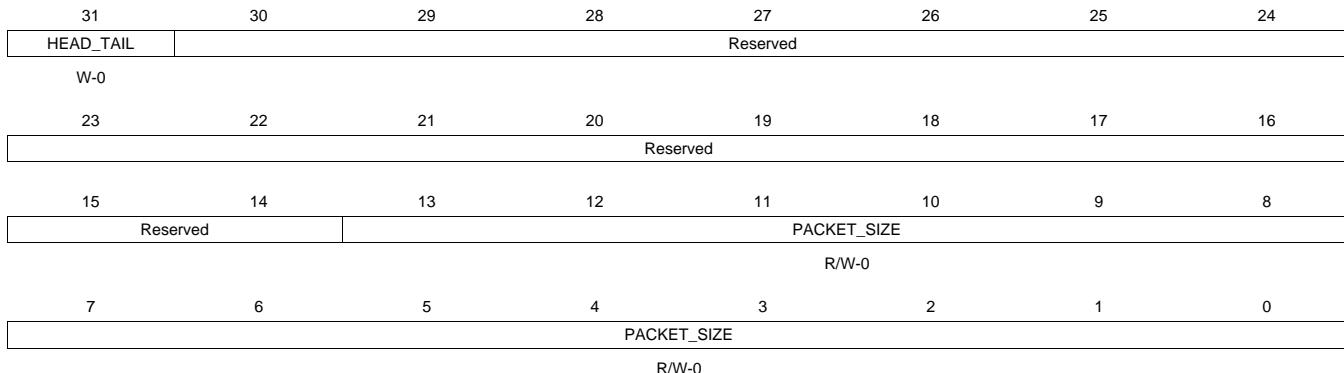
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-418. QUEUE\_23\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.129 QUEUE\_23\_C Register (offset = 2178h) [reset = 0h]**

QUEUE\_23\_C is shown in [Figure 16-405](#) and described in [Table 16-419](#).

**Figure 16-405. QUEUE\_23\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

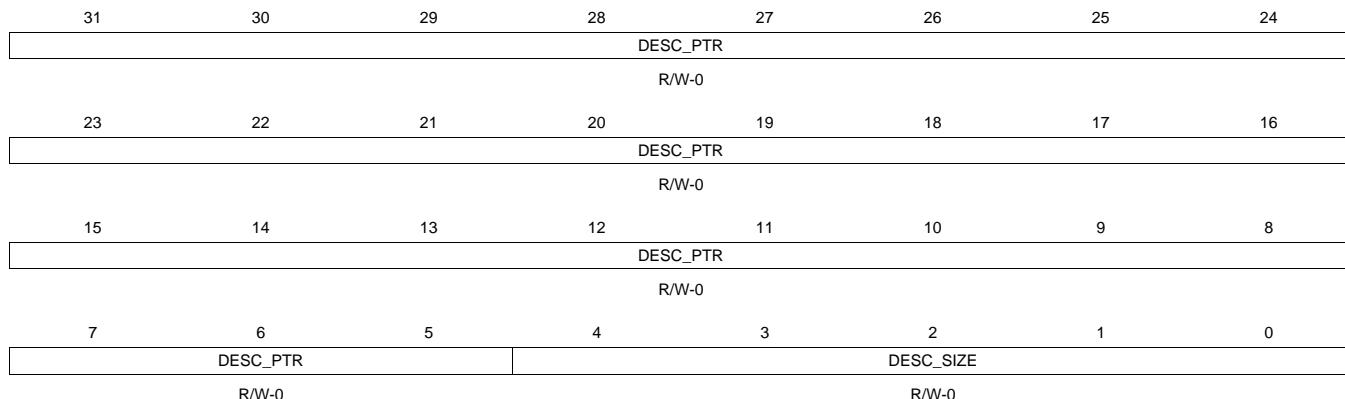
**Table 16-419. QUEUE\_23\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.130 QUEUE\_23\_D Register (offset = 217Ch) [reset = 0h]

QUEUE\_23\_D is shown in [Figure 16-406](#) and described in [Table 16-420](#).

**Figure 16-406. QUEUE\_23\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-420. QUEUE\_23\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.131 QUEUE\_24\_A Register (offset = 2180h) [reset = 0h]

QUEUE\_24\_A is shown in [Figure 16-407](#) and described in [Table 16-421](#).

**Figure 16-407. QUEUE\_24\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-421. QUEUE\_24\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.132 QUEUE\_24\_B Register (offset = 2184h) [reset = 0h]**

QUEUE\_24\_B is shown in [Figure 16-408](#) and described in [Table 16-422](#).

**Figure 16-408. QUEUE\_24\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

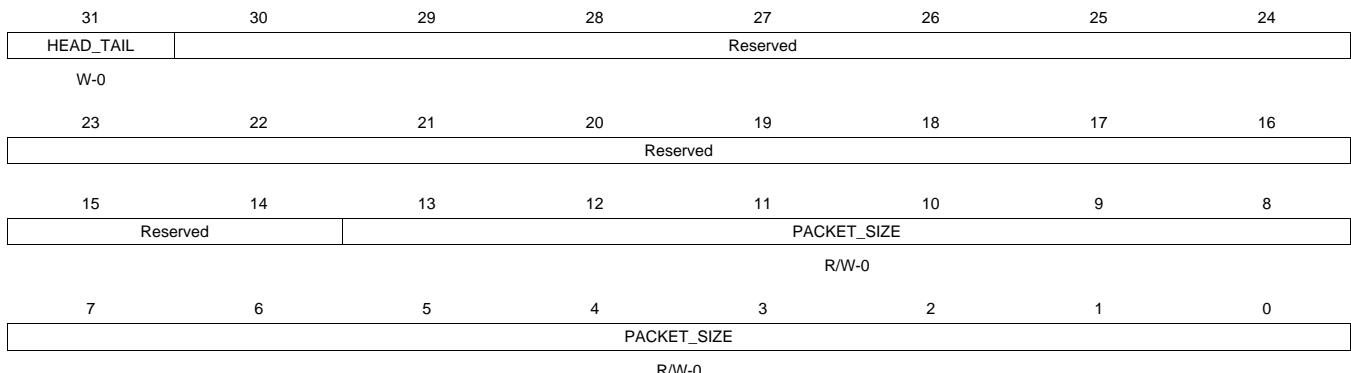
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-422. QUEUE\_24\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.133 QUEUE\_24\_C Register (offset = 2188h) [reset = 0h]**

QUEUE\_24\_C is shown in [Figure 16-409](#) and described in [Table 16-423](#).

**Figure 16-409. QUEUE\_24\_C Register**


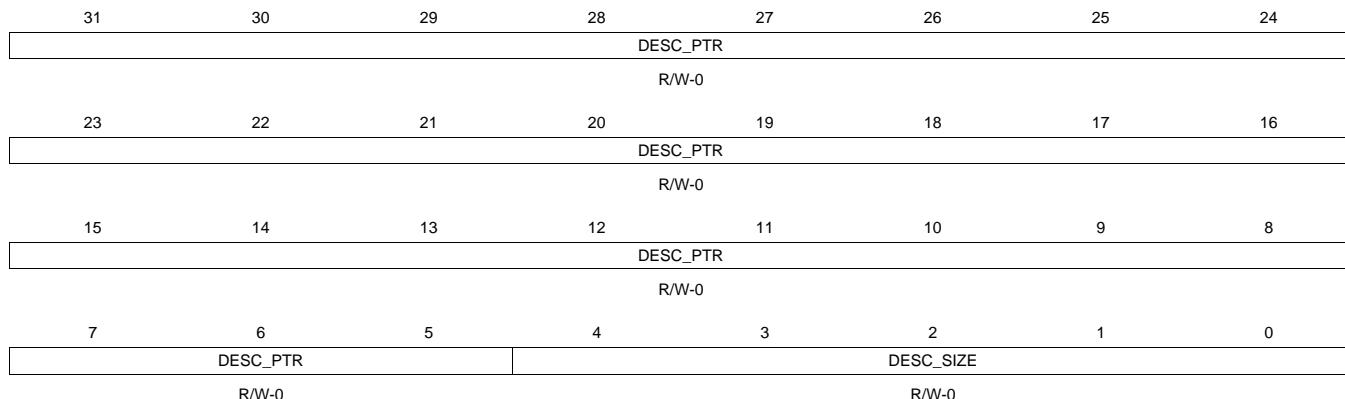
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-423. QUEUE\_24\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.134 QUEUE\_24\_D Register (offset = 218Ch) [reset = 0h]**

 QUEUE\_24\_D is shown in [Figure 16-410](#) and described in [Table 16-424](#).

**Figure 16-410. QUEUE\_24\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-424. QUEUE\_24\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.135 QUEUE\_25\_A Register (offset = 2190h) [reset = 0h]**

QUEUE\_25\_A is shown in [Figure 16-411](#) and described in [Table 16-425](#).

**Figure 16-411. QUEUE\_25\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-425. QUEUE\_25\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.136 QUEUE\_25\_B Register (offset = 2194h) [reset = 0h]**

QUEUE\_25\_B is shown in [Figure 16-412](#) and described in [Table 16-426](#).

**Figure 16-412. QUEUE\_25\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

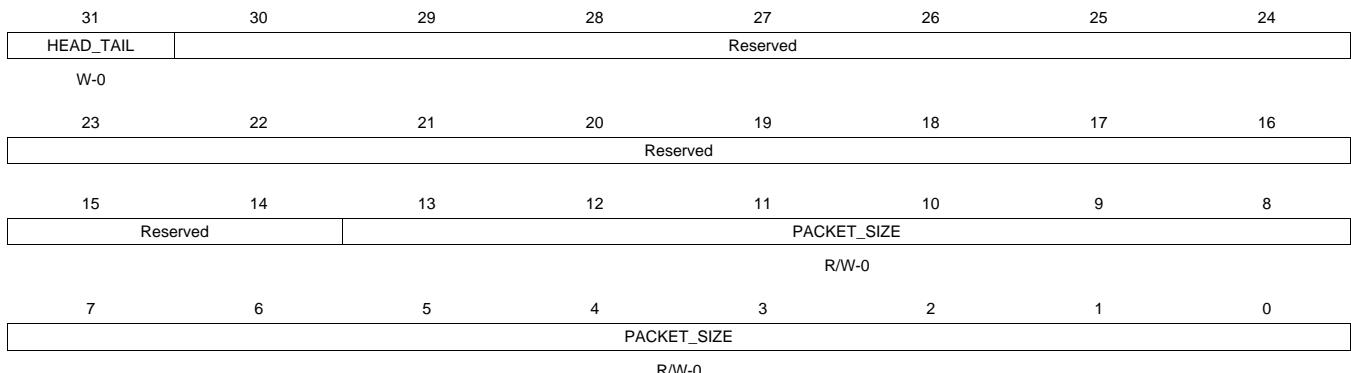
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-426. QUEUE\_25\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.137 QUEUE\_25\_C Register (offset = 2198h) [reset = 0h]**

QUEUE\_25\_C is shown in [Figure 16-413](#) and described in [Table 16-427](#).

**Figure 16-413. QUEUE\_25\_C Register**


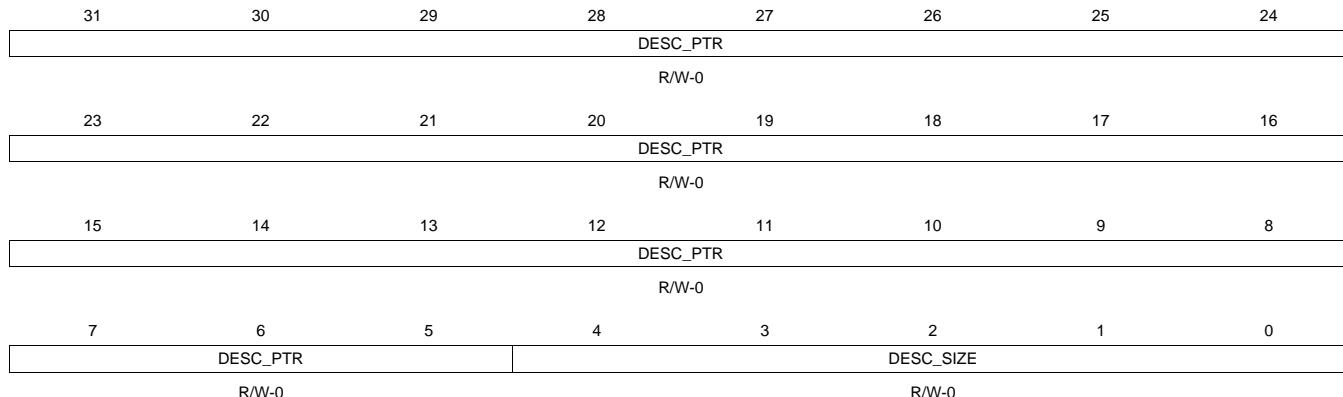
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-427. QUEUE\_25\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.138 QUEUE\_25\_D Register (offset = 219Ch) [reset = 0h]**

 QUEUE\_25\_D is shown in [Figure 16-414](#) and described in [Table 16-428](#).

**Figure 16-414. QUEUE\_25\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-428. QUEUE\_25\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.139 QUEUE\_26\_A Register (offset = 21A0h) [reset = 0h]**

QUEUE\_26\_A is shown in [Figure 16-415](#) and described in [Table 16-429](#).

**Figure 16-415. QUEUE\_26\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-429. QUEUE\_26\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.140 QUEUE\_26\_B Register (offset = 21A4h) [reset = 0h]**

QUEUE\_26\_B is shown in [Figure 16-416](#) and described in [Table 16-430](#).

**Figure 16-416. QUEUE\_26\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

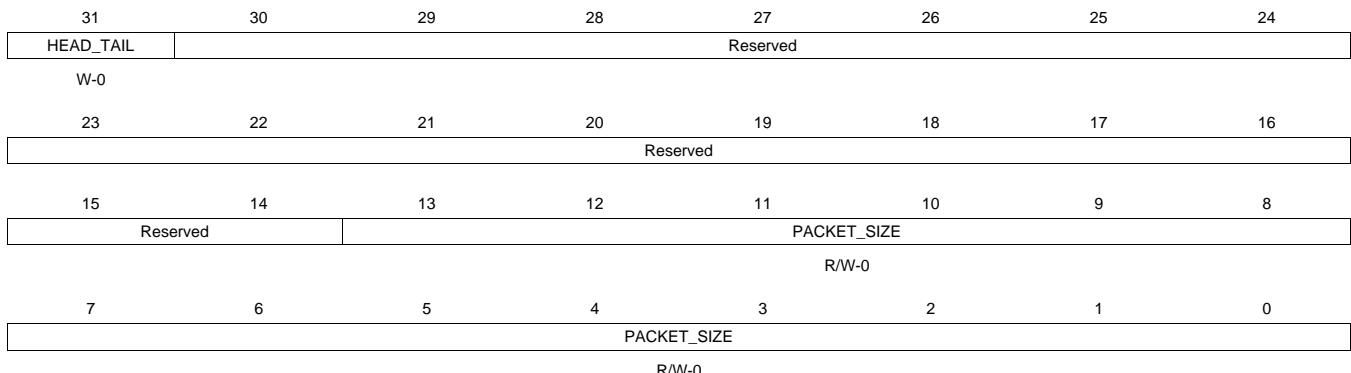
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-430. QUEUE\_26\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.141 QUEUE\_26\_C Register (offset = 21A8h) [reset = 0h]**

QUEUE\_26\_C is shown in [Figure 16-417](#) and described in [Table 16-431](#).

**Figure 16-417. QUEUE\_26\_C Register**


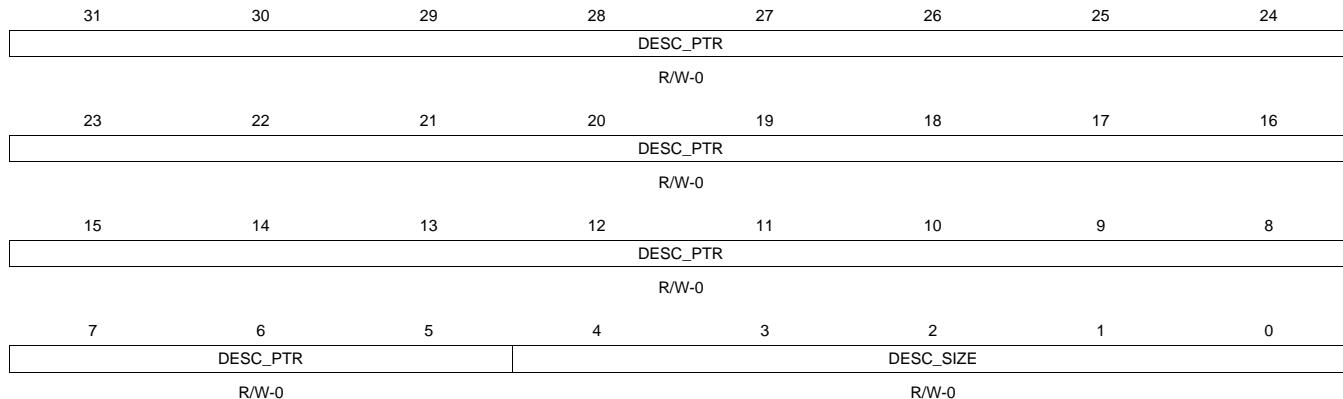
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-431. QUEUE\_26\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.142 QUEUE\_26\_D Register (offset = 21ACh) [reset = 0h]**

 QUEUE\_26\_D is shown in [Figure 16-418](#) and described in [Table 16-432](#).

**Figure 16-418. QUEUE\_26\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-432. QUEUE\_26\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.143 QUEUE\_27\_A Register (offset = 21B0h) [reset = 0h]**

QUEUE\_27\_A is shown in [Figure 16-419](#) and described in [Table 16-433](#).

**Figure 16-419. QUEUE\_27\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-433. QUEUE\_27\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.144 QUEUE\_27\_B Register (offset = 21B4h) [reset = 0h]**

QUEUE\_27\_B is shown in [Figure 16-420](#) and described in [Table 16-434](#).

**Figure 16-420. QUEUE\_27\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

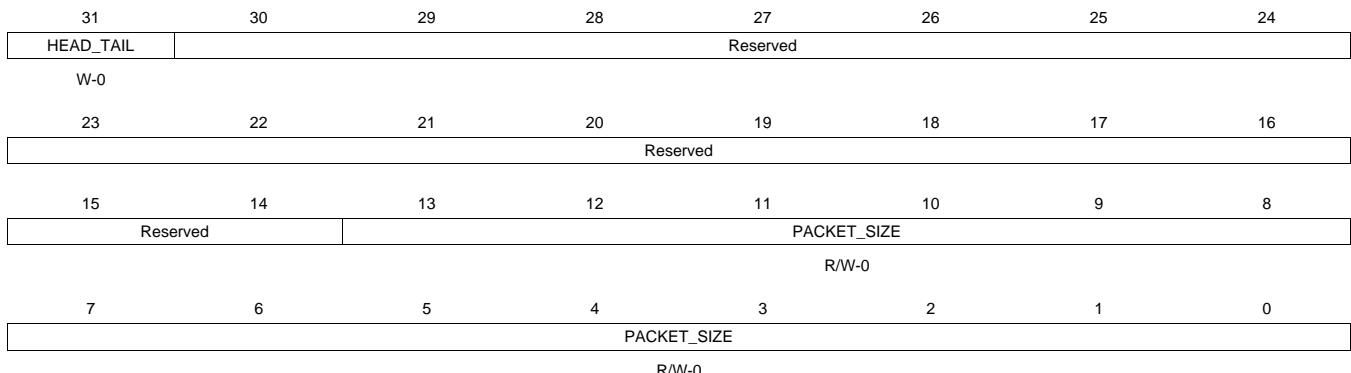
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-434. QUEUE\_27\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.145 QUEUE\_27\_C Register (offset = 21B8h) [reset = 0h]**

QUEUE\_27\_C is shown in [Figure 16-421](#) and described in [Table 16-435](#).

**Figure 16-421. QUEUE\_27\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

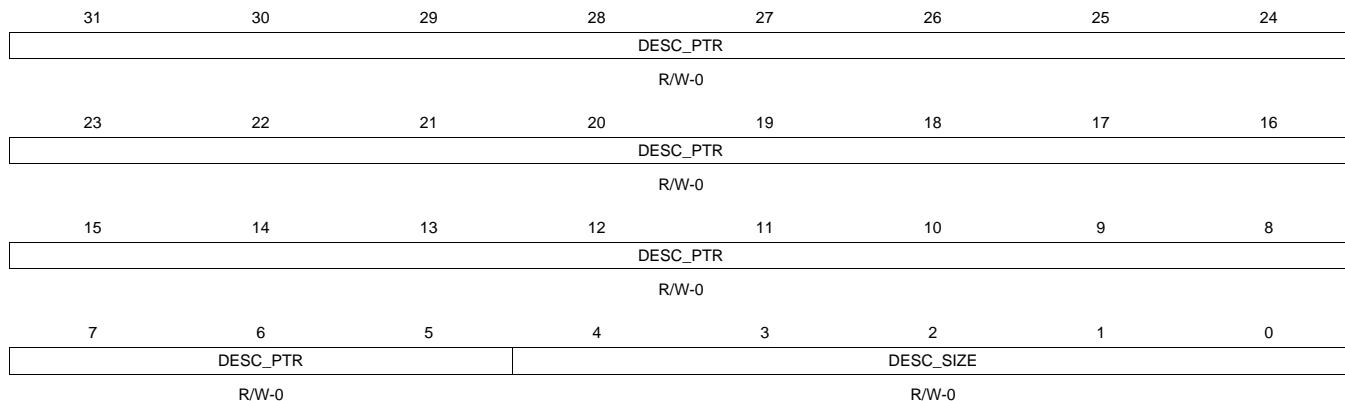
**Table 16-435. QUEUE\_27\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.146 QUEUE\_27\_D Register (offset = 21BCh) [reset = 0h]

QUEUE\_27\_D is shown in [Figure 16-422](#) and described in [Table 16-436](#).

**Figure 16-422. QUEUE\_27\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-436. QUEUE\_27\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.147 QUEUE\_28\_A Register (offset = 21C0h) [reset = 0h]**

QUEUE\_28\_A is shown in [Figure 16-423](#) and described in [Table 16-437](#).

**Figure 16-423. QUEUE\_28\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-437. QUEUE\_28\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.148 QUEUE\_28\_B Register (offset = 21C4h) [reset = 0h]**

QUEUE\_28\_B is shown in [Figure 16-424](#) and described in [Table 16-438](#).

**Figure 16-424. QUEUE\_28\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

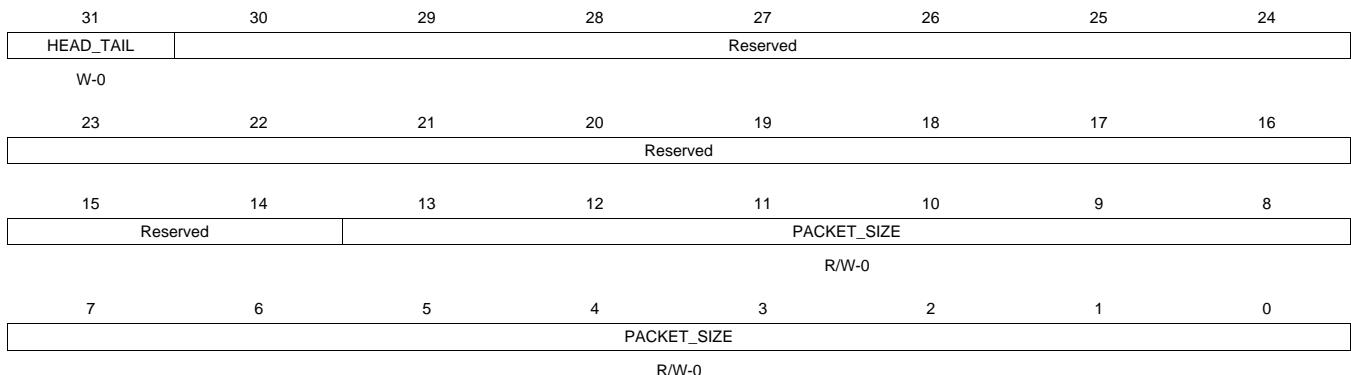
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-438. QUEUE\_28\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.149 QUEUE\_28\_C Register (offset = 21C8h) [reset = 0h]**

QUEUE\_28\_C is shown in [Figure 16-425](#) and described in [Table 16-439](#).

**Figure 16-425. QUEUE\_28\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

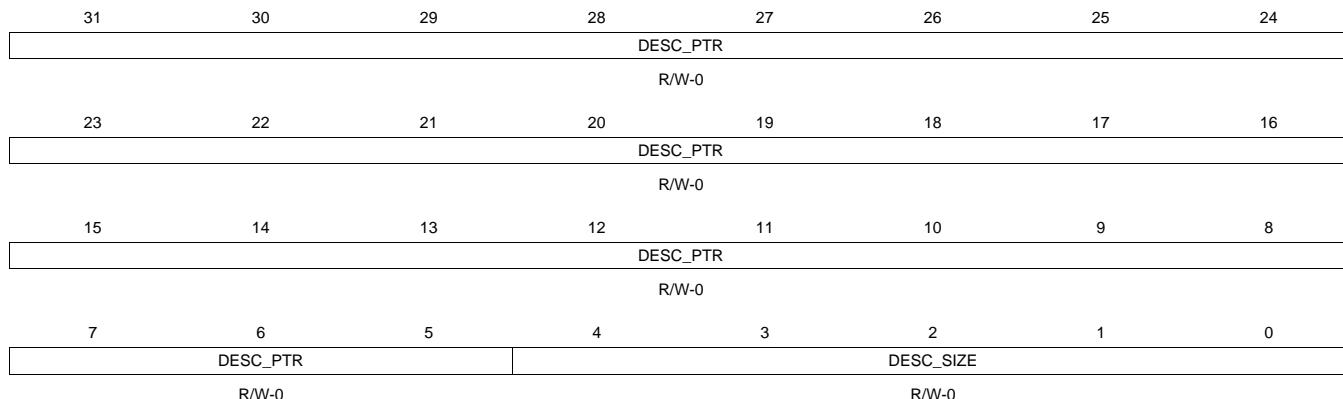
**Table 16-439. QUEUE\_28\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.150 QUEUE\_28\_D Register (offset = 21CCh) [reset = 0h]

QUEUE\_28\_D is shown in [Figure 16-426](#) and described in [Table 16-440](#).

**Figure 16-426. QUEUE\_28\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-440. QUEUE\_28\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.151 QUEUE\_29\_A Register (offset = 21D0h) [reset = 0h]**

QUEUE\_29\_A is shown in [Figure 16-427](#) and described in [Table 16-441](#).

**Figure 16-427. QUEUE\_29\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-441. QUEUE\_29\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.152 QUEUE\_29\_B Register (offset = 21D4h) [reset = 0h]**

QUEUE\_29\_B is shown in [Figure 16-428](#) and described in [Table 16-442](#).

**Figure 16-428. QUEUE\_29\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

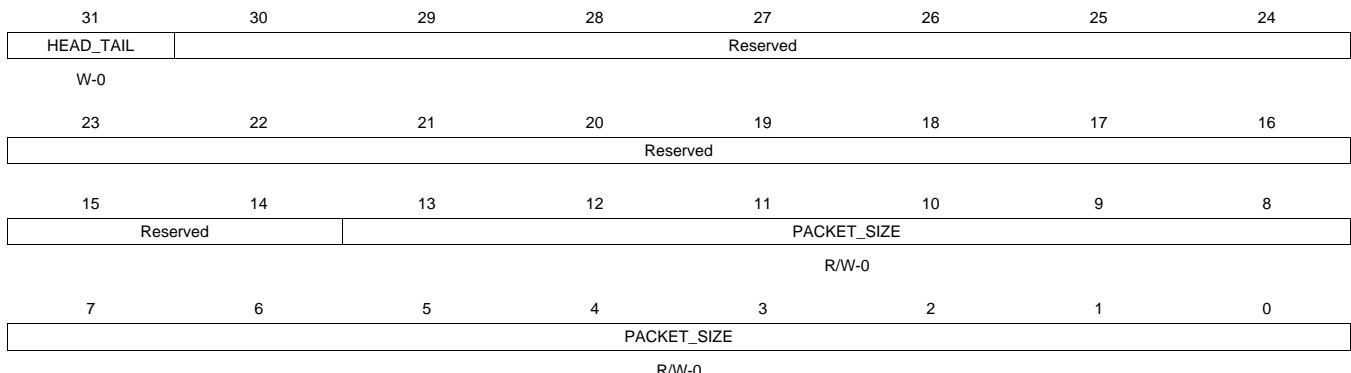
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-442. QUEUE\_29\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.153 QUEUE\_29\_C Register (offset = 21D8h) [reset = 0h]**

QUEUE\_29\_C is shown in [Figure 16-429](#) and described in [Table 16-443](#).

**Figure 16-429. QUEUE\_29\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

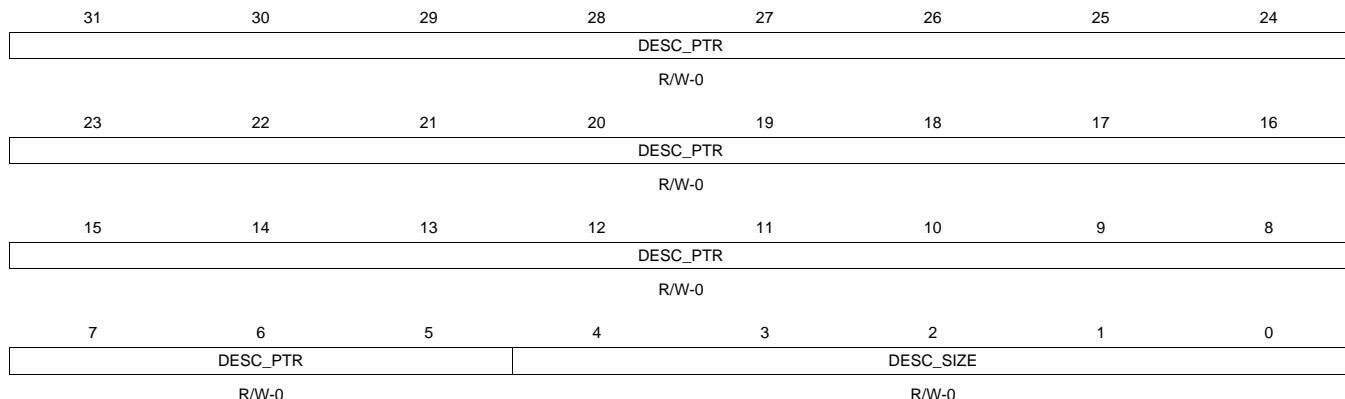
**Table 16-443. QUEUE\_29\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.154 QUEUE\_29\_D Register (offset = 21DCh) [reset = 0h]

QUEUE\_29\_D is shown in [Figure 16-430](#) and described in [Table 16-444](#).

**Figure 16-430. QUEUE\_29\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-444. QUEUE\_29\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.155 QUEUE\_30\_A Register (offset = 21E0h) [reset = 0h]**

QUEUE\_30\_A is shown in [Figure 16-431](#) and described in [Table 16-445](#).

**Figure 16-431. QUEUE\_30\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-445. QUEUE\_30\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.156 QUEUE\_30\_B Register (offset = 21E4h) [reset = 0h]**

QUEUE\_30\_B is shown in [Figure 16-432](#) and described in [Table 16-446](#).

**Figure 16-432. QUEUE\_30\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

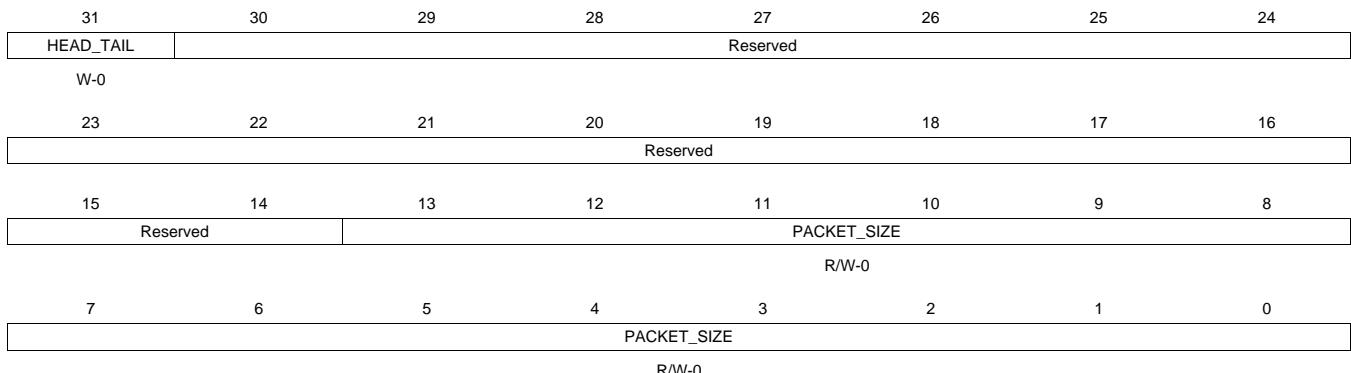
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-446. QUEUE\_30\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.157 QUEUE\_30\_C Register (offset = 21E8h) [reset = 0h]**

QUEUE\_30\_C is shown in [Figure 16-433](#) and described in [Table 16-447](#).

**Figure 16-433. QUEUE\_30\_C Register**


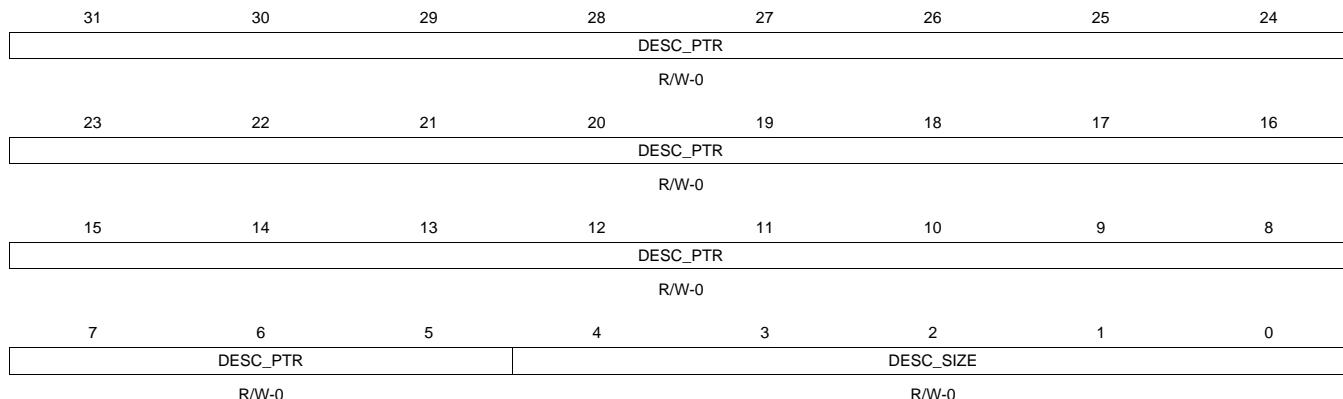
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-447. QUEUE\_30\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.158 QUEUE\_30\_D Register (offset = 21ECh) [reset = 0h]**

 QUEUE\_30\_D is shown in [Figure 16-434](#) and described in [Table 16-448](#).

**Figure 16-434. QUEUE\_30\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-448. QUEUE\_30\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.159 QUEUE\_31\_A Register (offset = 21F0h) [reset = 0h]**

QUEUE\_31\_A is shown in [Figure 16-435](#) and described in [Table 16-449](#).

**Figure 16-435. QUEUE\_31\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-449. QUEUE\_31\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.160 QUEUE\_31\_B Register (offset = 21F4h) [reset = 0h]**

QUEUE\_31\_B is shown in [Figure 16-436](#) and described in [Table 16-450](#).

**Figure 16-436. QUEUE\_31\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

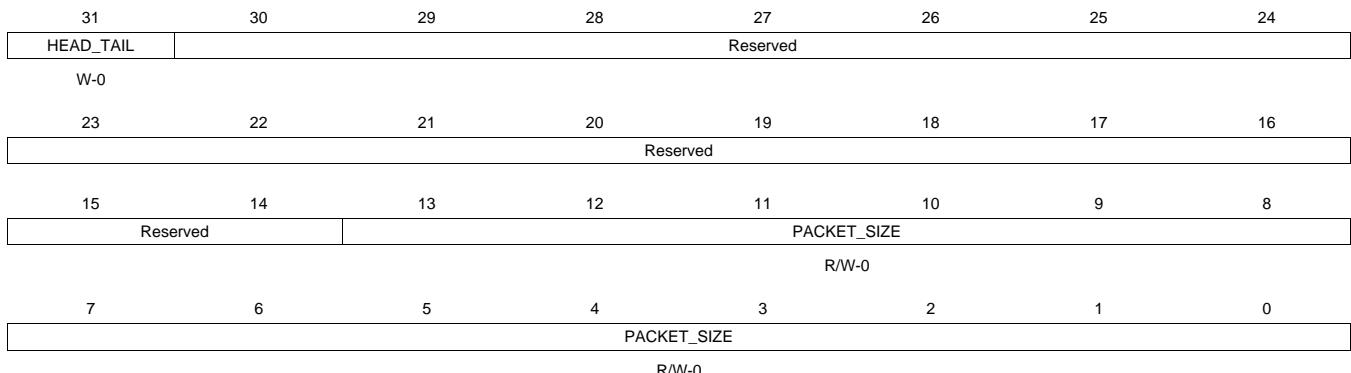
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-450. QUEUE\_31\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.161 QUEUE\_31\_C Register (offset = 21F8h) [reset = 0h]**

QUEUE\_31\_C is shown in [Figure 16-437](#) and described in [Table 16-451](#).

**Figure 16-437. QUEUE\_31\_C Register**


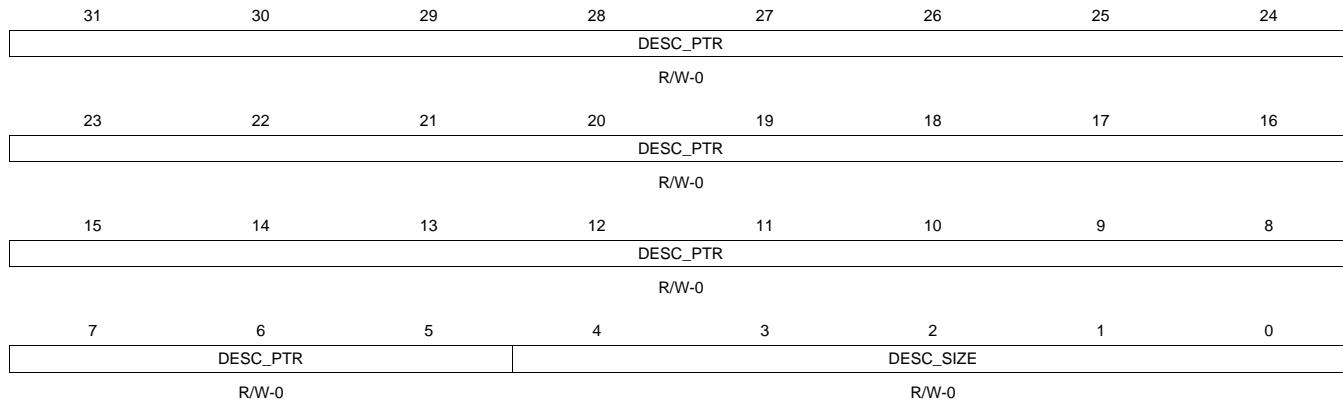
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-451. QUEUE\_31\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.162 QUEUE\_31\_D Register (offset = 21FCh) [reset = 0h]**

 QUEUE\_31\_D is shown in [Figure 16-438](#) and described in [Table 16-452](#).

**Figure 16-438. QUEUE\_31\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-452. QUEUE\_31\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.163 QUEUE\_32\_A Register (offset = 2200h) [reset = 0h]**

QUEUE\_32\_A is shown in [Figure 16-439](#) and described in [Table 16-453](#).

**Figure 16-439. QUEUE\_32\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-453. QUEUE\_32\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.164 QUEUE\_32\_B Register (offset = 2204h) [reset = 0h]**

QUEUE\_32\_B is shown in [Figure 16-440](#) and described in [Table 16-454](#).

**Figure 16-440. QUEUE\_32\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

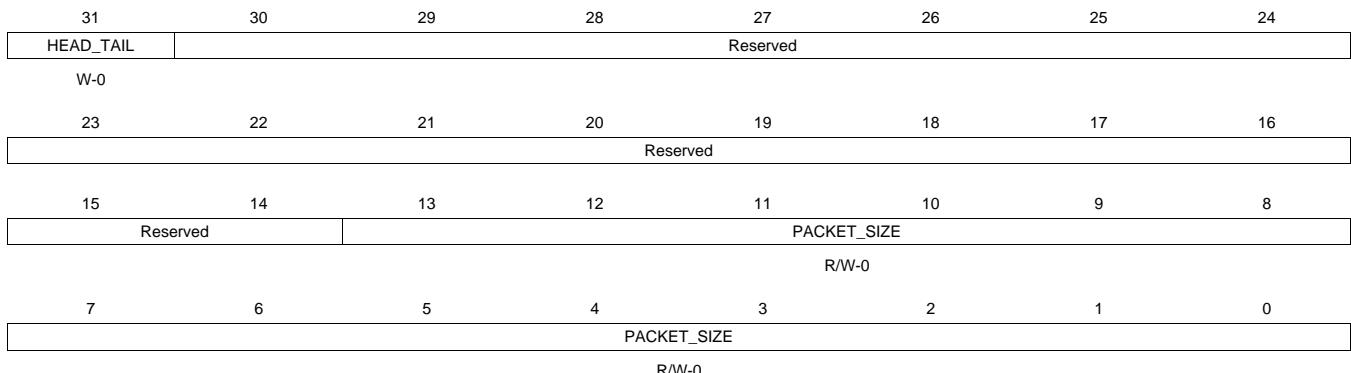
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-454. QUEUE\_32\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.165 QUEUE\_32\_C Register (offset = 2208h) [reset = 0h]**

QUEUE\_32\_C is shown in [Figure 16-441](#) and described in [Table 16-455](#).

**Figure 16-441. QUEUE\_32\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

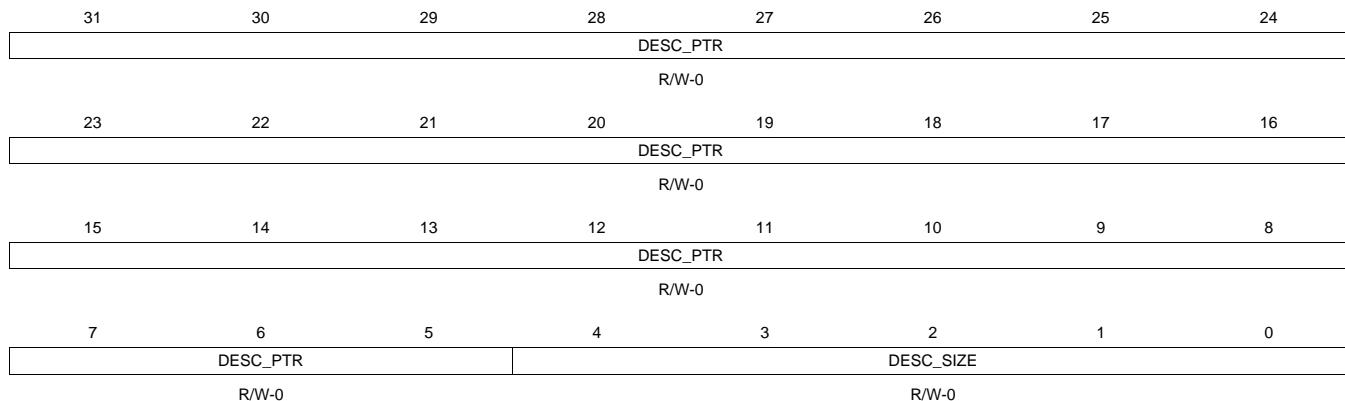
**Table 16-455. QUEUE\_32\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.166 QUEUE\_32\_D Register (offset = 220Ch) [reset = 0h]

QUEUE\_32\_D is shown in [Figure 16-442](#) and described in [Table 16-456](#).

**Figure 16-442. QUEUE\_32\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-456. QUEUE\_32\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.167 QUEUE\_33\_A Register (offset = 2210h) [reset = 0h]**

QUEUE\_33\_A is shown in [Figure 16-443](#) and described in [Table 16-457](#).

**Figure 16-443. QUEUE\_33\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-457. QUEUE\_33\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.168 QUEUE\_33\_B Register (offset = 2214h) [reset = 0h]**

QUEUE\_33\_B is shown in [Figure 16-444](#) and described in [Table 16-458](#).

**Figure 16-444. QUEUE\_33\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

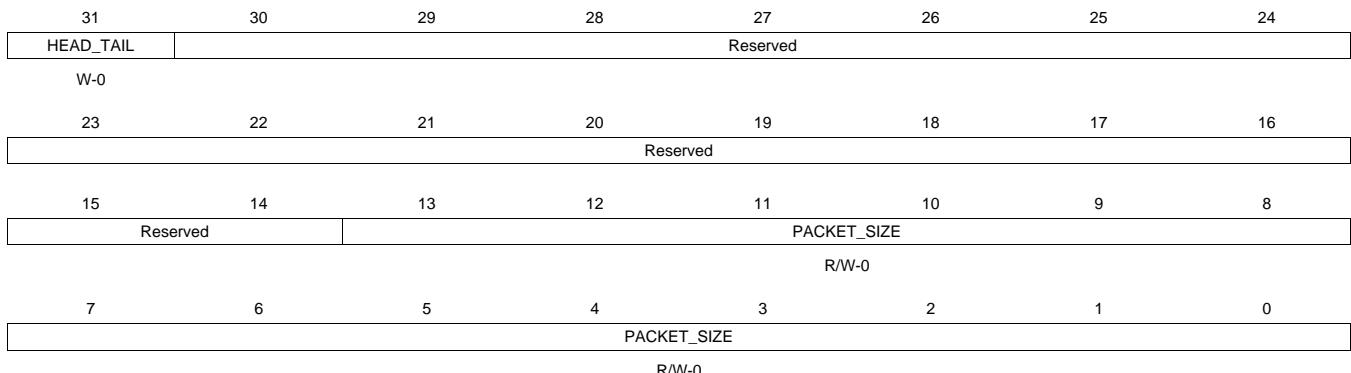
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-458. QUEUE\_33\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.169 QUEUE\_33\_C Register (offset = 2218h) [reset = 0h]**

QUEUE\_33\_C is shown in [Figure 16-445](#) and described in [Table 16-459](#).

**Figure 16-445. QUEUE\_33\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

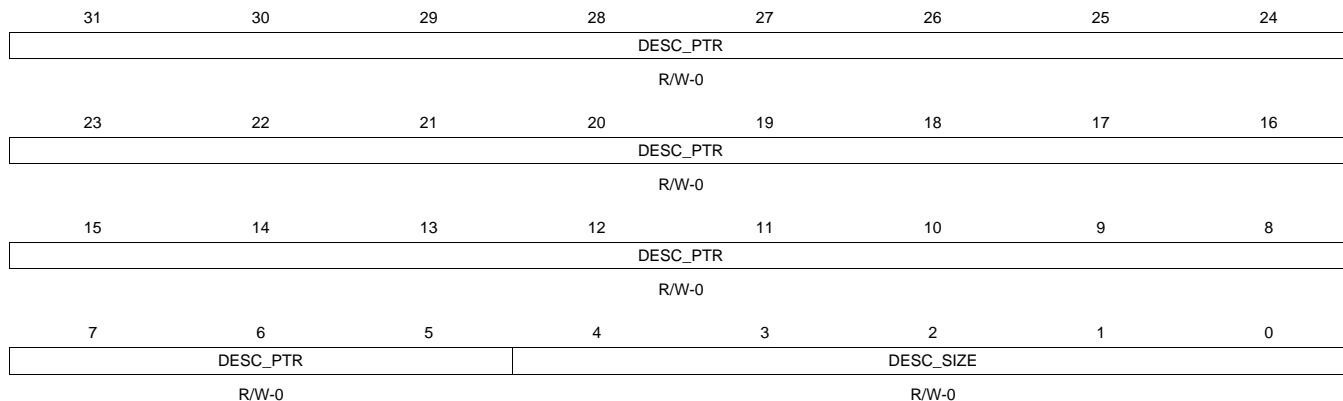
**Table 16-459. QUEUE\_33\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.170 QUEUE\_33\_D Register (offset = 221Ch) [reset = 0h]

QUEUE\_33\_D is shown in [Figure 16-446](#) and described in [Table 16-460](#).

**Figure 16-446. QUEUE\_33\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-460. QUEUE\_33\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.171 QUEUE\_34\_A Register (offset = 2220h) [reset = 0h]**

QUEUE\_34\_A is shown in [Figure 16-447](#) and described in [Table 16-461](#).

**Figure 16-447. QUEUE\_34\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-461. QUEUE\_34\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.172 QUEUE\_34\_B Register (offset = 2224h) [reset = 0h]**

QUEUE\_34\_B is shown in [Figure 16-448](#) and described in [Table 16-462](#).

**Figure 16-448. QUEUE\_34\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

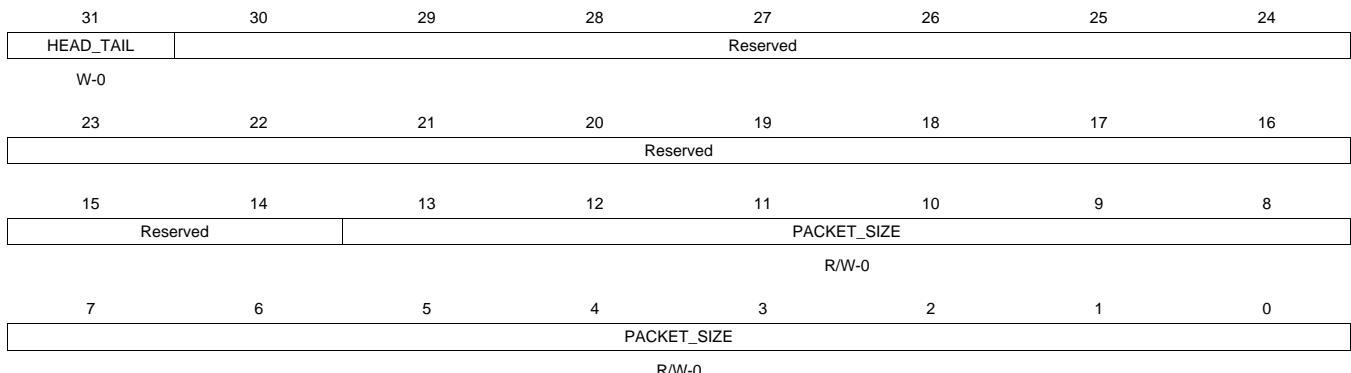
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-462. QUEUE\_34\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.173 QUEUE\_34\_C Register (offset = 2228h) [reset = 0h]**

QUEUE\_34\_C is shown in [Figure 16-449](#) and described in [Table 16-463](#).

**Figure 16-449. QUEUE\_34\_C Register**


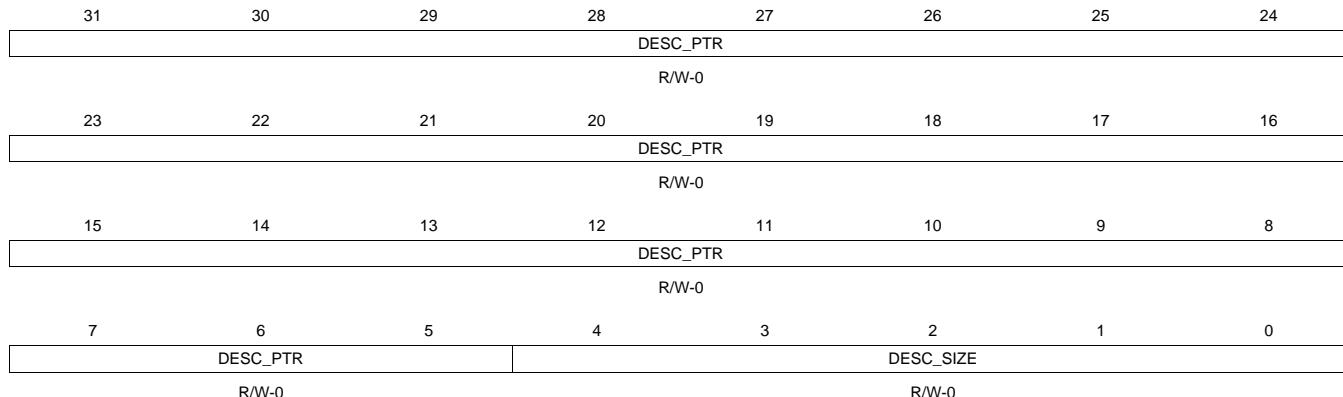
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-463. QUEUE\_34\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.174 QUEUE\_34\_D Register (offset = 222Ch) [reset = 0h]**

 QUEUE\_34\_D is shown in [Figure 16-450](#) and described in [Table 16-464](#).

**Figure 16-450. QUEUE\_34\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-464. QUEUE\_34\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.175 QUEUE\_35\_A Register (offset = 2230h) [reset = 0h]**

QUEUE\_35\_A is shown in [Figure 16-451](#) and described in [Table 16-465](#).

**Figure 16-451. QUEUE\_35\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-465. QUEUE\_35\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.176 QUEUE\_35\_B Register (offset = 2234h) [reset = 0h]**

QUEUE\_35\_B is shown in [Figure 16-452](#) and described in [Table 16-466](#).

**Figure 16-452. QUEUE\_35\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

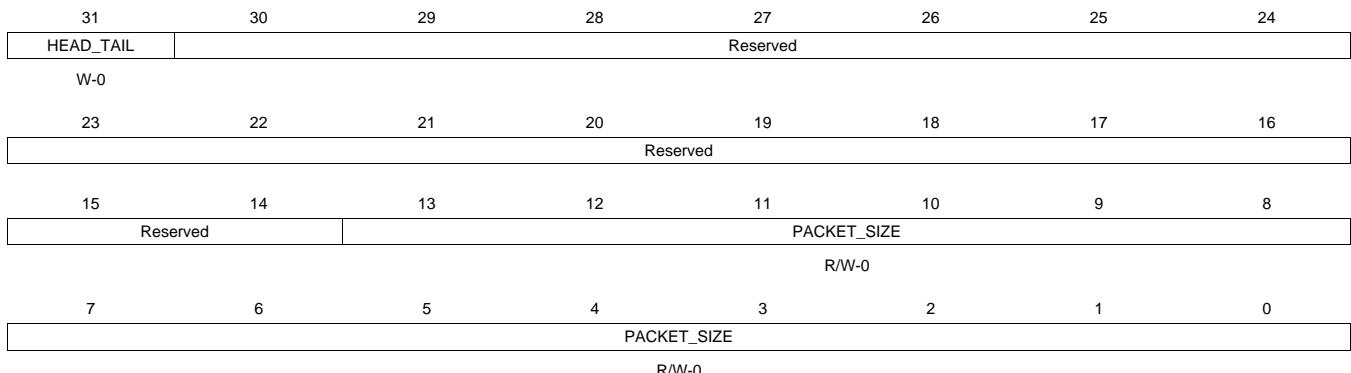
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-466. QUEUE\_35\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.177 QUEUE\_35\_C Register (offset = 2238h) [reset = 0h]**

QUEUE\_35\_C is shown in [Figure 16-453](#) and described in [Table 16-467](#).

**Figure 16-453. QUEUE\_35\_C Register**


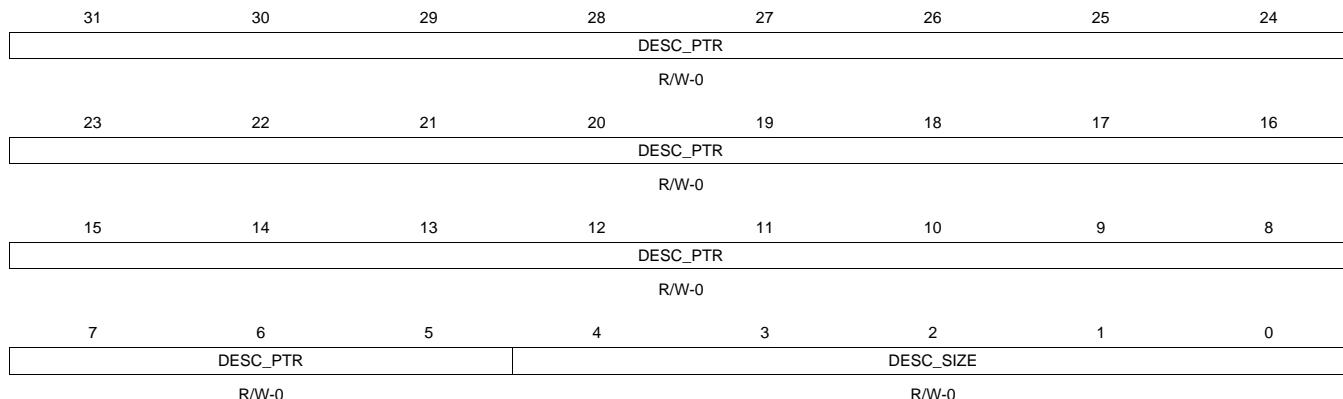
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-467. QUEUE\_35\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.178 QUEUE\_35\_D Register (offset = 223Ch) [reset = 0h]**

QUEUE\_35\_D is shown in [Figure 16-454](#) and described in [Table 16-468](#).

**Figure 16-454. QUEUE\_35\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-468. QUEUE\_35\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.179 QUEUE\_36\_A Register (offset = 2240h) [reset = 0h]**

QUEUE\_36\_A is shown in [Figure 16-455](#) and described in [Table 16-469](#).

**Figure 16-455. QUEUE\_36\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-469. QUEUE\_36\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.180 QUEUE\_36\_B Register (offset = 2244h) [reset = 0h]**

QUEUE\_36\_B is shown in [Figure 16-456](#) and described in [Table 16-470](#).

**Figure 16-456. QUEUE\_36\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

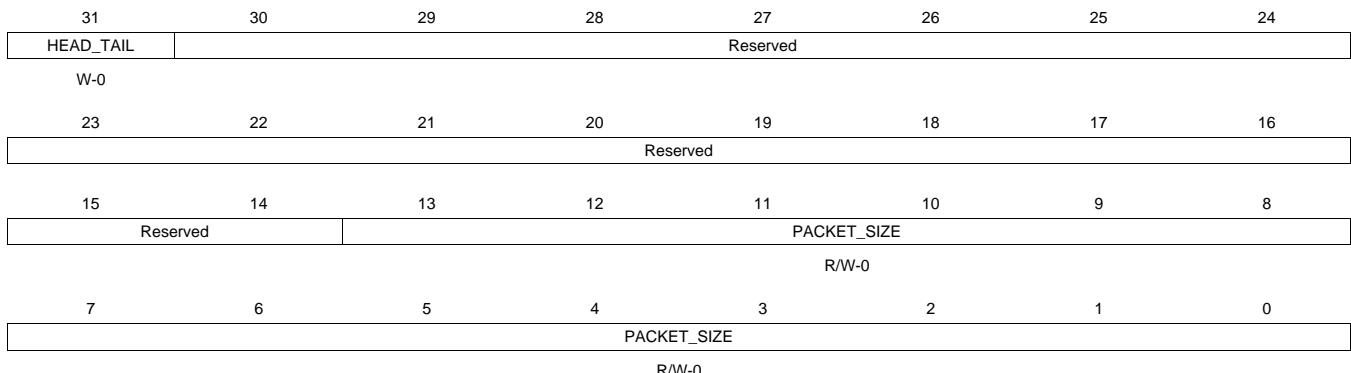
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-470. QUEUE\_36\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.181 QUEUE\_36\_C Register (offset = 2248h) [reset = 0h]**

QUEUE\_36\_C is shown in [Figure 16-457](#) and described in [Table 16-471](#).

**Figure 16-457. QUEUE\_36\_C Register**


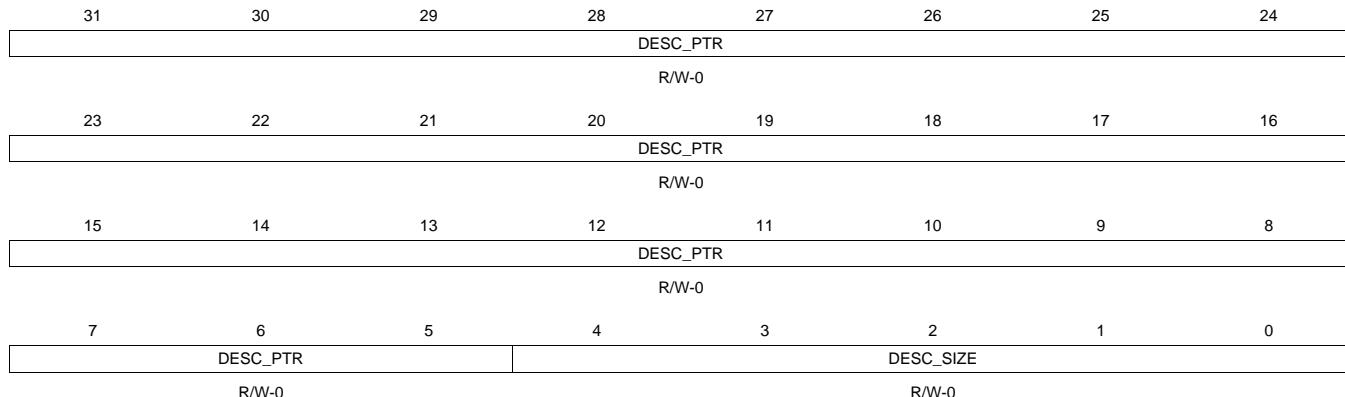
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-471. QUEUE\_36\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.182 QUEUE\_36\_D Register (offset = 224Ch) [reset = 0h]**

 QUEUE\_36\_D is shown in [Figure 16-458](#) and described in [Table 16-472](#).

**Figure 16-458. QUEUE\_36\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-472. QUEUE\_36\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.183 QUEUE\_37\_A Register (offset = 2250h) [reset = 0h]**

QUEUE\_37\_A is shown in [Figure 16-459](#) and described in [Table 16-473](#).

**Figure 16-459. QUEUE\_37\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-473. QUEUE\_37\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.184 QUEUE\_37\_B Register (offset = 2254h) [reset = 0h]**

QUEUE\_37\_B is shown in [Figure 16-460](#) and described in [Table 16-474](#).

**Figure 16-460. QUEUE\_37\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

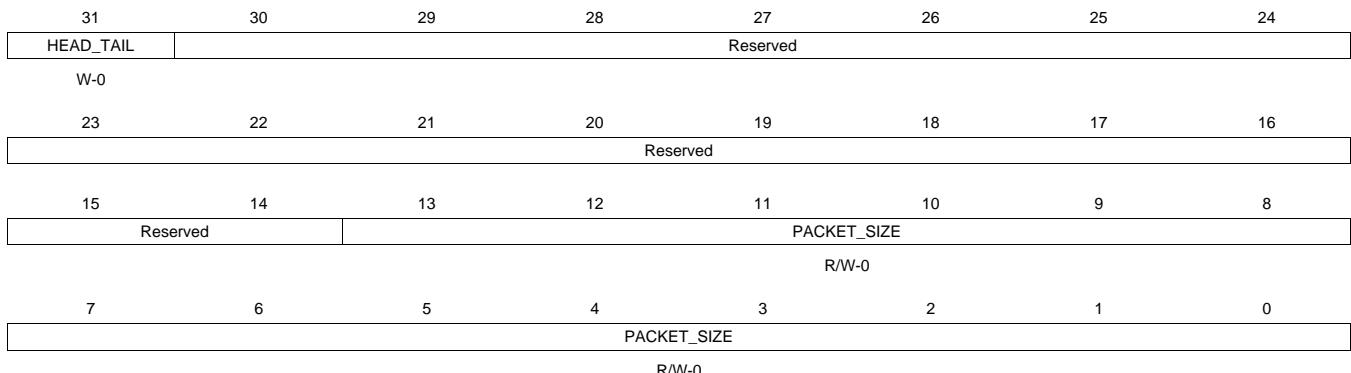
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-474. QUEUE\_37\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.185 QUEUE\_37\_C Register (offset = 2258h) [reset = 0h]**

QUEUE\_37\_C is shown in [Figure 16-461](#) and described in [Table 16-475](#).

**Figure 16-461. QUEUE\_37\_C Register**


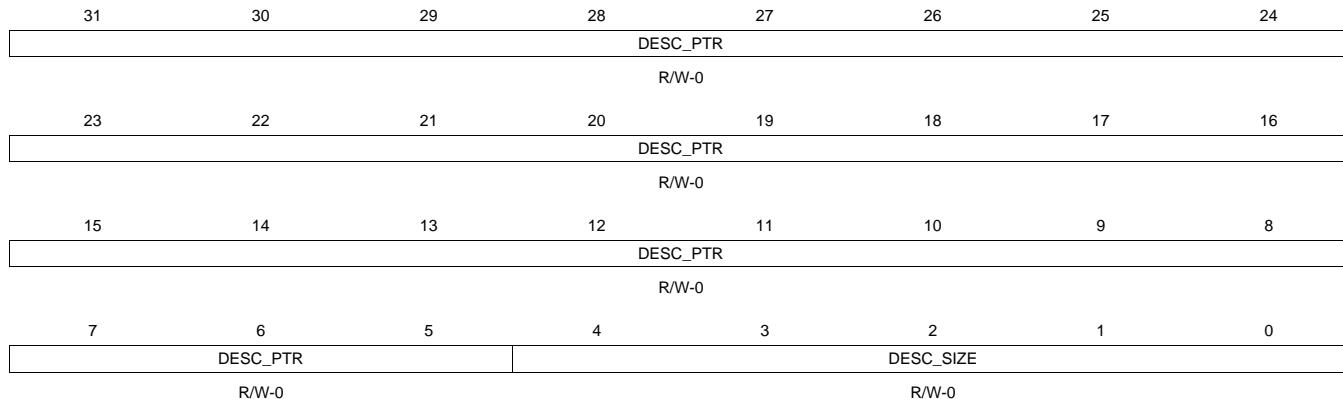
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-475. QUEUE\_37\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.186 QUEUE\_37\_D Register (offset = 225Ch) [reset = 0h]**

QUEUE\_37\_D is shown in [Figure 16-462](#) and described in [Table 16-476](#).

**Figure 16-462. QUEUE\_37\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-476. QUEUE\_37\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.187 QUEUE\_38\_A Register (offset = 2260h) [reset = 0h]**

QUEUE\_38\_A is shown in [Figure 16-463](#) and described in [Table 16-477](#).

**Figure 16-463. QUEUE\_38\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-477. QUEUE\_38\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.188 QUEUE\_38\_B Register (offset = 2264h) [reset = 0h]**

QUEUE\_38\_B is shown in [Figure 16-464](#) and described in [Table 16-478](#).

**Figure 16-464. QUEUE\_38\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

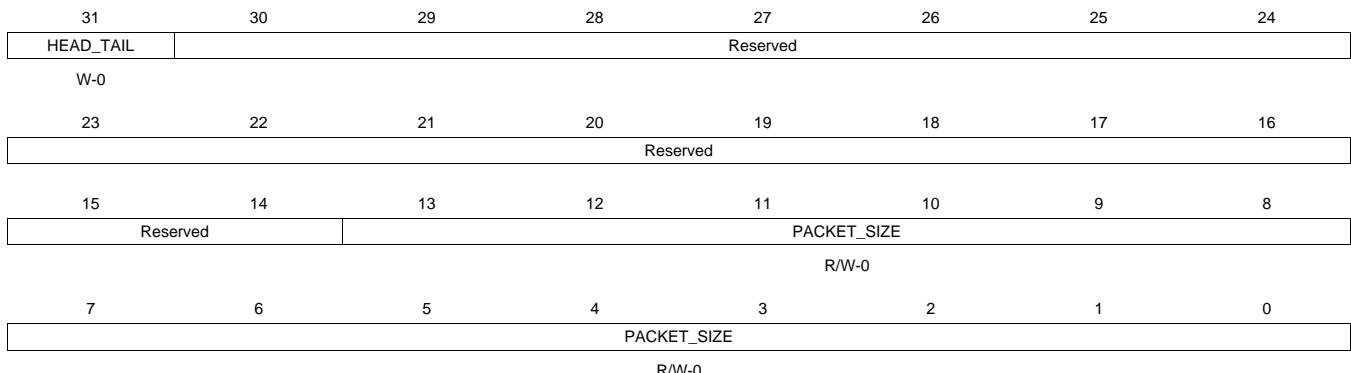
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-478. QUEUE\_38\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.189 QUEUE\_38\_C Register (offset = 2268h) [reset = 0h]**

QUEUE\_38\_C is shown in [Figure 16-465](#) and described in [Table 16-479](#).

**Figure 16-465. QUEUE\_38\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

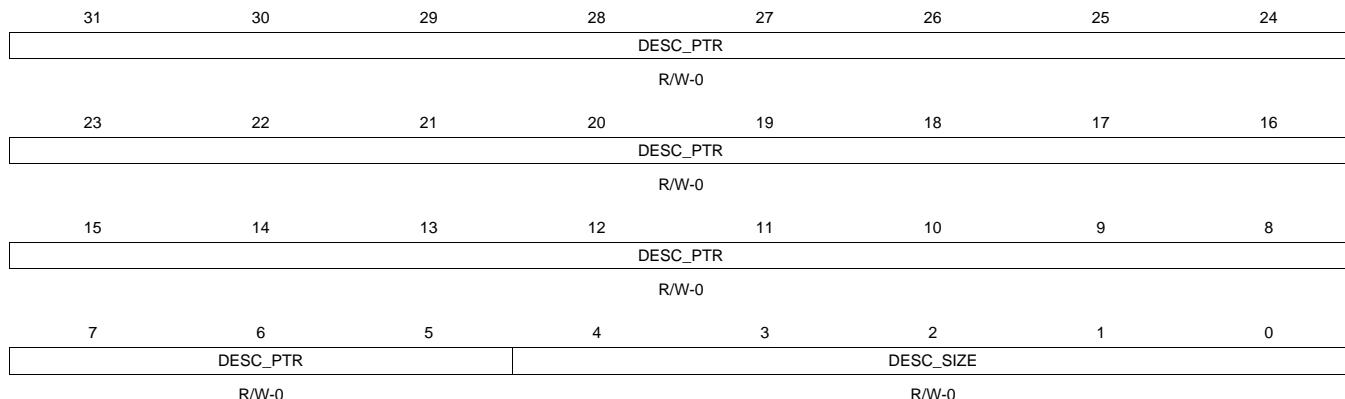
**Table 16-479. QUEUE\_38\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.190 QUEUE\_38\_D Register (offset = 226Ch) [reset = 0h]

QUEUE\_38\_D is shown in [Figure 16-466](#) and described in [Table 16-480](#).

**Figure 16-466. QUEUE\_38\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-480. QUEUE\_38\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.191 QUEUE\_39\_A Register (offset = 2270h) [reset = 0h]**

QUEUE\_39\_A is shown in [Figure 16-467](#) and described in [Table 16-481](#).

**Figure 16-467. QUEUE\_39\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-481. QUEUE\_39\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.192 QUEUE\_39\_B Register (offset = 2274h) [reset = 0h]**

QUEUE\_39\_B is shown in [Figure 16-468](#) and described in [Table 16-482](#).

**Figure 16-468. QUEUE\_39\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

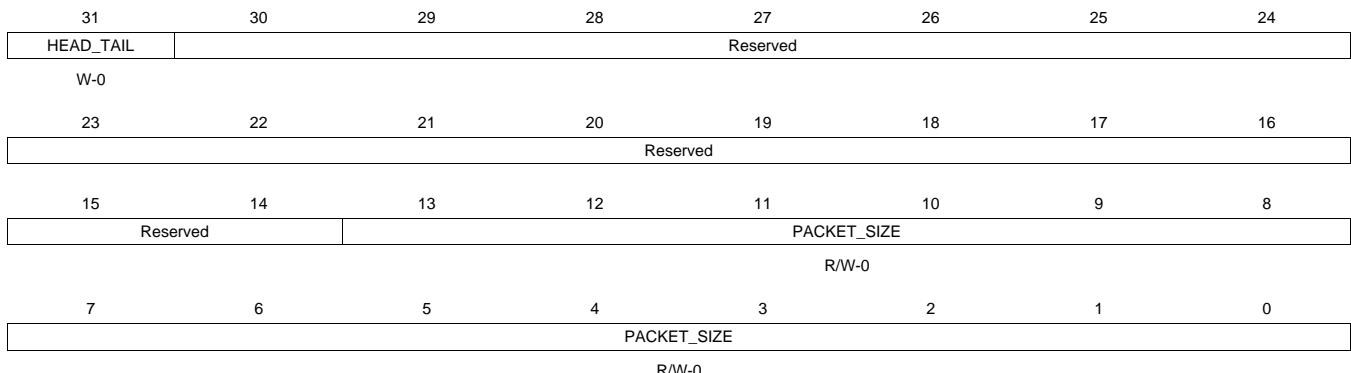
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-482. QUEUE\_39\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.193 QUEUE\_39\_C Register (offset = 2278h) [reset = 0h]**

QUEUE\_39\_C is shown in [Figure 16-469](#) and described in [Table 16-483](#).

**Figure 16-469. QUEUE\_39\_C Register**


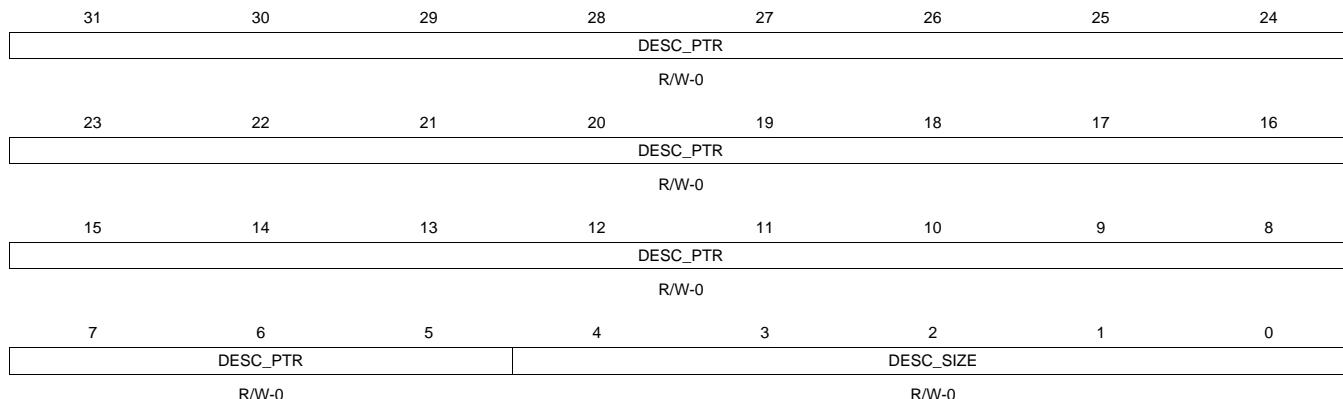
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-483. QUEUE\_39\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.194 QUEUE\_39\_D Register (offset = 227Ch) [reset = 0h]**

 QUEUE\_39\_D is shown in [Figure 16-470](#) and described in [Table 16-484](#).

**Figure 16-470. QUEUE\_39\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-484. QUEUE\_39\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.195 QUEUE\_40\_A Register (offset = 2280h) [reset = 0h]**

QUEUE\_40\_A is shown in [Figure 16-471](#) and described in [Table 16-485](#).

**Figure 16-471. QUEUE\_40\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-485. QUEUE\_40\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.196 QUEUE\_40\_B Register (offset = 2284h) [reset = 0h]**

QUEUE\_40\_B is shown in [Figure 16-472](#) and described in [Table 16-486](#).

**Figure 16-472. QUEUE\_40\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

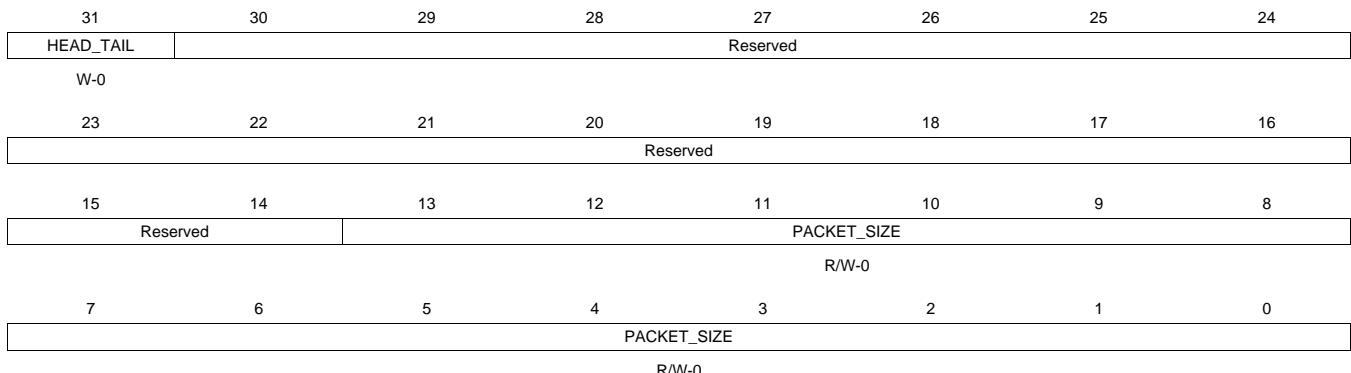
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-486. QUEUE\_40\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.197 QUEUE\_40\_C Register (offset = 2288h) [reset = 0h]**

QUEUE\_40\_C is shown in [Figure 16-473](#) and described in [Table 16-487](#).

**Figure 16-473. QUEUE\_40\_C Register**


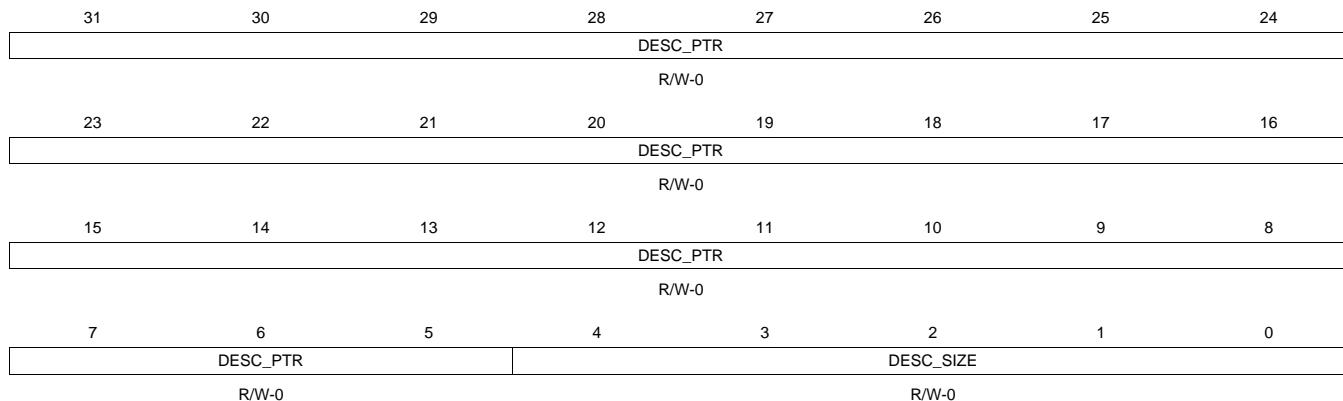
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-487. QUEUE\_40\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.198 QUEUE\_40\_D Register (offset = 228Ch) [reset = 0h]**

 QUEUE\_40\_D is shown in [Figure 16-474](#) and described in [Table 16-488](#).

**Figure 16-474. QUEUE\_40\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-488. QUEUE\_40\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.199 QUEUE\_41\_A Register (offset = 2290h) [reset = 0h]**

QUEUE\_41\_A is shown in [Figure 16-475](#) and described in [Table 16-489](#).

**Figure 16-475. QUEUE\_41\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-489. QUEUE\_41\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.200 QUEUE\_41\_B Register (offset = 2294h) [reset = 0h]**

QUEUE\_41\_B is shown in [Figure 16-476](#) and described in [Table 16-490](#).

**Figure 16-476. QUEUE\_41\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

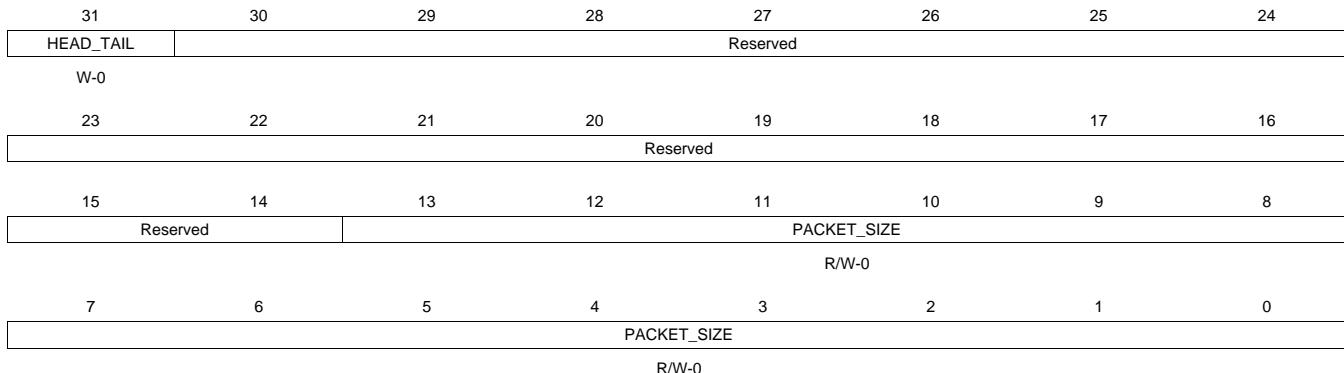
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-490. QUEUE\_41\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.201 QUEUE\_41\_C Register (offset = 2298h) [reset = 0h]**

QUEUE\_41\_C is shown in [Figure 16-477](#) and described in [Table 16-491](#).

**Figure 16-477. QUEUE\_41\_C Register**


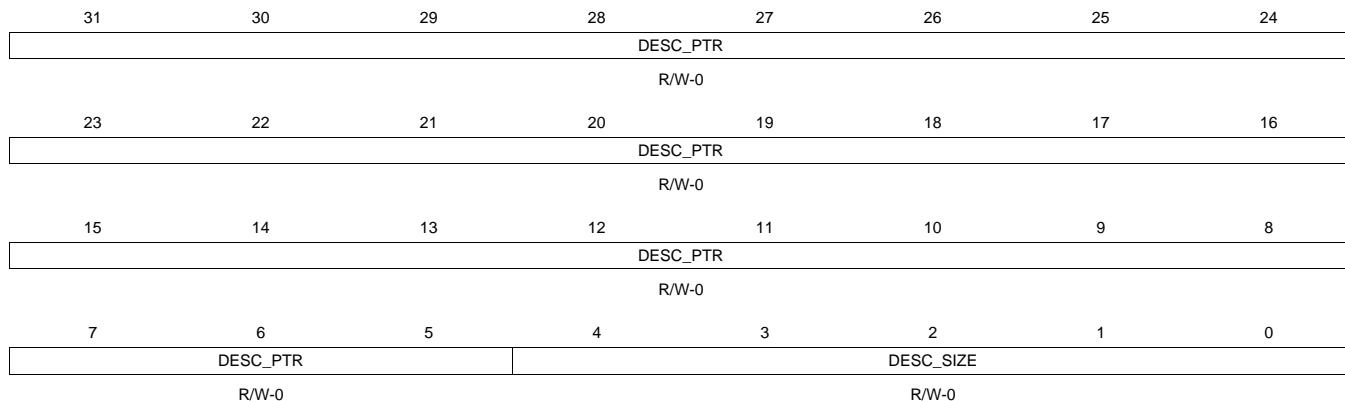
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-491. QUEUE\_41\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.202 QUEUE\_41\_D Register (offset = 229Ch) [reset = 0h]**

 QUEUE\_41\_D is shown in [Figure 16-478](#) and described in [Table 16-492](#).

**Figure 16-478. QUEUE\_41\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-492. QUEUE\_41\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.203 QUEUE\_42\_A Register (offset = 22A0h) [reset = 0h]**

QUEUE\_42\_A is shown in [Figure 16-479](#) and described in [Table 16-493](#).

**Figure 16-479. QUEUE\_42\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-493. QUEUE\_42\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.204 QUEUE\_42\_B Register (offset = 22A4h) [reset = 0h]**

QUEUE\_42\_B is shown in [Figure 16-480](#) and described in [Table 16-494](#).

**Figure 16-480. QUEUE\_42\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

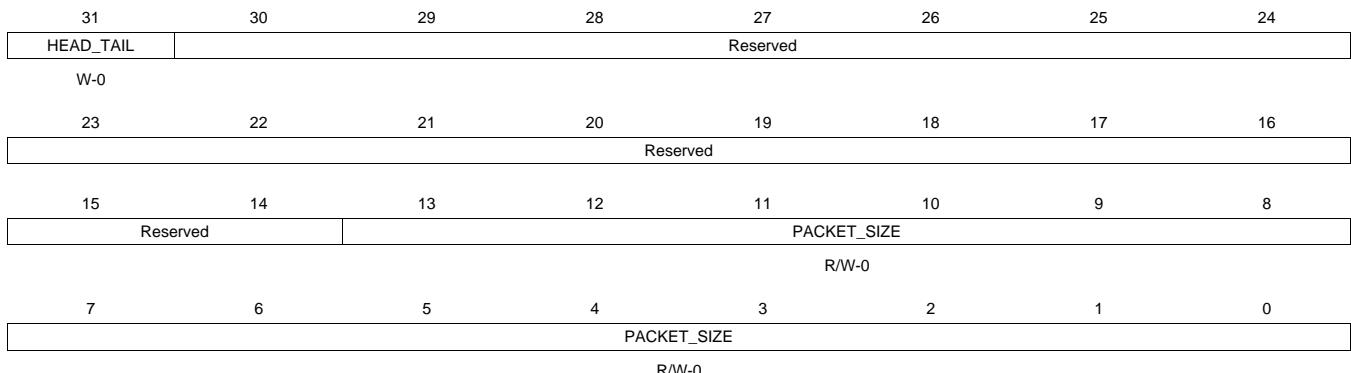
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-494. QUEUE\_42\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.205 QUEUE\_42\_C Register (offset = 22A8h) [reset = 0h]**

QUEUE\_42\_C is shown in [Figure 16-481](#) and described in [Table 16-495](#).

**Figure 16-481. QUEUE\_42\_C Register**


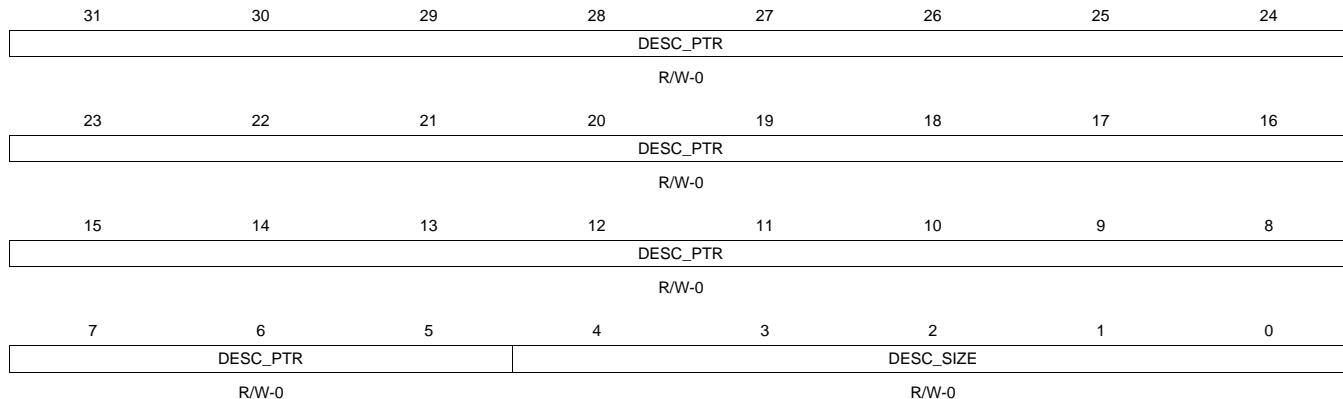
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-495. QUEUE\_42\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.206 QUEUE\_42\_D Register (offset = 22ACh) [reset = 0h]**

 QUEUE\_42\_D is shown in [Figure 16-482](#) and described in [Table 16-496](#).

**Figure 16-482. QUEUE\_42\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-496. QUEUE\_42\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.207 QUEUE\_43\_A Register (offset = 22B0h) [reset = 0h]**

QUEUE\_43\_A is shown in [Figure 16-483](#) and described in [Table 16-497](#).

**Figure 16-483. QUEUE\_43\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-497. QUEUE\_43\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.208 QUEUE\_43\_B Register (offset = 22B4h) [reset = 0h]**

QUEUE\_43\_B is shown in [Figure 16-484](#) and described in [Table 16-498](#).

**Figure 16-484. QUEUE\_43\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

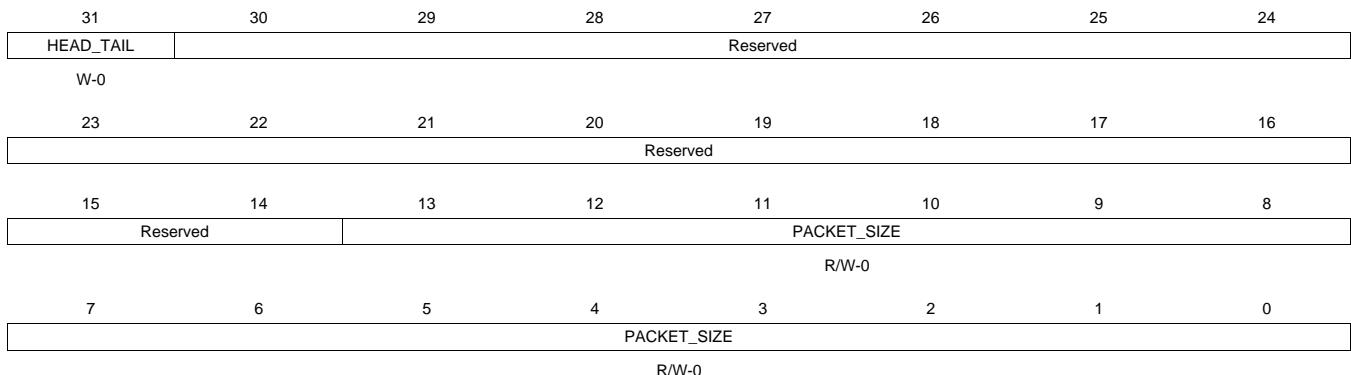
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-498. QUEUE\_43\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.209 QUEUE\_43\_C Register (offset = 22B8h) [reset = 0h]**

QUEUE\_43\_C is shown in [Figure 16-485](#) and described in [Table 16-499](#).

**Figure 16-485. QUEUE\_43\_C Register**


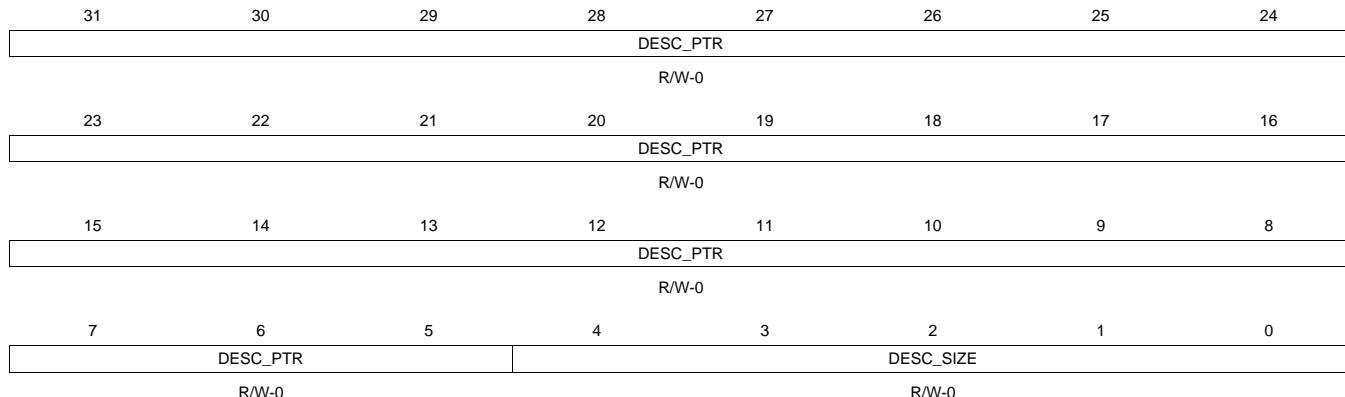
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-499. QUEUE\_43\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.210 QUEUE\_43\_D Register (offset = 22BCh) [reset = 0h]**

 QUEUE\_43\_D is shown in [Figure 16-486](#) and described in [Table 16-500](#).

**Figure 16-486. QUEUE\_43\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-500. QUEUE\_43\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.211 QUEUE\_44\_A Register (offset = 22C0h) [reset = 0h]**

QUEUE\_44\_A is shown in [Figure 16-487](#) and described in [Table 16-501](#).

**Figure 16-487. QUEUE\_44\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-501. QUEUE\_44\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.212 QUEUE\_44\_B Register (offset = 22C4h) [reset = 0h]**

QUEUE\_44\_B is shown in [Figure 16-488](#) and described in [Table 16-502](#).

**Figure 16-488. QUEUE\_44\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

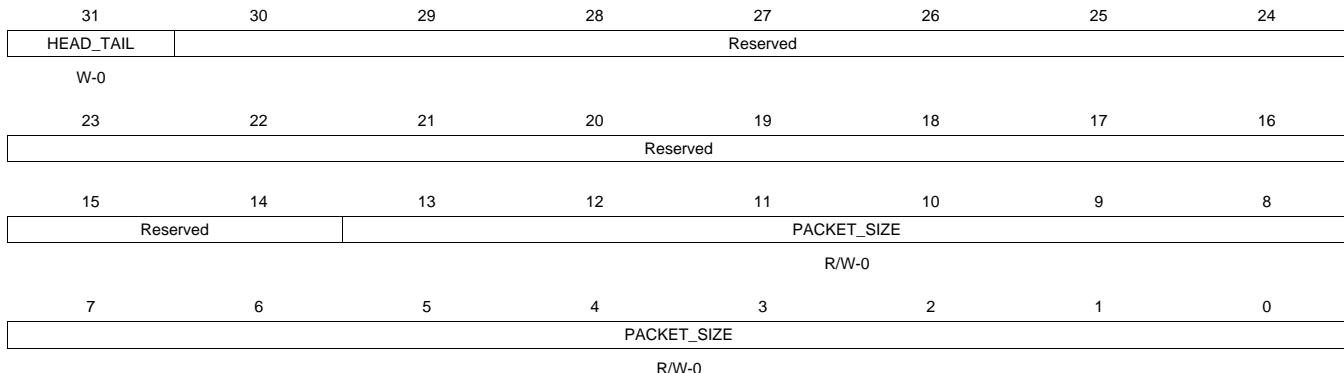
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-502. QUEUE\_44\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.213 QUEUE\_44\_C Register (offset = 22C8h) [reset = 0h]**

QUEUE\_44\_C is shown in [Figure 16-489](#) and described in [Table 16-503](#).

**Figure 16-489. QUEUE\_44\_C Register**


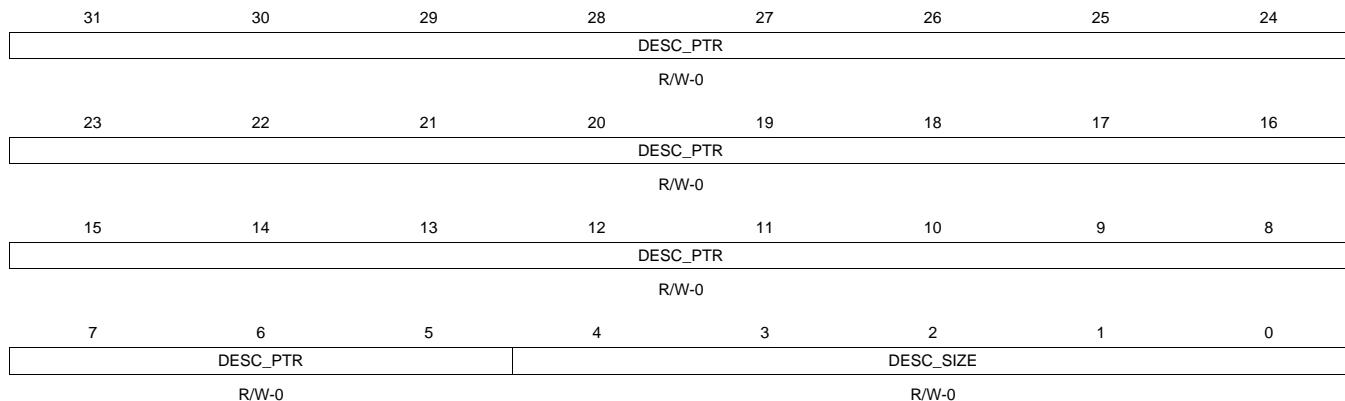
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-503. QUEUE\_44\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.214 QUEUE\_44\_D Register (offset = 22CCh) [reset = 0h]**

 QUEUE\_44\_D is shown in [Figure 16-490](#) and described in [Table 16-504](#).

**Figure 16-490. QUEUE\_44\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-504. QUEUE\_44\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.215 QUEUE\_45\_A Register (offset = 22D0h) [reset = 0h]**

QUEUE\_45\_A is shown in [Figure 16-491](#) and described in [Table 16-505](#).

**Figure 16-491. QUEUE\_45\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-505. QUEUE\_45\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.216 QUEUE\_45\_B Register (offset = 22D4h) [reset = 0h]**

QUEUE\_45\_B is shown in [Figure 16-492](#) and described in [Table 16-506](#).

**Figure 16-492. QUEUE\_45\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

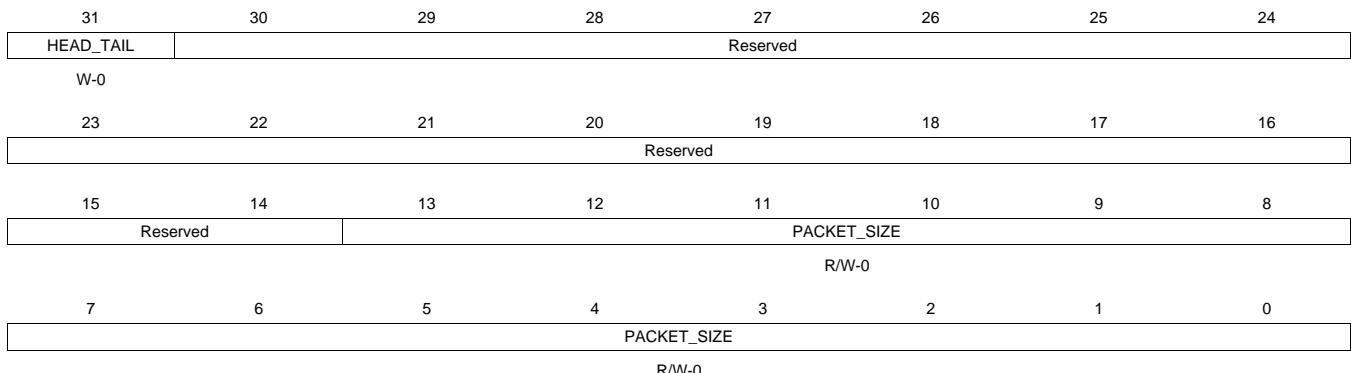
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-506. QUEUE\_45\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.217 QUEUE\_45\_C Register (offset = 22D8h) [reset = 0h]**

QUEUE\_45\_C is shown in [Figure 16-493](#) and described in [Table 16-507](#).

**Figure 16-493. QUEUE\_45\_C Register**


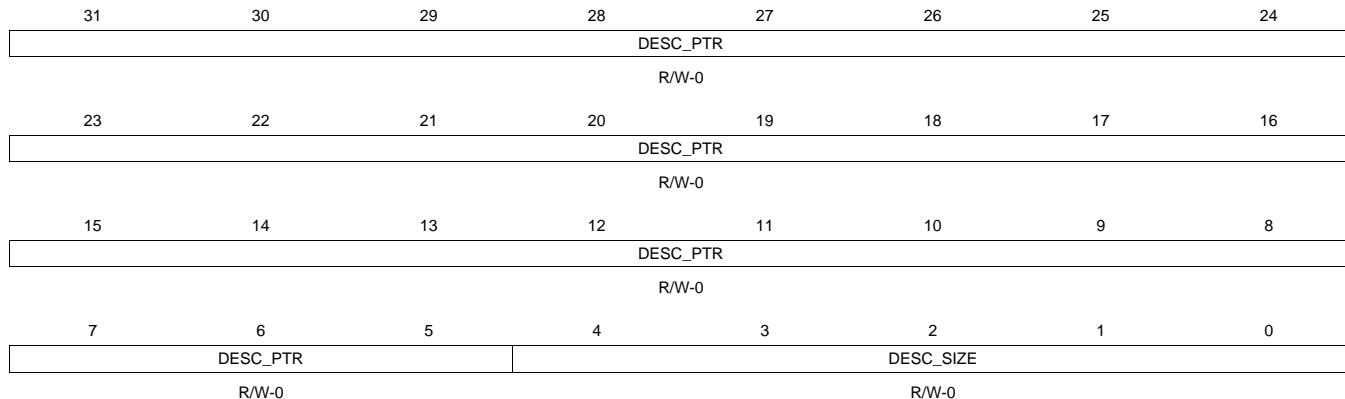
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-507. QUEUE\_45\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.218 QUEUE\_45\_D Register (offset = 22DCh) [reset = 0h]**

 QUEUE\_45\_D is shown in [Figure 16-494](#) and described in [Table 16-508](#).

**Figure 16-494. QUEUE\_45\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-508. QUEUE\_45\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.219 QUEUE\_46\_A Register (offset = 22E0h) [reset = 0h]**

QUEUE\_46\_A is shown in [Figure 16-495](#) and described in [Table 16-509](#).

**Figure 16-495. QUEUE\_46\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-509. QUEUE\_46\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.220 QUEUE\_46\_B Register (offset = 22E4h) [reset = 0h]**

QUEUE\_46\_B is shown in [Figure 16-496](#) and described in [Table 16-510](#).

**Figure 16-496. QUEUE\_46\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

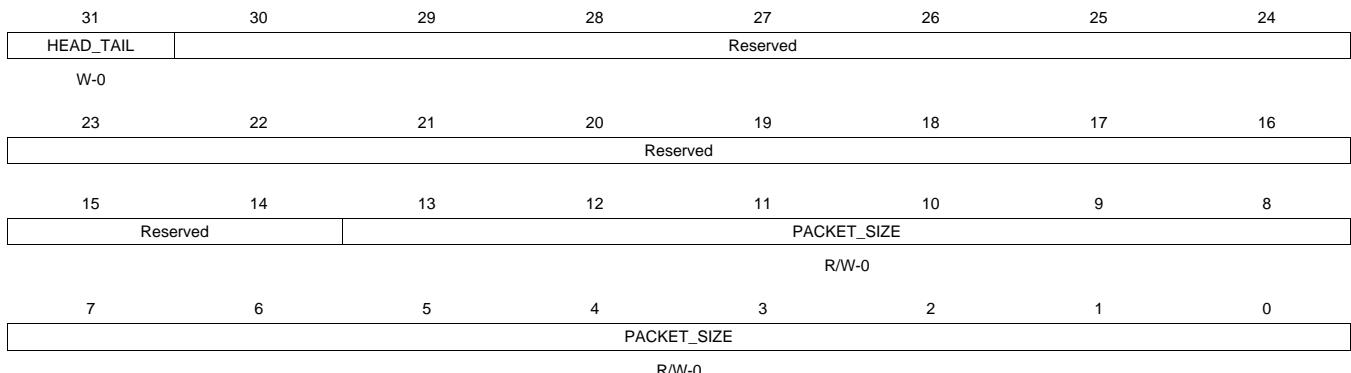
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-510. QUEUE\_46\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.221 QUEUE\_46\_C Register (offset = 22E8h) [reset = 0h]**

QUEUE\_46\_C is shown in [Figure 16-497](#) and described in [Table 16-511](#).

**Figure 16-497. QUEUE\_46\_C Register**


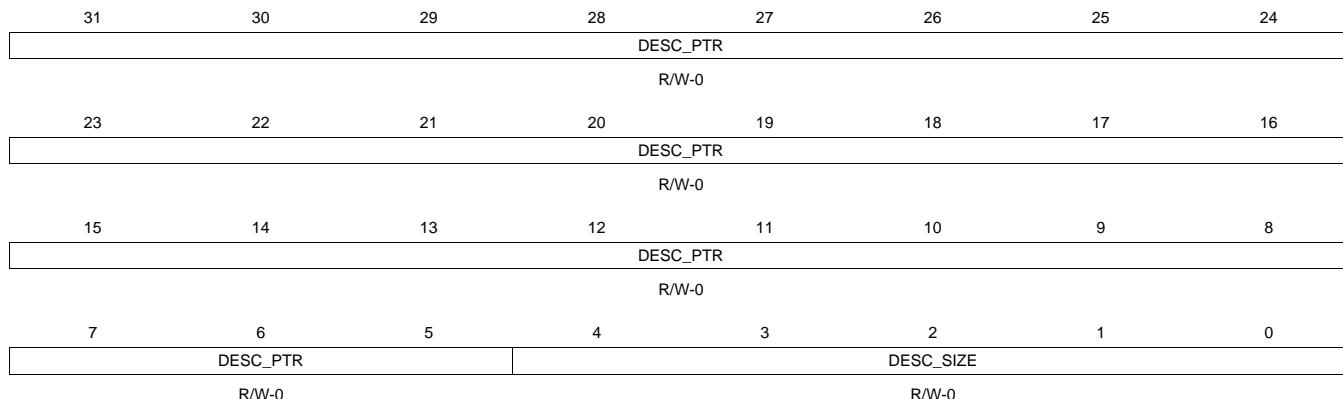
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-511. QUEUE\_46\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.222 QUEUE\_46\_D Register (offset = 22ECh) [reset = 0h]**

 QUEUE\_46\_D is shown in [Figure 16-498](#) and described in [Table 16-512](#).

**Figure 16-498. QUEUE\_46\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-512. QUEUE\_46\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.223 QUEUE\_47\_A Register (offset = 22F0h) [reset = 0h]**

QUEUE\_47\_A is shown in [Figure 16-499](#) and described in [Table 16-513](#).

**Figure 16-499. QUEUE\_47\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-513. QUEUE\_47\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.224 QUEUE\_47\_B Register (offset = 22F4h) [reset = 0h]**

QUEUE\_47\_B is shown in [Figure 16-500](#) and described in [Table 16-514](#).

**Figure 16-500. QUEUE\_47\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

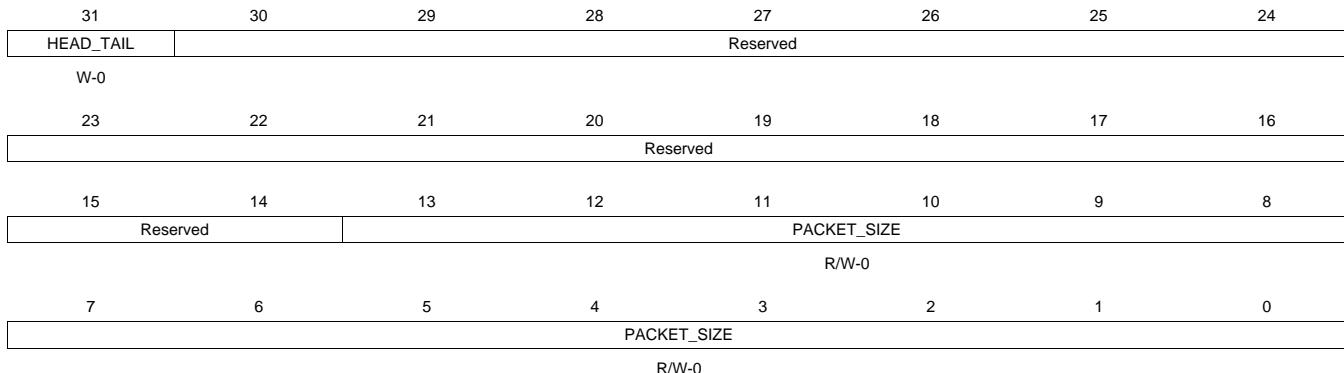
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-514. QUEUE\_47\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.225 QUEUE\_47\_C Register (offset = 22F8h) [reset = 0h]**

QUEUE\_47\_C is shown in [Figure 16-501](#) and described in [Table 16-515](#).

**Figure 16-501. QUEUE\_47\_C Register**


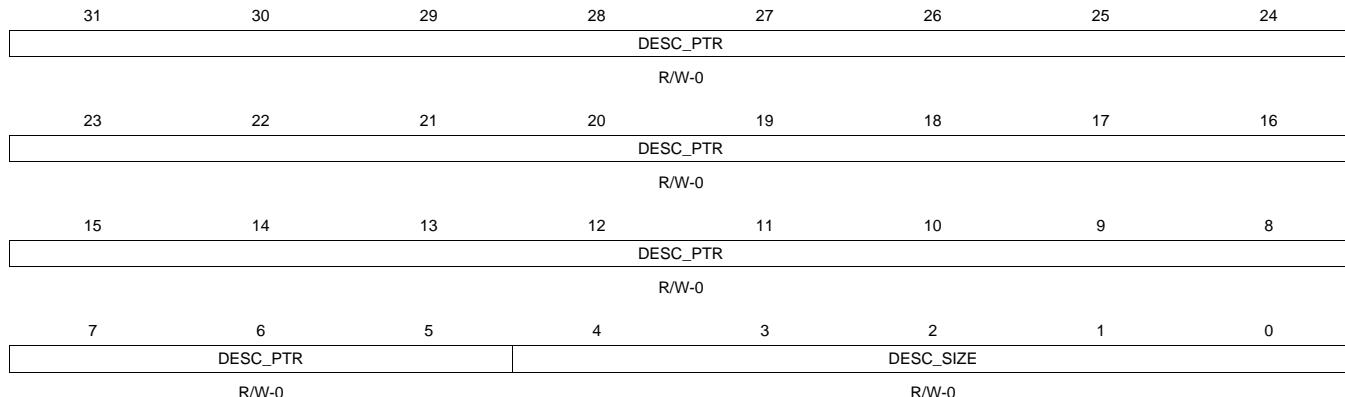
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-515. QUEUE\_47\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.226 QUEUE\_47\_D Register (offset = 22FCh) [reset = 0h]**

 QUEUE\_47\_D is shown in [Figure 16-502](#) and described in [Table 16-516](#).

**Figure 16-502. QUEUE\_47\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-516. QUEUE\_47\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.227 QUEUE\_48\_A Register (offset = 2300h) [reset = 0h]**

QUEUE\_48\_A is shown in [Figure 16-503](#) and described in [Table 16-517](#).

**Figure 16-503. QUEUE\_48\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-517. QUEUE\_48\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.228 QUEUE\_48\_B Register (offset = 2304h) [reset = 0h]**

QUEUE\_48\_B is shown in [Figure 16-504](#) and described in [Table 16-518](#).

**Figure 16-504. QUEUE\_48\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

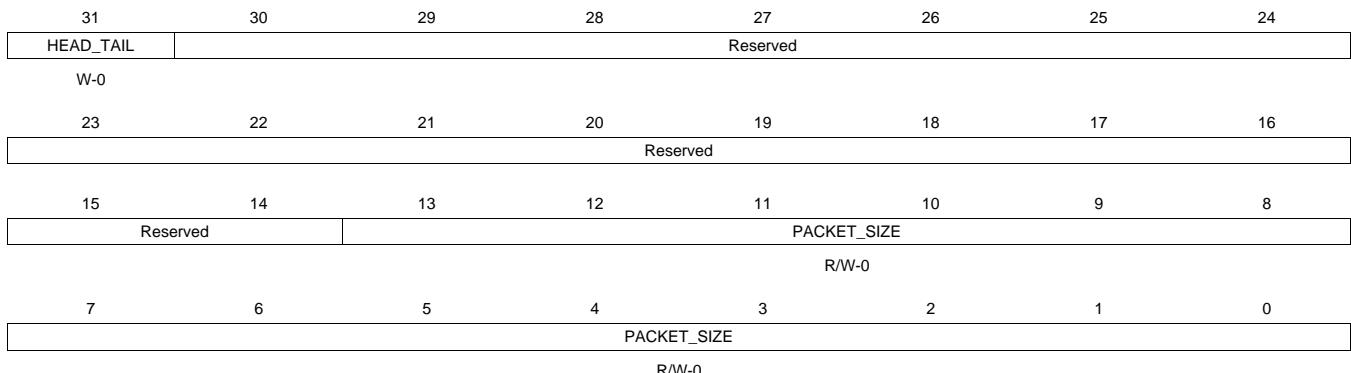
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-518. QUEUE\_48\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.229 QUEUE\_48\_C Register (offset = 2308h) [reset = 0h]**

QUEUE\_48\_C is shown in [Figure 16-505](#) and described in [Table 16-519](#).

**Figure 16-505. QUEUE\_48\_C Register**


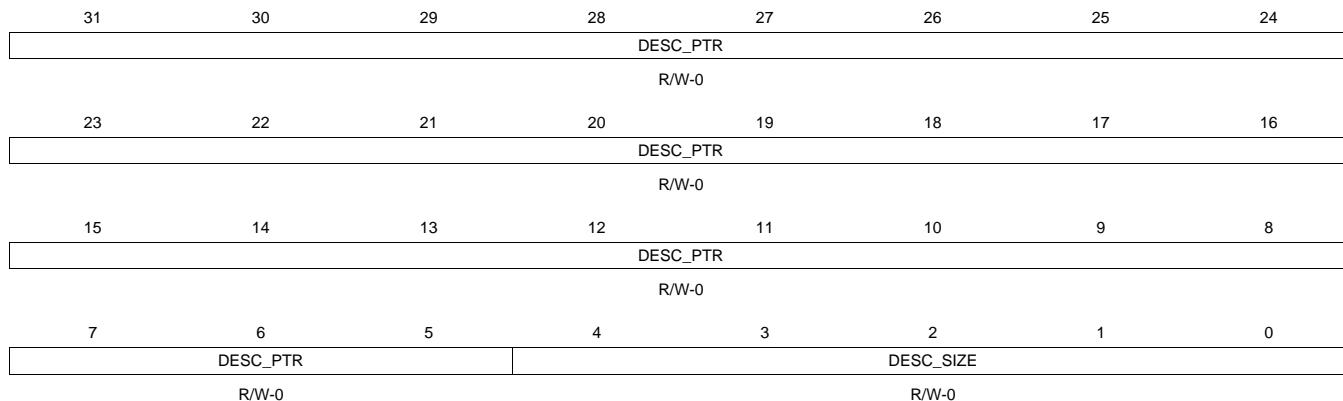
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-519. QUEUE\_48\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.230 QUEUE\_48\_D Register (offset = 230Ch) [reset = 0h]**

 QUEUE\_48\_D is shown in [Figure 16-506](#) and described in [Table 16-520](#).

**Figure 16-506. QUEUE\_48\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-520. QUEUE\_48\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.231 QUEUE\_49\_A Register (offset = 2310h) [reset = 0h]**

QUEUE\_49\_A is shown in [Figure 16-507](#) and described in [Table 16-521](#).

**Figure 16-507. QUEUE\_49\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-521. QUEUE\_49\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.232 QUEUE\_49\_B Register (offset = 2314h) [reset = 0h]**

QUEUE\_49\_B is shown in [Figure 16-508](#) and described in [Table 16-522](#).

**Figure 16-508. QUEUE\_49\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

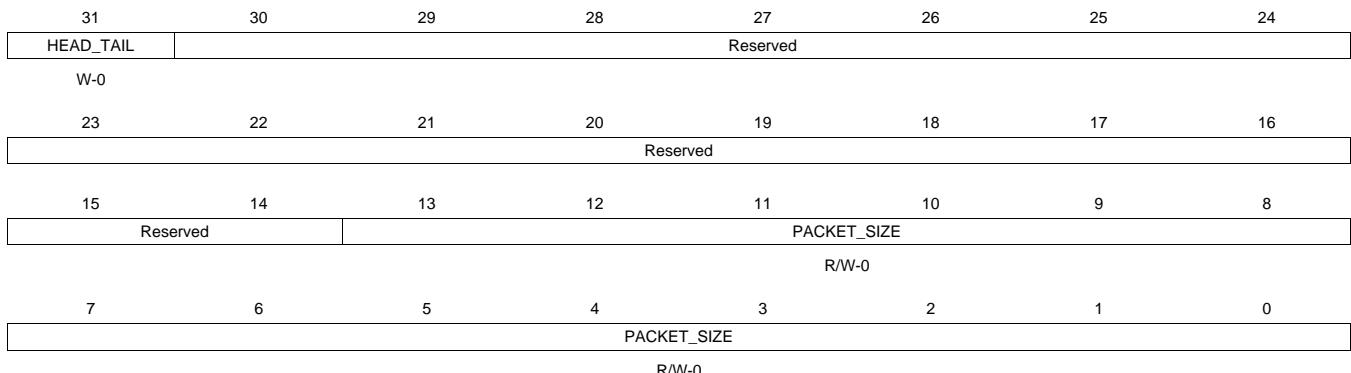
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-522. QUEUE\_49\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.233 QUEUE\_49\_C Register (offset = 2318h) [reset = 0h]**

QUEUE\_49\_C is shown in [Figure 16-509](#) and described in [Table 16-523](#).

**Figure 16-509. QUEUE\_49\_C Register**


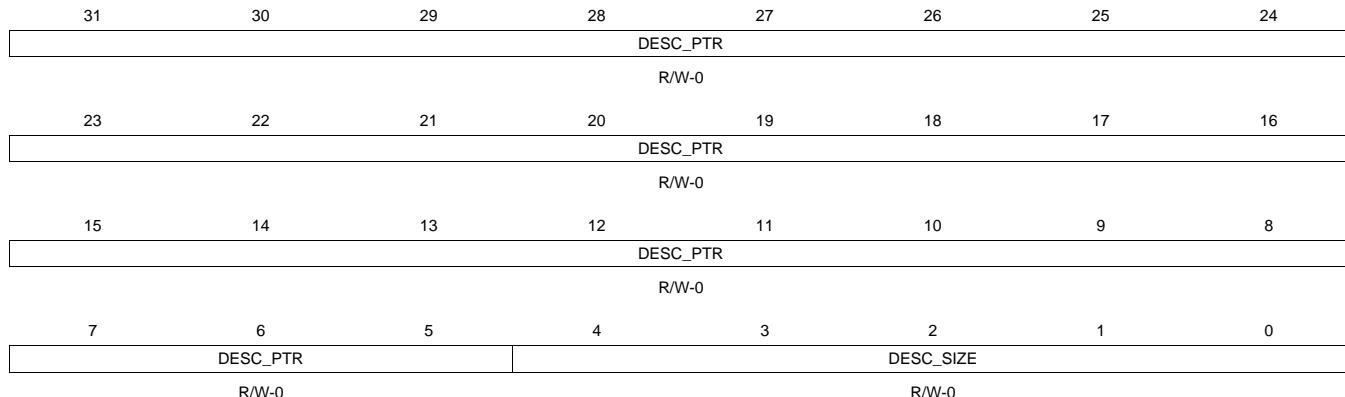
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-523. QUEUE\_49\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.234 QUEUE\_49\_D Register (offset = 231Ch) [reset = 0h]**

 QUEUE\_49\_D is shown in [Figure 16-510](#) and described in [Table 16-524](#).

**Figure 16-510. QUEUE\_49\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-524. QUEUE\_49\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.235 QUEUE\_50\_A Register (offset = 2320h) [reset = 0h]**

QUEUE\_50\_A is shown in [Figure 16-511](#) and described in [Table 16-525](#).

**Figure 16-511. QUEUE\_50\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-525. QUEUE\_50\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.236 QUEUE\_50\_B Register (offset = 2324h) [reset = 0h]**

QUEUE\_50\_B is shown in [Figure 16-512](#) and described in [Table 16-526](#).

**Figure 16-512. QUEUE\_50\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

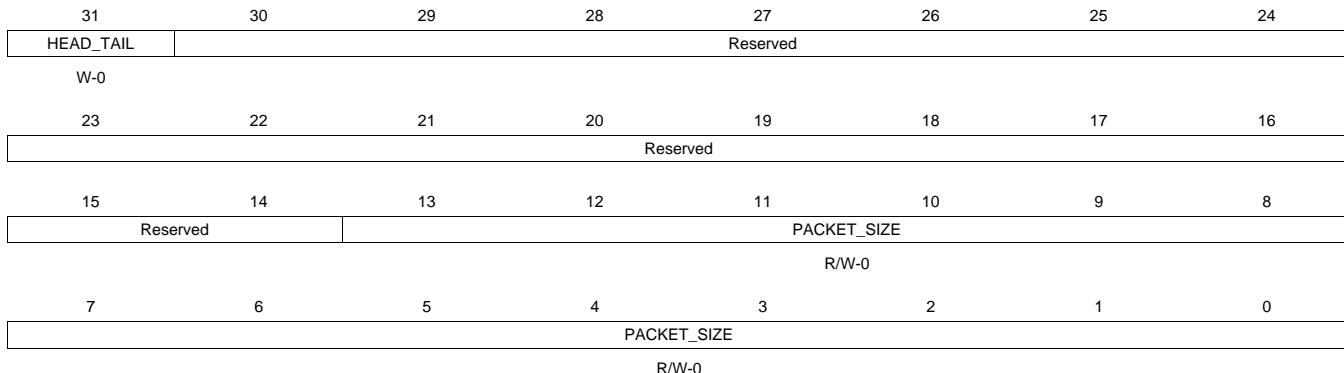
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-526. QUEUE\_50\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.237 QUEUE\_50\_C Register (offset = 2328h) [reset = 0h]**

QUEUE\_50\_C is shown in [Figure 16-513](#) and described in [Table 16-527](#).

**Figure 16-513. QUEUE\_50\_C Register**


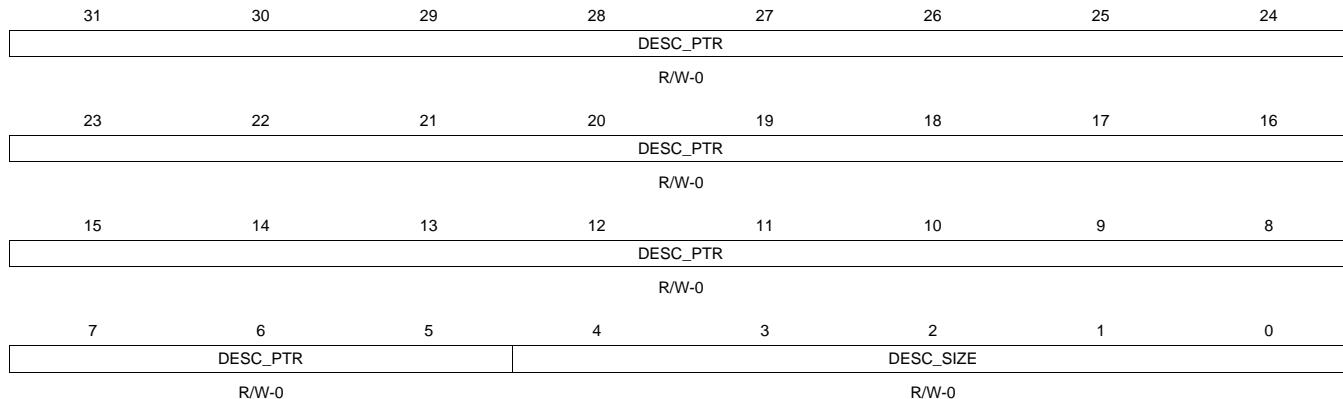
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-527. QUEUE\_50\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.238 QUEUE\_50\_D Register (offset = 232Ch) [reset = 0h]**

 QUEUE\_50\_D is shown in [Figure 16-514](#) and described in [Table 16-528](#).

**Figure 16-514. QUEUE\_50\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-528. QUEUE\_50\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.239 QUEUE\_51\_A Register (offset = 2330h) [reset = 0h]**

QUEUE\_51\_A is shown in [Figure 16-515](#) and described in [Table 16-529](#).

**Figure 16-515. QUEUE\_51\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-529. QUEUE\_51\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.240 QUEUE\_51\_B Register (offset = 2334h) [reset = 0h]**

QUEUE\_51\_B is shown in [Figure 16-516](#) and described in [Table 16-530](#).

**Figure 16-516. QUEUE\_51\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

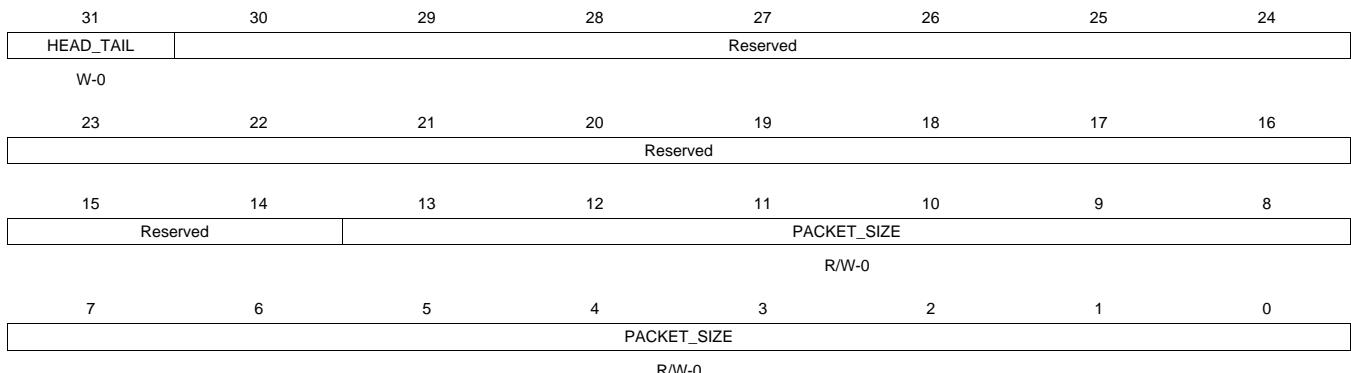
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-530. QUEUE\_51\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.241 QUEUE\_51\_C Register (offset = 2338h) [reset = 0h]**

QUEUE\_51\_C is shown in [Figure 16-517](#) and described in [Table 16-531](#).

**Figure 16-517. QUEUE\_51\_C Register**


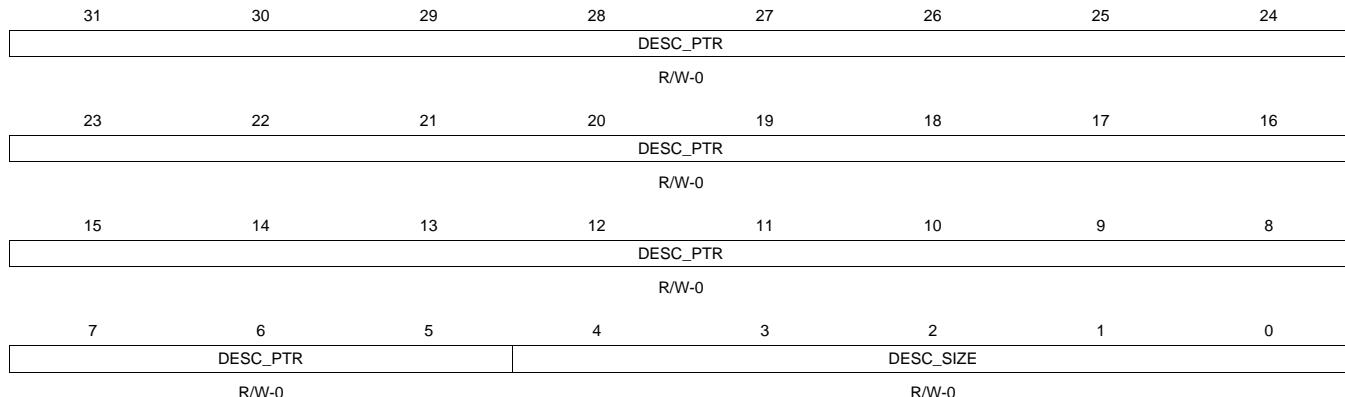
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-531. QUEUE\_51\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.242 QUEUE\_51\_D Register (offset = 233Ch) [reset = 0h]**

 QUEUE\_51\_D is shown in [Figure 16-518](#) and described in [Table 16-532](#).

**Figure 16-518. QUEUE\_51\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-532. QUEUE\_51\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.243 QUEUE\_52\_A Register (offset = 2340h) [reset = 0h]**

QUEUE\_52\_A is shown in [Figure 16-519](#) and described in [Table 16-533](#).

**Figure 16-519. QUEUE\_52\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-533. QUEUE\_52\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.244 QUEUE\_52\_B Register (offset = 2344h) [reset = 0h]**

QUEUE\_52\_B is shown in [Figure 16-520](#) and described in [Table 16-534](#).

**Figure 16-520. QUEUE\_52\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

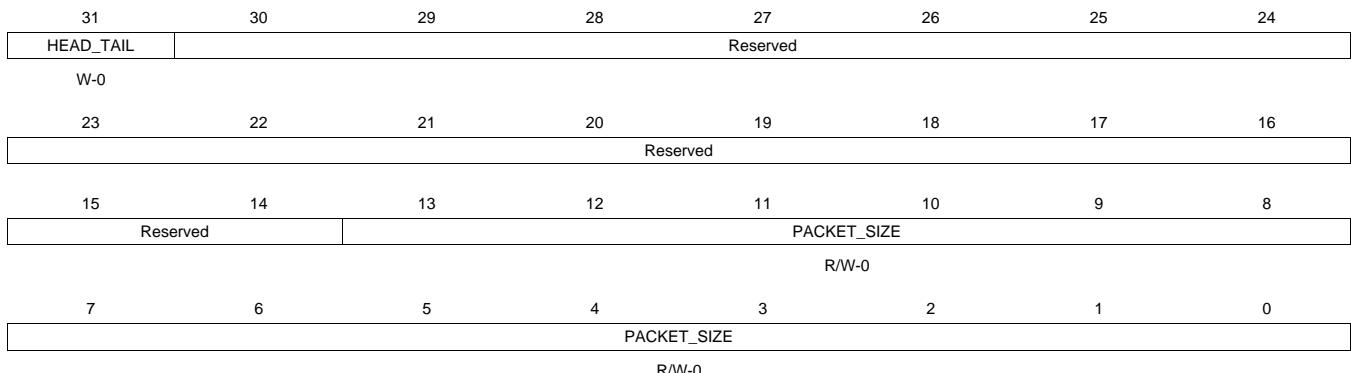
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-534. QUEUE\_52\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.245 QUEUE\_52\_C Register (offset = 2348h) [reset = 0h]**

QUEUE\_52\_C is shown in [Figure 16-521](#) and described in [Table 16-535](#).

**Figure 16-521. QUEUE\_52\_C Register**


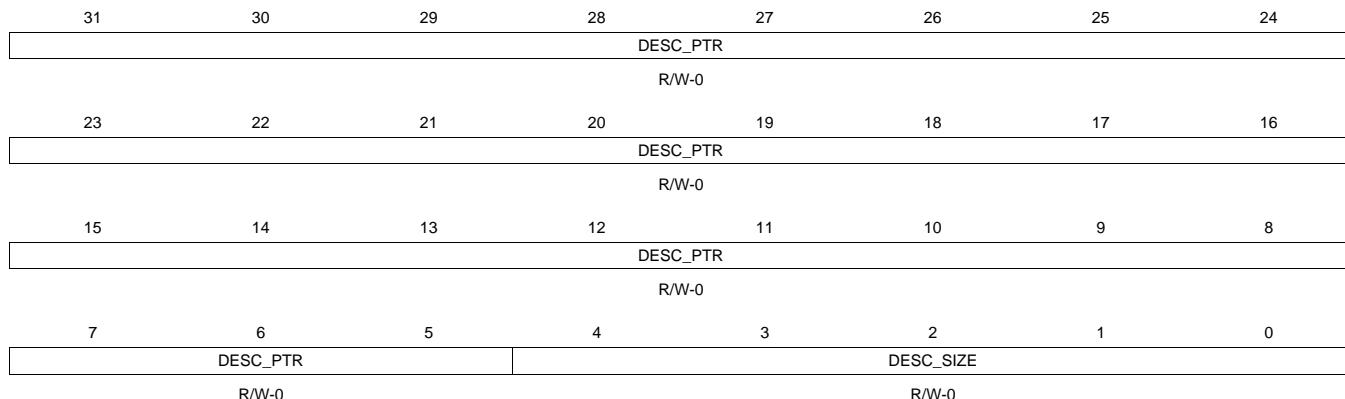
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-535. QUEUE\_52\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.246 QUEUE\_52\_D Register (offset = 234Ch) [reset = 0h]**

 QUEUE\_52\_D is shown in [Figure 16-522](#) and described in [Table 16-536](#).

**Figure 16-522. QUEUE\_52\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-536. QUEUE\_52\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.247 QUEUE\_53\_A Register (offset = 2350h) [reset = 0h]**

QUEUE\_53\_A is shown in [Figure 16-523](#) and described in [Table 16-537](#).

**Figure 16-523. QUEUE\_53\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-537. QUEUE\_53\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.248 QUEUE\_53\_B Register (offset = 2354h) [reset = 0h]**

QUEUE\_53\_B is shown in [Figure 16-524](#) and described in [Table 16-538](#).

**Figure 16-524. QUEUE\_53\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

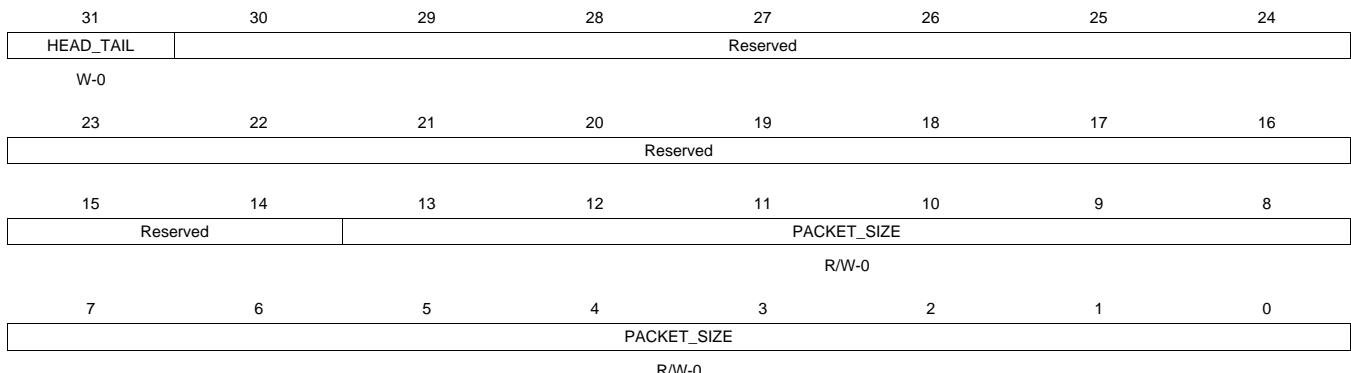
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-538. QUEUE\_53\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.249 QUEUE\_53\_C Register (offset = 2358h) [reset = 0h]**

QUEUE\_53\_C is shown in [Figure 16-525](#) and described in [Table 16-539](#).

**Figure 16-525. QUEUE\_53\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

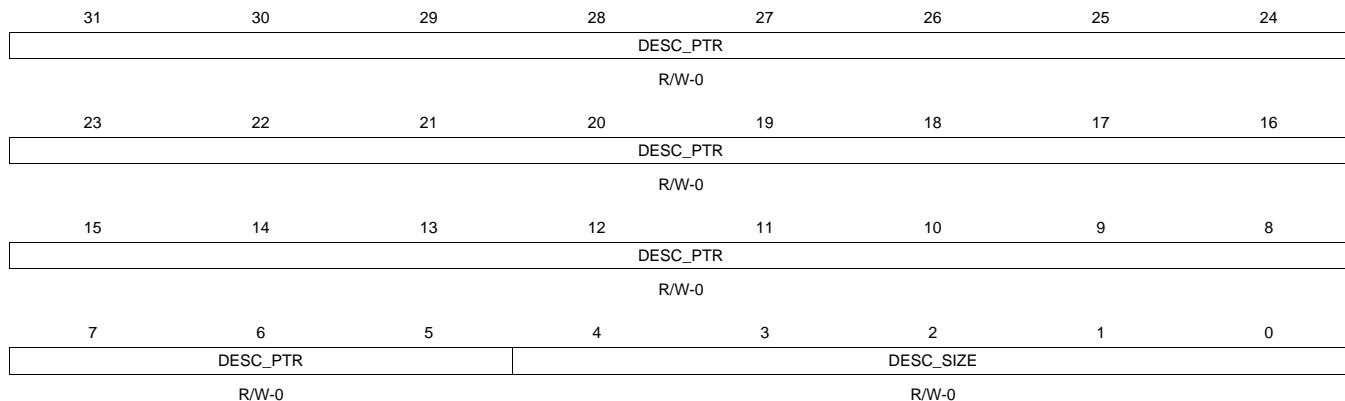
**Table 16-539. QUEUE\_53\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.250 QUEUE\_53\_D Register (offset = 235Ch) [reset = 0h]

QUEUE\_53\_D is shown in [Figure 16-526](#) and described in [Table 16-540](#).

**Figure 16-526. QUEUE\_53\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-540. QUEUE\_53\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.251 QUEUE\_54\_A Register (offset = 2360h) [reset = 0h]**

QUEUE\_54\_A is shown in [Figure 16-527](#) and described in [Table 16-541](#).

**Figure 16-527. QUEUE\_54\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-541. QUEUE\_54\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.252 QUEUE\_54\_B Register (offset = 2364h) [reset = 0h]**

QUEUE\_54\_B is shown in [Figure 16-528](#) and described in [Table 16-542](#).

**Figure 16-528. QUEUE\_54\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

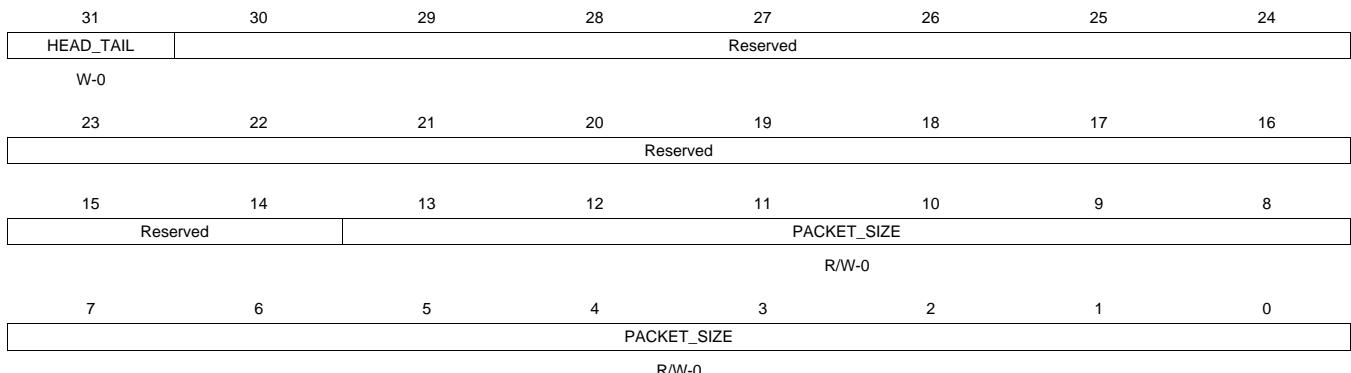
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-542. QUEUE\_54\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.253 QUEUE\_54\_C Register (offset = 2368h) [reset = 0h]**

QUEUE\_54\_C is shown in [Figure 16-529](#) and described in [Table 16-543](#).

**Figure 16-529. QUEUE\_54\_C Register**


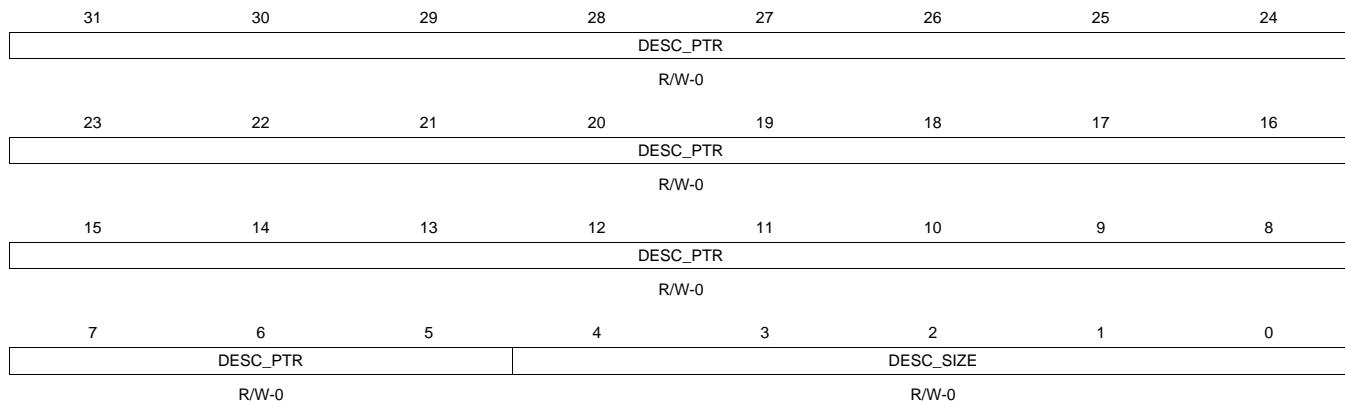
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-543. QUEUE\_54\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.254 QUEUE\_54\_D Register (offset = 236Ch) [reset = 0h]**

 QUEUE\_54\_D is shown in [Figure 16-530](#) and described in [Table 16-544](#).

**Figure 16-530. QUEUE\_54\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-544. QUEUE\_54\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.255 QUEUE\_55\_A Register (offset = 2370h) [reset = 0h]**

QUEUE\_55\_A is shown in [Figure 16-531](#) and described in [Table 16-545](#).

**Figure 16-531. QUEUE\_55\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-545. QUEUE\_55\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.256 QUEUE\_55\_B Register (offset = 2374h) [reset = 0h]**

QUEUE\_55\_B is shown in [Figure 16-532](#) and described in [Table 16-546](#).

**Figure 16-532. QUEUE\_55\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

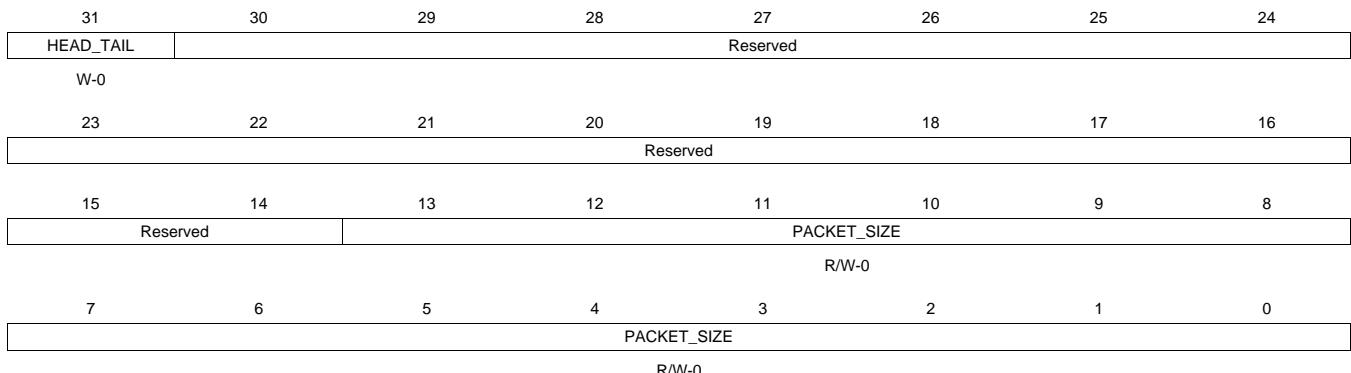
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-546. QUEUE\_55\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.257 QUEUE\_55\_C Register (offset = 2378h) [reset = 0h]**

QUEUE\_55\_C is shown in [Figure 16-533](#) and described in [Table 16-547](#).

**Figure 16-533. QUEUE\_55\_C Register**


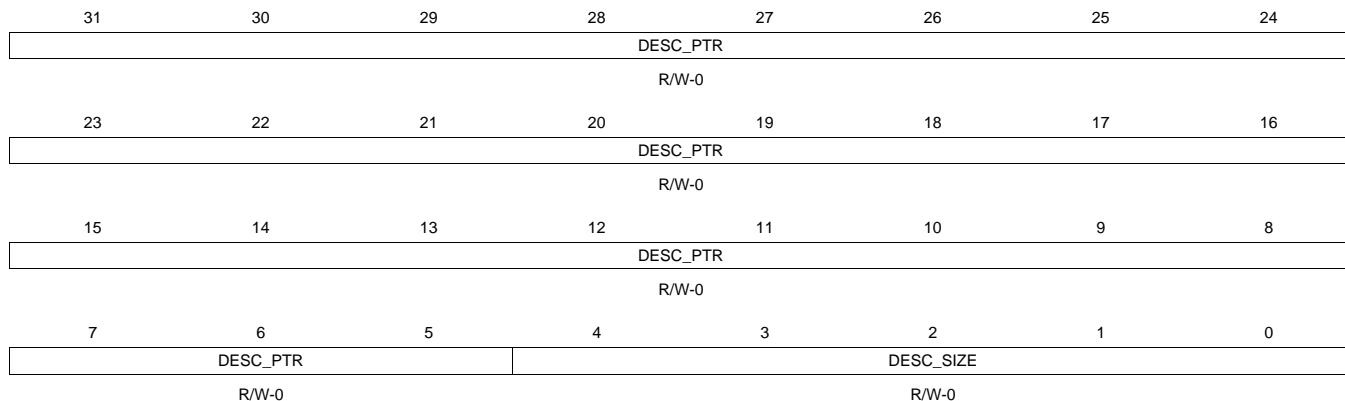
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-547. QUEUE\_55\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.258 QUEUE\_55\_D Register (offset = 237Ch) [reset = 0h]**

 QUEUE\_55\_D is shown in [Figure 16-534](#) and described in [Table 16-548](#).

**Figure 16-534. QUEUE\_55\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-548. QUEUE\_55\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.259 QUEUE\_56\_A Register (offset = 2380h) [reset = 0h]**

QUEUE\_56\_A is shown in [Figure 16-535](#) and described in [Table 16-549](#).

**Figure 16-535. QUEUE\_56\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-549. QUEUE\_56\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.260 QUEUE\_56\_B Register (offset = 2384h) [reset = 0h]**

QUEUE\_56\_B is shown in [Figure 16-536](#) and described in [Table 16-550](#).

**Figure 16-536. QUEUE\_56\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

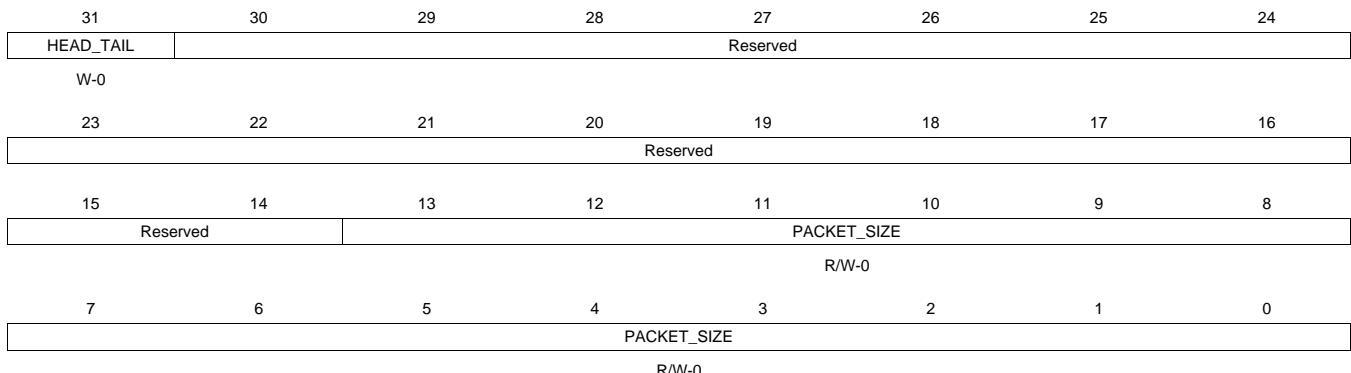
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-550. QUEUE\_56\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.261 QUEUE\_56\_C Register (offset = 2388h) [reset = 0h]**

QUEUE\_56\_C is shown in [Figure 16-537](#) and described in [Table 16-551](#).

**Figure 16-537. QUEUE\_56\_C Register**


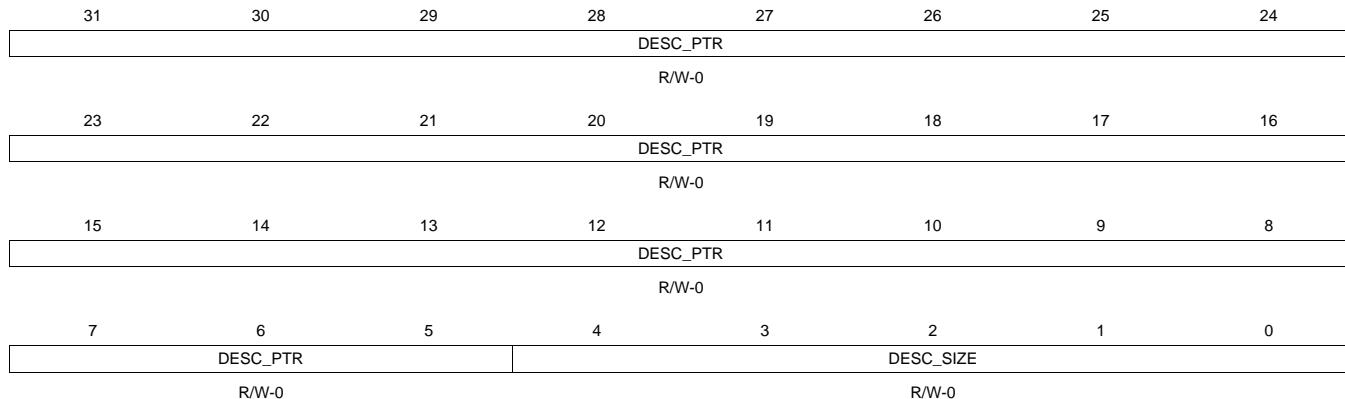
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-551. QUEUE\_56\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.262 QUEUE\_56\_D Register (offset = 238Ch) [reset = 0h]**

 QUEUE\_56\_D is shown in [Figure 16-538](#) and described in [Table 16-552](#).

**Figure 16-538. QUEUE\_56\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-552. QUEUE\_56\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.263 QUEUE\_57\_A Register (offset = 2390h) [reset = 0h]**

QUEUE\_57\_A is shown in [Figure 16-539](#) and described in [Table 16-553](#).

**Figure 16-539. QUEUE\_57\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-553. QUEUE\_57\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.264 QUEUE\_57\_B Register (offset = 2394h) [reset = 0h]**

QUEUE\_57\_B is shown in [Figure 16-540](#) and described in [Table 16-554](#).

**Figure 16-540. QUEUE\_57\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

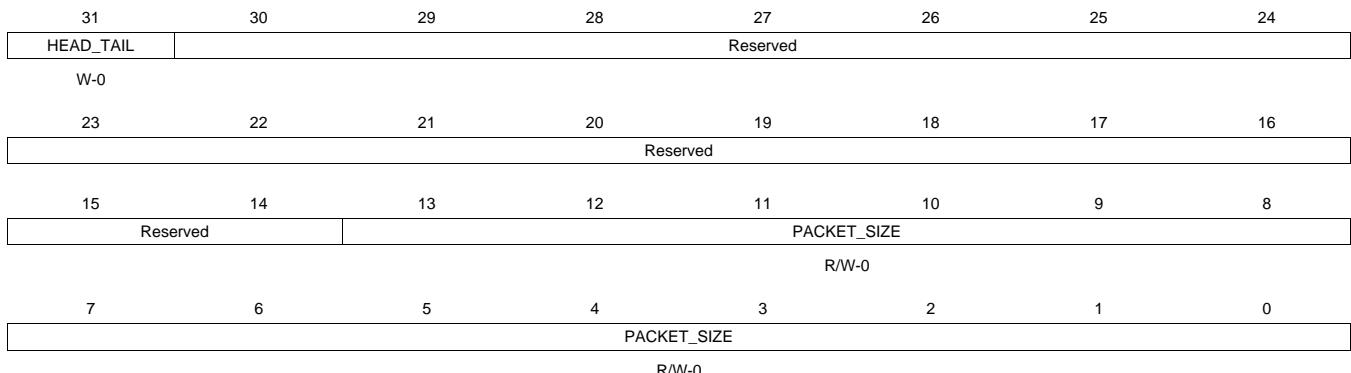
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-554. QUEUE\_57\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.265 QUEUE\_57\_C Register (offset = 2398h) [reset = 0h]**

QUEUE\_57\_C is shown in [Figure 16-541](#) and described in [Table 16-555](#).

**Figure 16-541. QUEUE\_57\_C Register**


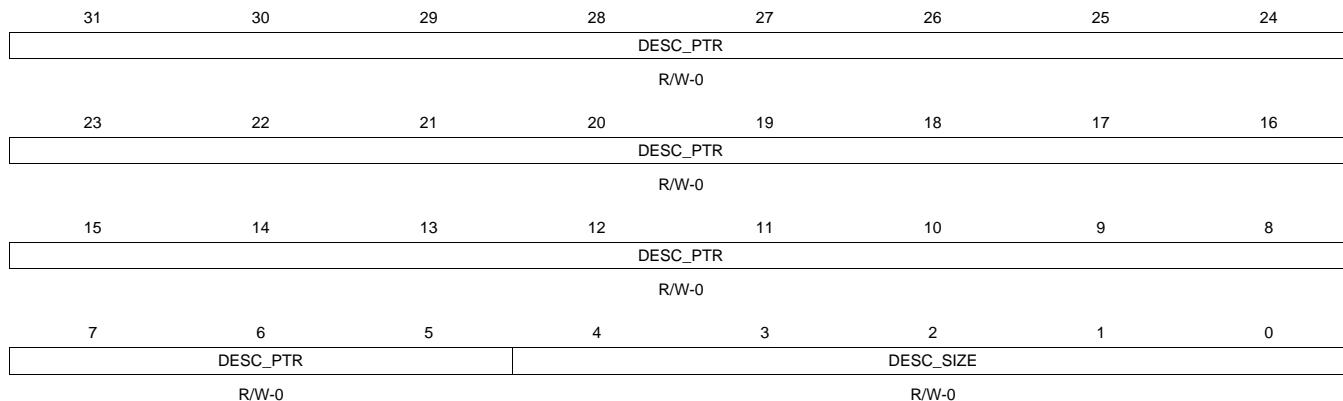
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-555. QUEUE\_57\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.266 QUEUE\_57\_D Register (offset = 239Ch) [reset = 0h]**

 QUEUE\_57\_D is shown in [Figure 16-542](#) and described in [Table 16-556](#).

**Figure 16-542. QUEUE\_57\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-556. QUEUE\_57\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.267 QUEUE\_58\_A Register (offset = 23A0h) [reset = 0h]**

QUEUE\_58\_A is shown in [Figure 16-543](#) and described in [Table 16-557](#).

**Figure 16-543. QUEUE\_58\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-557. QUEUE\_58\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.268 QUEUE\_58\_B Register (offset = 23A4h) [reset = 0h]**

QUEUE\_58\_B is shown in [Figure 16-544](#) and described in [Table 16-558](#).

**Figure 16-544. QUEUE\_58\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

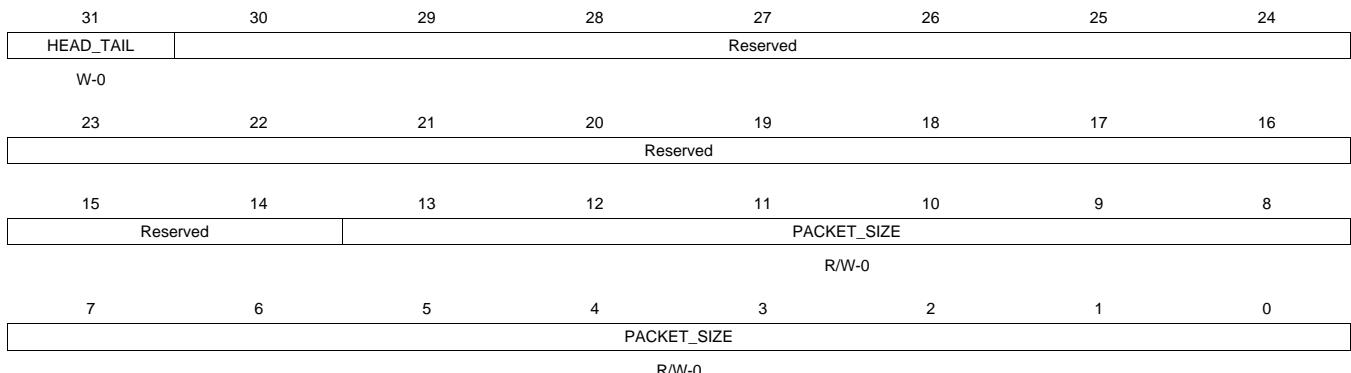
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-558. QUEUE\_58\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.269 QUEUE\_58\_C Register (offset = 23A8h) [reset = 0h]**

QUEUE\_58\_C is shown in [Figure 16-545](#) and described in [Table 16-559](#).

**Figure 16-545. QUEUE\_58\_C Register**


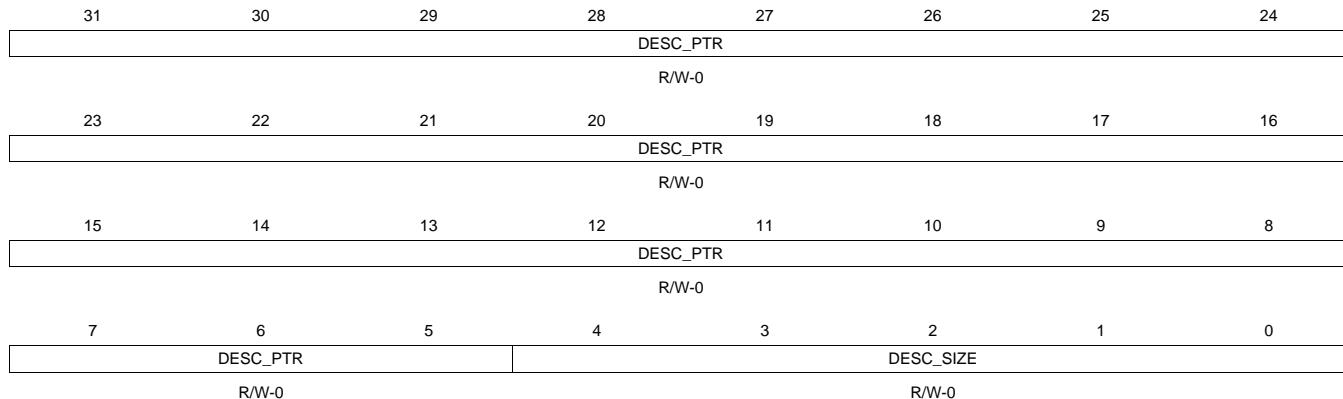
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-559. QUEUE\_58\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.270 QUEUE\_58\_D Register (offset = 23ACh) [reset = 0h]**

 QUEUE\_58\_D is shown in [Figure 16-546](#) and described in [Table 16-560](#).

**Figure 16-546. QUEUE\_58\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-560. QUEUE\_58\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.271 QUEUE\_59\_A Register (offset = 23B0h) [reset = 0h]**

QUEUE\_59\_A is shown in [Figure 16-547](#) and described in [Table 16-561](#).

**Figure 16-547. QUEUE\_59\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-561. QUEUE\_59\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.272 QUEUE\_59\_B Register (offset = 23B4h) [reset = 0h]**

QUEUE\_59\_B is shown in [Figure 16-548](#) and described in [Table 16-562](#).

**Figure 16-548. QUEUE\_59\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

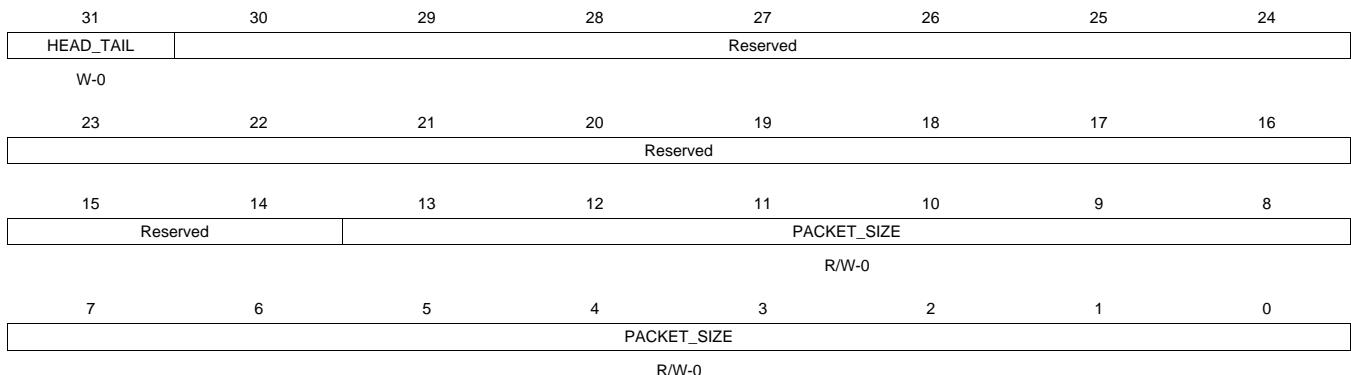
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-562. QUEUE\_59\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.273 QUEUE\_59\_C Register (offset = 23B8h) [reset = 0h]**

QUEUE\_59\_C is shown in [Figure 16-549](#) and described in [Table 16-563](#).

**Figure 16-549. QUEUE\_59\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

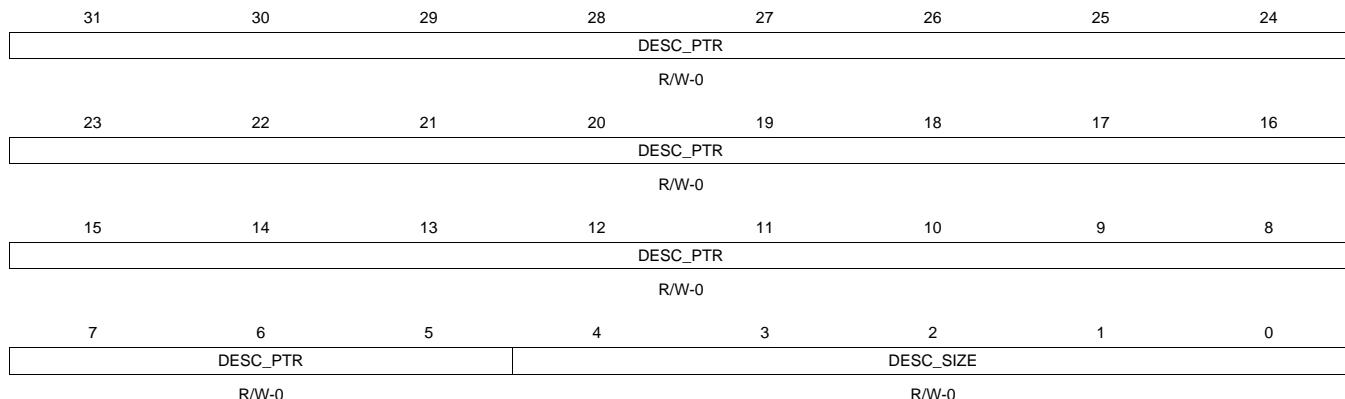
**Table 16-563. QUEUE\_59\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.274 QUEUE\_59\_D Register (offset = 23BCh) [reset = 0h]

QUEUE\_59\_D is shown in [Figure 16-550](#) and described in [Table 16-564](#).

**Figure 16-550. QUEUE\_59\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-564. QUEUE\_59\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.275 QUEUE\_60\_A Register (offset = 23C0h) [reset = 0h]**

QUEUE\_60\_A is shown in [Figure 16-551](#) and described in [Table 16-565](#).

**Figure 16-551. QUEUE\_60\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-565. QUEUE\_60\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.276 QUEUE\_60\_B Register (offset = 23C4h) [reset = 0h]**

QUEUE\_60\_B is shown in [Figure 16-552](#) and described in [Table 16-566](#).

**Figure 16-552. QUEUE\_60\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

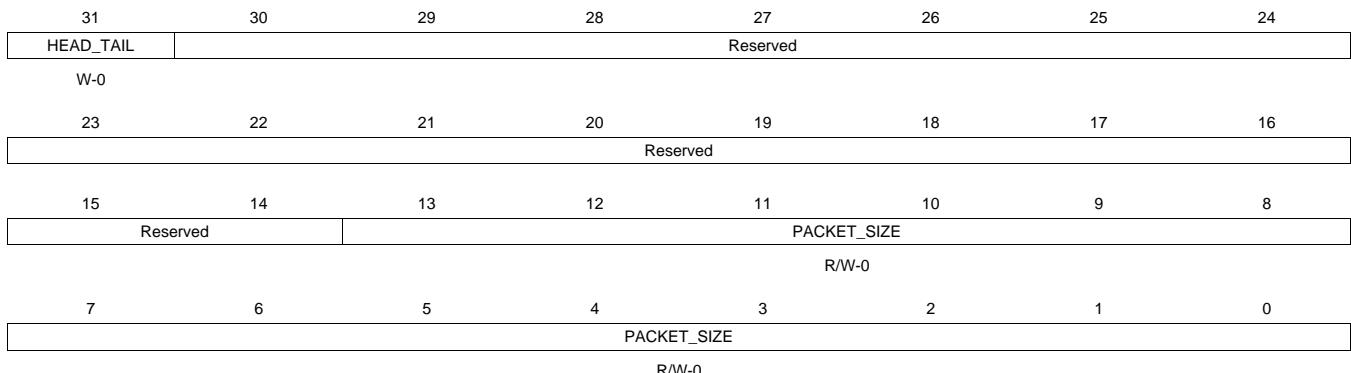
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-566. QUEUE\_60\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.277 QUEUE\_60\_C Register (offset = 23C8h) [reset = 0h]**

QUEUE\_60\_C is shown in [Figure 16-553](#) and described in [Table 16-567](#).

**Figure 16-553. QUEUE\_60\_C Register**


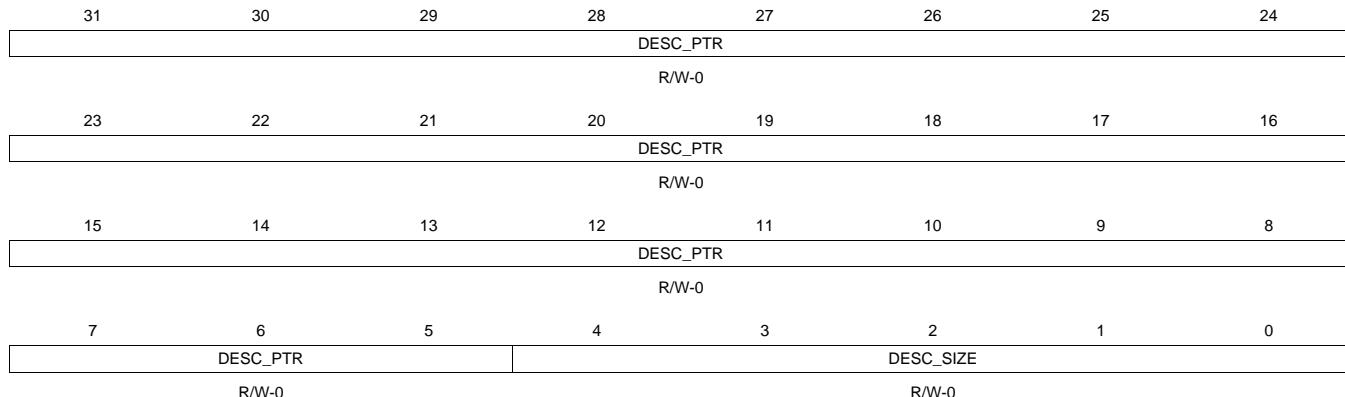
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-567. QUEUE\_60\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.278 QUEUE\_60\_D Register (offset = 23CCh) [reset = 0h]**

 QUEUE\_60\_D is shown in [Figure 16-554](#) and described in [Table 16-568](#).

**Figure 16-554. QUEUE\_60\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-568. QUEUE\_60\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.279 QUEUE\_61\_A Register (offset = 23D0h) [reset = 0h]**

QUEUE\_61\_A is shown in [Figure 16-555](#) and described in [Table 16-569](#).

**Figure 16-555. QUEUE\_61\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-569. QUEUE\_61\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.280 QUEUE\_61\_B Register (offset = 23D4h) [reset = 0h]**

QUEUE\_61\_B is shown in [Figure 16-556](#) and described in [Table 16-570](#).

**Figure 16-556. QUEUE\_61\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

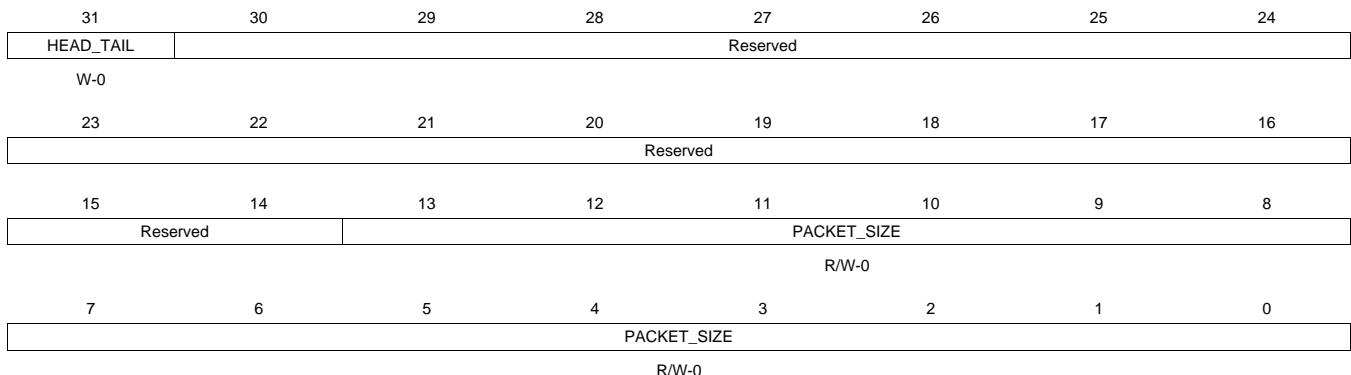
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-570. QUEUE\_61\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.281 QUEUE\_61\_C Register (offset = 23D8h) [reset = 0h]**

QUEUE\_61\_C is shown in [Figure 16-557](#) and described in [Table 16-571](#).

**Figure 16-557. QUEUE\_61\_C Register**


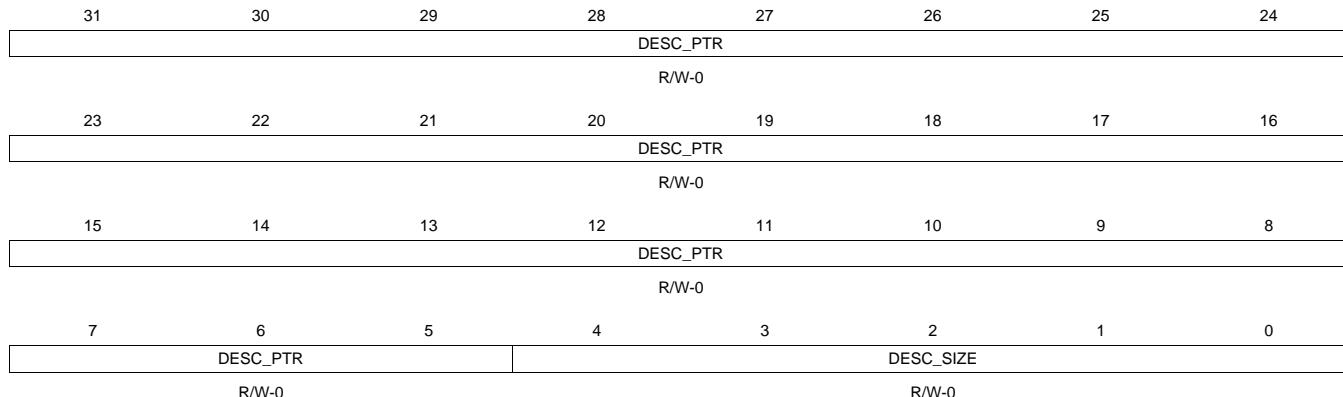
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-571. QUEUE\_61\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.282 QUEUE\_61\_D Register (offset = 23DCh) [reset = 0h]**

QUEUE\_61\_D is shown in [Figure 16-558](#) and described in [Table 16-572](#).

**Figure 16-558. QUEUE\_61\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-572. QUEUE\_61\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.283 QUEUE\_62\_A Register (offset = 23E0h) [reset = 0h]**

QUEUE\_62\_A is shown in [Figure 16-559](#) and described in [Table 16-573](#).

**Figure 16-559. QUEUE\_62\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-573. QUEUE\_62\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.284 QUEUE\_62\_B Register (offset = 23E4h) [reset = 0h]**

QUEUE\_62\_B is shown in [Figure 16-560](#) and described in [Table 16-574](#).

**Figure 16-560. QUEUE\_62\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

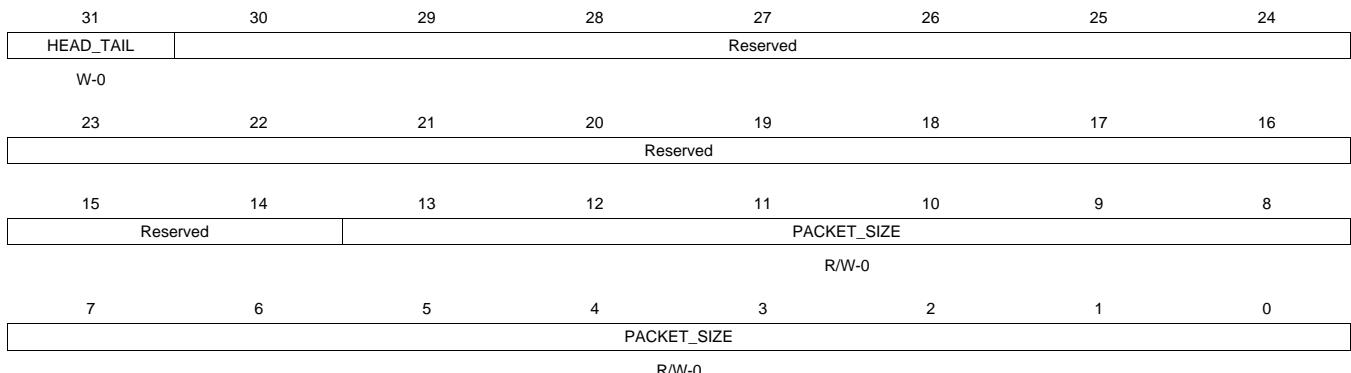
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-574. QUEUE\_62\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.285 QUEUE\_62\_C Register (offset = 23E8h) [reset = 0h]**

QUEUE\_62\_C is shown in [Figure 16-561](#) and described in [Table 16-575](#).

**Figure 16-561. QUEUE\_62\_C Register**


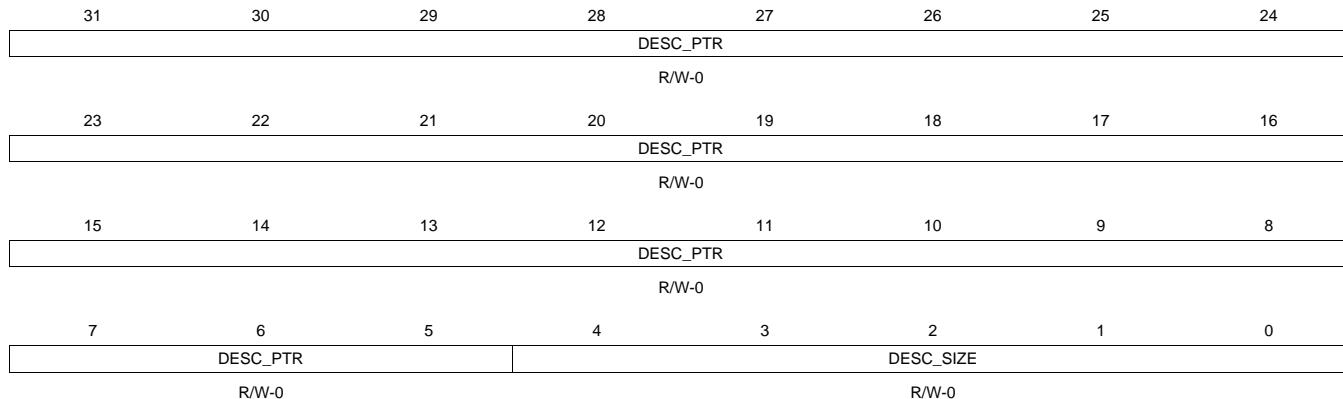
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-575. QUEUE\_62\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.286 QUEUE\_62\_D Register (offset = 23ECh) [reset = 0h]**

 QUEUE\_62\_D is shown in [Figure 16-562](#) and described in [Table 16-576](#).

**Figure 16-562. QUEUE\_62\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-576. QUEUE\_62\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.287 QUEUE\_63\_A Register (offset = 23F0h) [reset = 0h]**

QUEUE\_63\_A is shown in [Figure 16-563](#) and described in [Table 16-577](#).

**Figure 16-563. QUEUE\_63\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-577. QUEUE\_63\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.288 QUEUE\_63\_B Register (offset = 23F4h) [reset = 0h]**

QUEUE\_63\_B is shown in [Figure 16-564](#) and described in [Table 16-578](#).

**Figure 16-564. QUEUE\_63\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

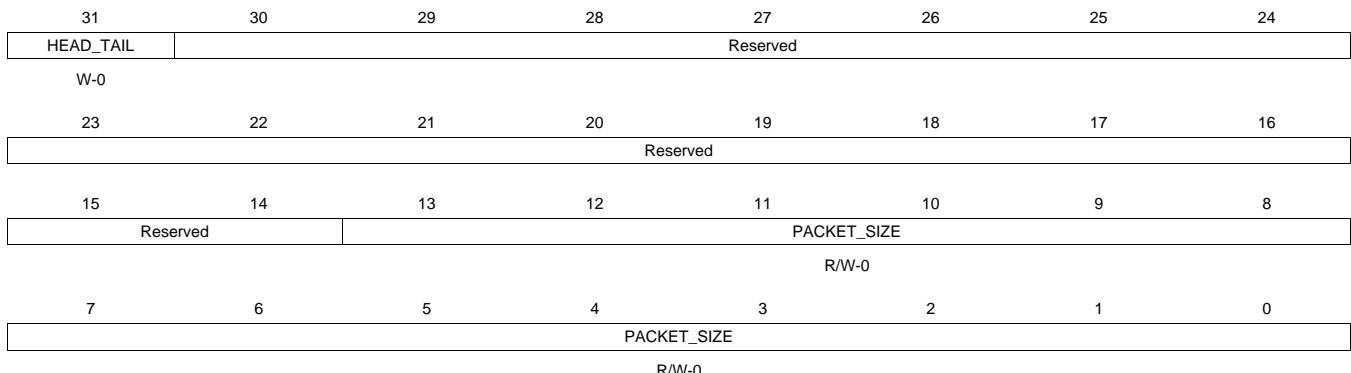
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-578. QUEUE\_63\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.289 QUEUE\_63\_C Register (offset = 23F8h) [reset = 0h]**

QUEUE\_63\_C is shown in [Figure 16-565](#) and described in [Table 16-579](#).

**Figure 16-565. QUEUE\_63\_C Register**


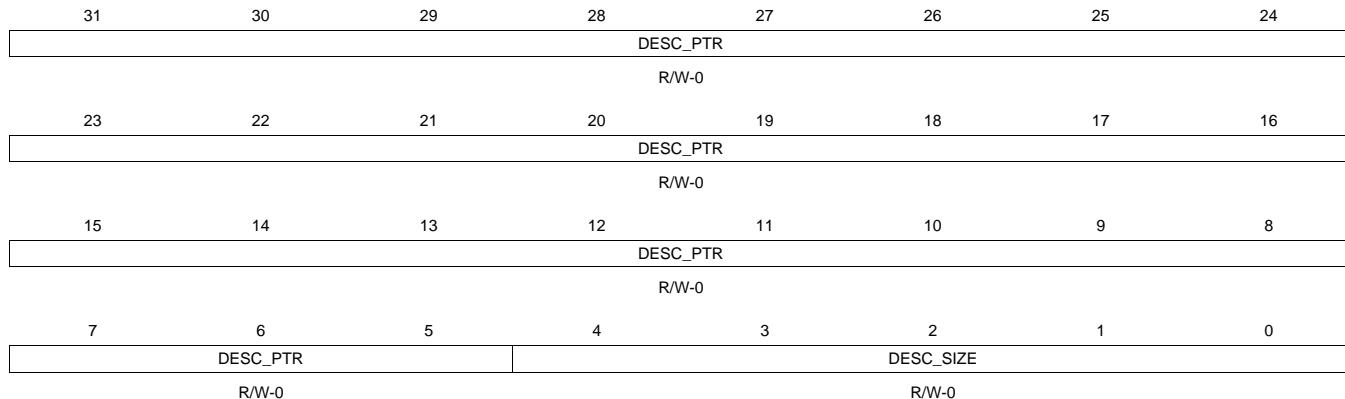
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-579. QUEUE\_63\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.290 QUEUE\_63\_D Register (offset = 23FCh) [reset = 0h]**

 QUEUE\_63\_D is shown in [Figure 16-566](#) and described in [Table 16-580](#).

**Figure 16-566. QUEUE\_63\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-580. QUEUE\_63\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.291 QUEUE\_64\_A Register (offset = 2400h) [reset = 0h]**

QUEUE\_64\_A is shown in [Figure 16-567](#) and described in [Table 16-581](#).

**Figure 16-567. QUEUE\_64\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-581. QUEUE\_64\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.292 QUEUE\_64\_B Register (offset = 2404h) [reset = 0h]**

QUEUE\_64\_B is shown in [Figure 16-568](#) and described in [Table 16-582](#).

**Figure 16-568. QUEUE\_64\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

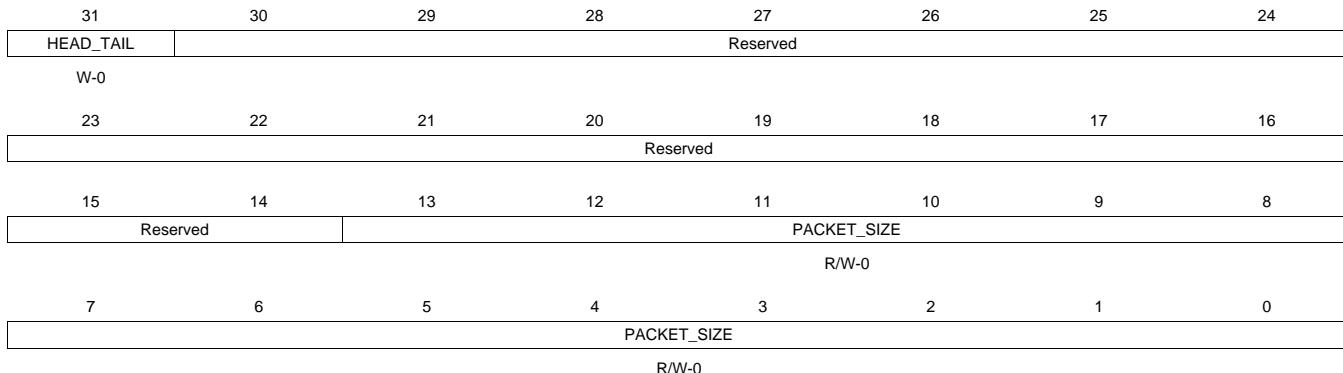
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-582. QUEUE\_64\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.293 QUEUE\_64\_C Register (offset = 2408h) [reset = 0h]**

QUEUE\_64\_C is shown in [Figure 16-569](#) and described in [Table 16-583](#).

**Figure 16-569. QUEUE\_64\_C Register**


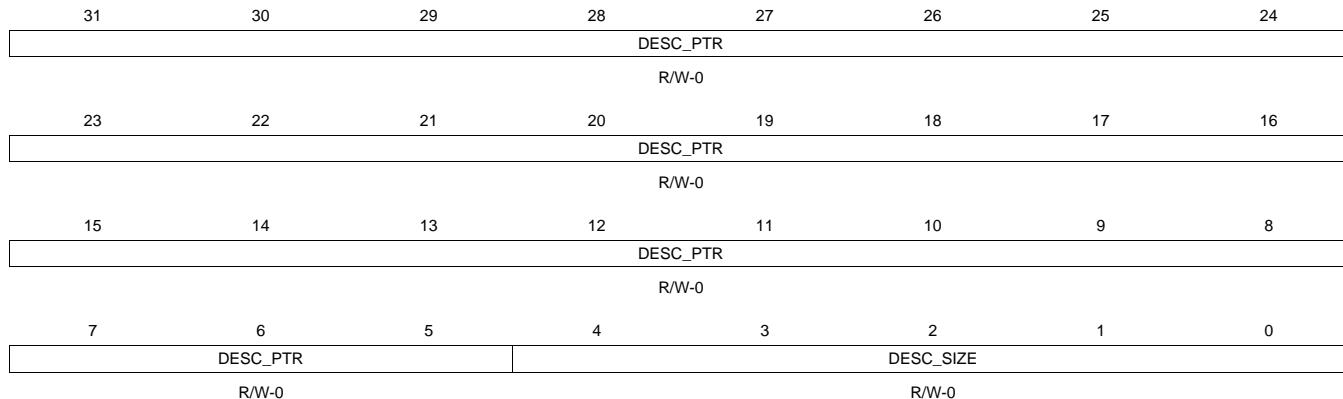
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-583. QUEUE\_64\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.294 QUEUE\_64\_D Register (offset = 240Ch) [reset = 0h]**

 QUEUE\_64\_D is shown in [Figure 16-570](#) and described in [Table 16-584](#).

**Figure 16-570. QUEUE\_64\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-584. QUEUE\_64\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.295 QUEUE\_65\_A Register (offset = 2410h) [reset = 0h]**

QUEUE\_65\_A is shown in [Figure 16-571](#) and described in [Table 16-585](#).

**Figure 16-571. QUEUE\_65\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-585. QUEUE\_65\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.296 QUEUE\_65\_B Register (offset = 2414h) [reset = 0h]**

QUEUE\_65\_B is shown in [Figure 16-572](#) and described in [Table 16-586](#).

**Figure 16-572. QUEUE\_65\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

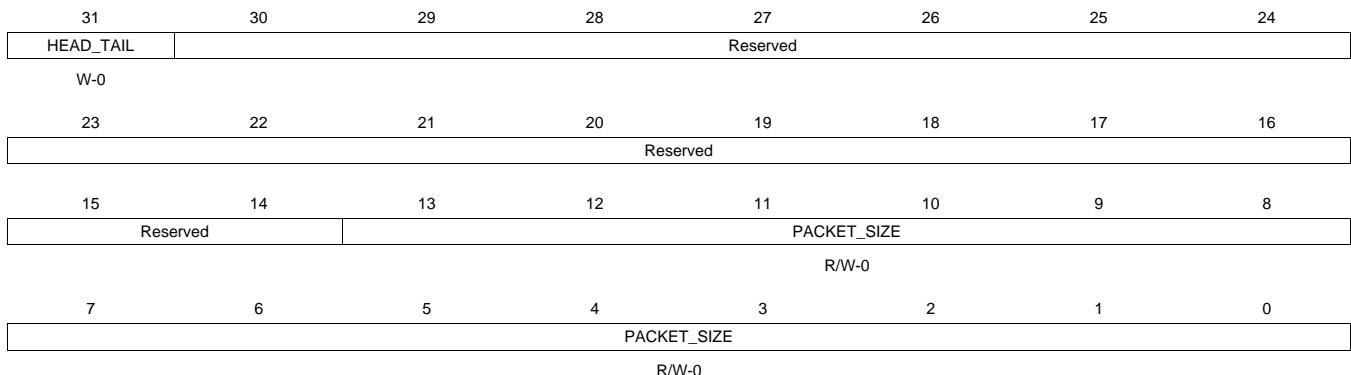
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-586. QUEUE\_65\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.297 QUEUE\_65\_C Register (offset = 2418h) [reset = 0h]**

QUEUE\_65\_C is shown in [Figure 16-573](#) and described in [Table 16-587](#).

**Figure 16-573. QUEUE\_65\_C Register**


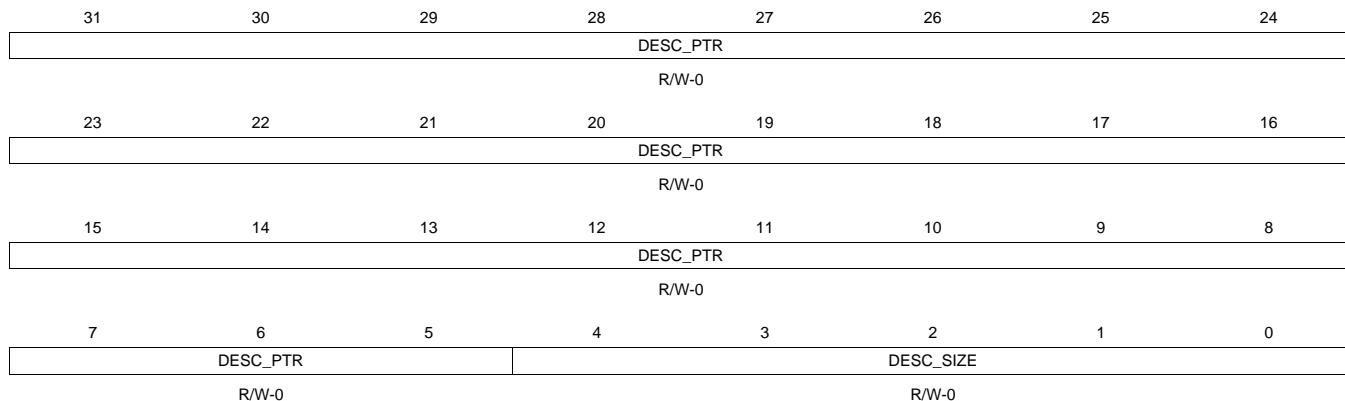
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-587. QUEUE\_65\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.298 QUEUE\_65\_D Register (offset = 241Ch) [reset = 0h]**

 QUEUE\_65\_D is shown in [Figure 16-574](#) and described in [Table 16-588](#).

**Figure 16-574. QUEUE\_65\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-588. QUEUE\_65\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.299 QUEUE\_66\_A Register (offset = 2420h) [reset = 0h]**

QUEUE\_66\_A is shown in [Figure 16-575](#) and described in [Table 16-589](#).

**Figure 16-575. QUEUE\_66\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-589. QUEUE\_66\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.300 QUEUE\_66\_B Register (offset = 2424h) [reset = 0h]**

QUEUE\_66\_B is shown in [Figure 16-576](#) and described in [Table 16-590](#).

**Figure 16-576. QUEUE\_66\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

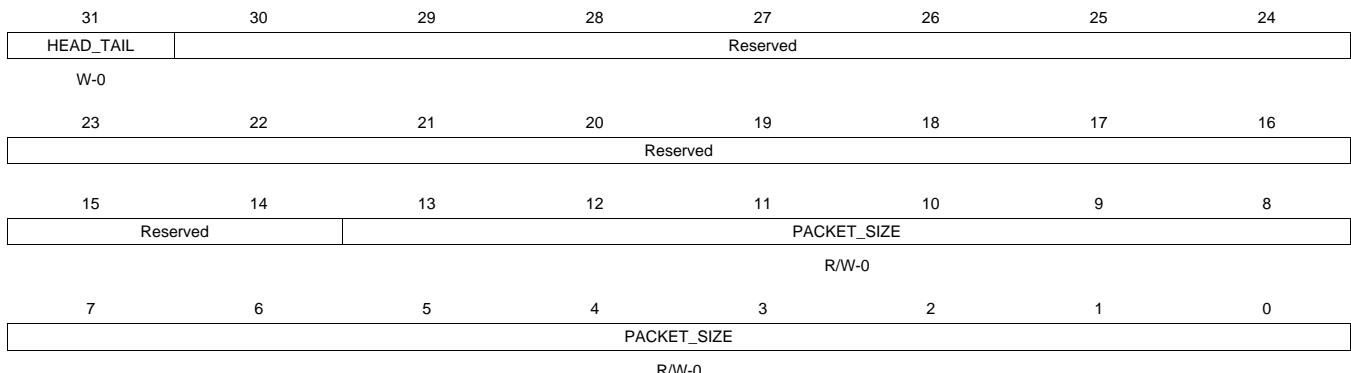
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-590. QUEUE\_66\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.301 QUEUE\_66\_C Register (offset = 2428h) [reset = 0h]**

QUEUE\_66\_C is shown in [Figure 16-577](#) and described in [Table 16-591](#).

**Figure 16-577. QUEUE\_66\_C Register**


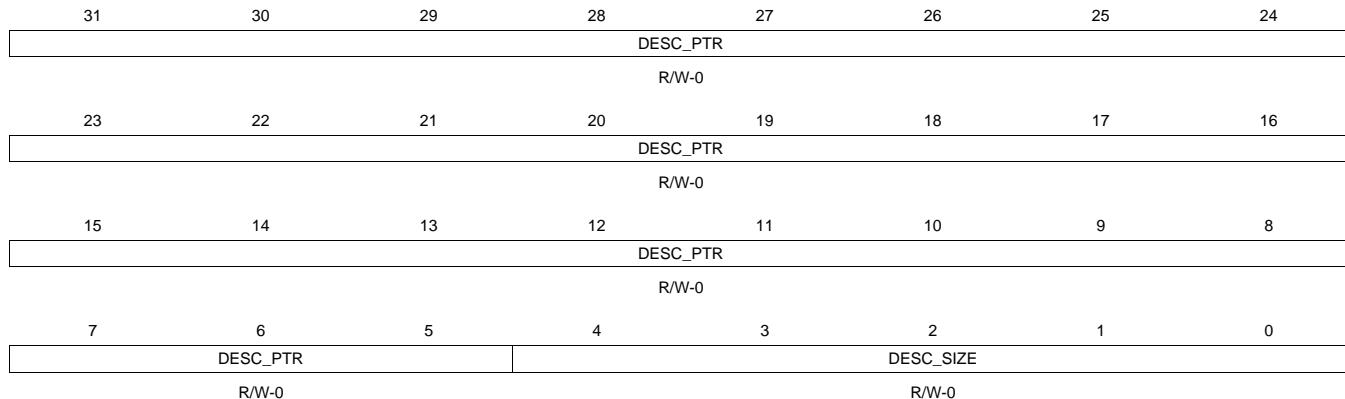
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-591. QUEUE\_66\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.302 QUEUE\_66\_D Register (offset = 242Ch) [reset = 0h]**

 QUEUE\_66\_D is shown in [Figure 16-578](#) and described in [Table 16-592](#).

**Figure 16-578. QUEUE\_66\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-592. QUEUE\_66\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.303 QUEUE\_67\_A Register (offset = 2430h) [reset = 0h]**

QUEUE\_67\_A is shown in [Figure 16-579](#) and described in [Table 16-593](#).

**Figure 16-579. QUEUE\_67\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-593. QUEUE\_67\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.304 QUEUE\_67\_B Register (offset = 2434h) [reset = 0h]**

QUEUE\_67\_B is shown in [Figure 16-580](#) and described in [Table 16-594](#).

**Figure 16-580. QUEUE\_67\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

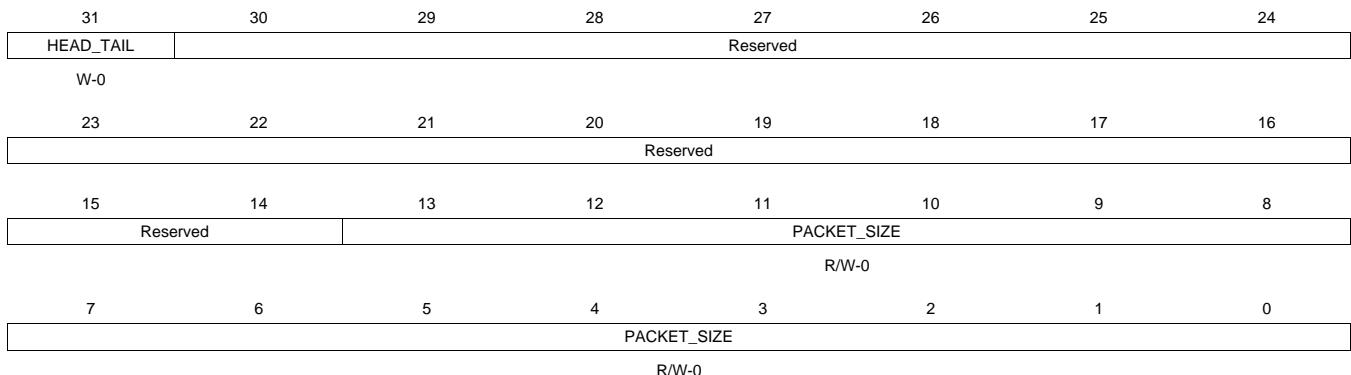
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-594. QUEUE\_67\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.305 QUEUE\_67\_C Register (offset = 2438h) [reset = 0h]**

QUEUE\_67\_C is shown in [Figure 16-581](#) and described in [Table 16-595](#).

**Figure 16-581. QUEUE\_67\_C Register**


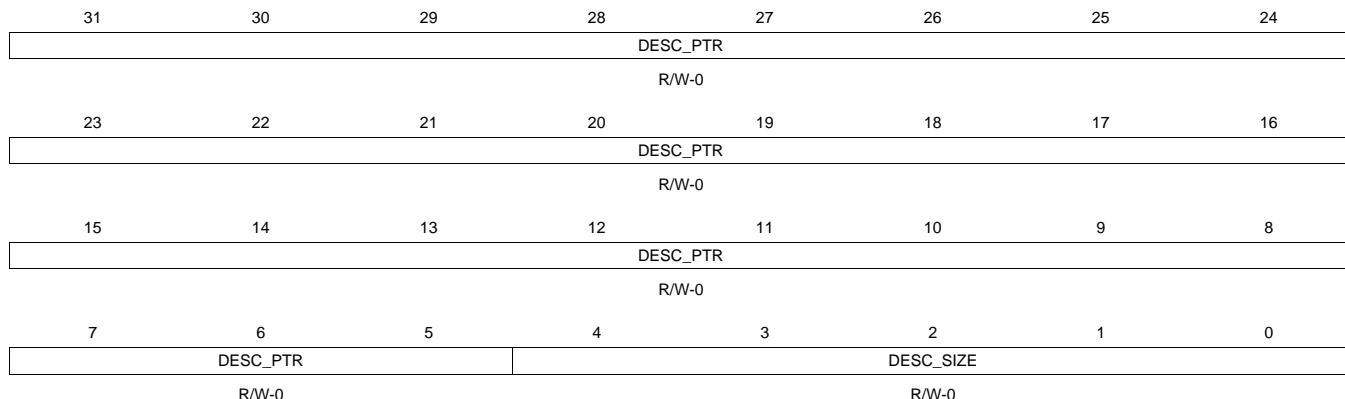
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-595. QUEUE\_67\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.306 QUEUE\_67\_D Register (offset = 243Ch) [reset = 0h]**

 QUEUE\_67\_D is shown in [Figure 16-582](#) and described in [Table 16-596](#).

**Figure 16-582. QUEUE\_67\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-596. QUEUE\_67\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.307 QUEUE\_68\_A Register (offset = 2440h) [reset = 0h]**

QUEUE\_68\_A is shown in [Figure 16-583](#) and described in [Table 16-597](#).

**Figure 16-583. QUEUE\_68\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-597. QUEUE\_68\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.308 QUEUE\_68\_B Register (offset = 2444h) [reset = 0h]**

QUEUE\_68\_B is shown in [Figure 16-584](#) and described in [Table 16-598](#).

**Figure 16-584. QUEUE\_68\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

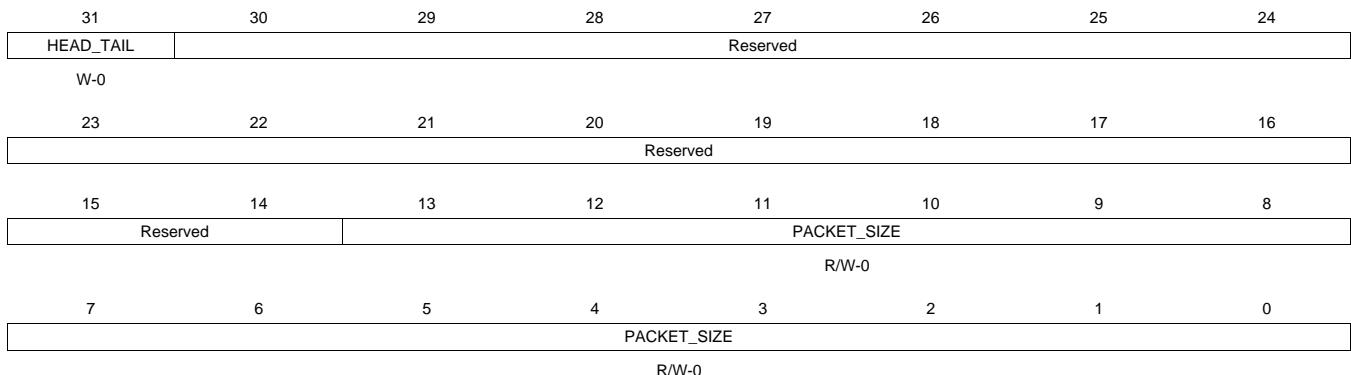
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-598. QUEUE\_68\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.309 QUEUE\_68\_C Register (offset = 2448h) [reset = 0h]**

QUEUE\_68\_C is shown in [Figure 16-585](#) and described in [Table 16-599](#).

**Figure 16-585. QUEUE\_68\_C Register**


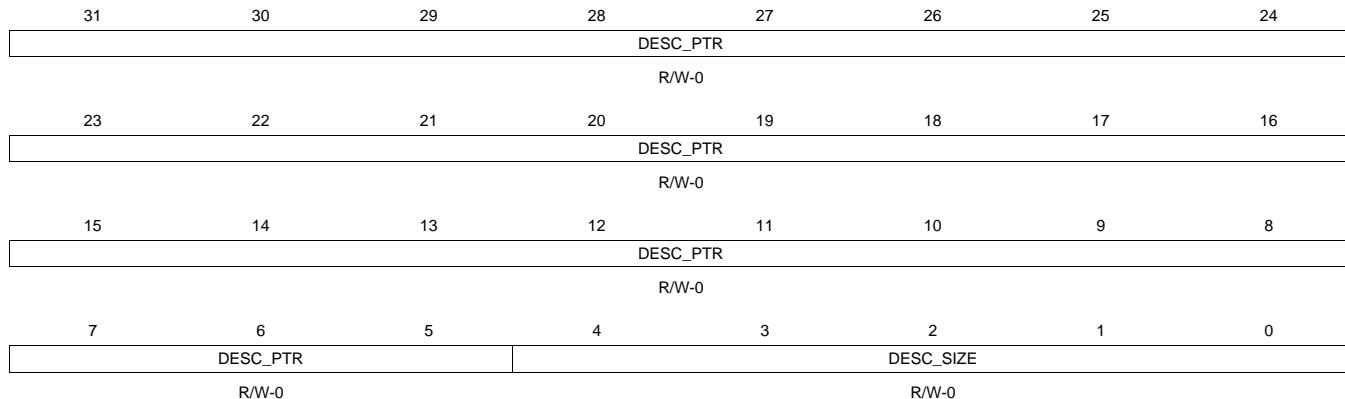
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-599. QUEUE\_68\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.310 QUEUE\_68\_D Register (offset = 244Ch) [reset = 0h]**

 QUEUE\_68\_D is shown in [Figure 16-586](#) and described in [Table 16-600](#).

**Figure 16-586. QUEUE\_68\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-600. QUEUE\_68\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.311 QUEUE\_69\_A Register (offset = 2450h) [reset = 0h]**

QUEUE\_69\_A is shown in [Figure 16-587](#) and described in [Table 16-601](#).

**Figure 16-587. QUEUE\_69\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-601. QUEUE\_69\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.312 QUEUE\_69\_B Register (offset = 2454h) [reset = 0h]**

QUEUE\_69\_B is shown in [Figure 16-588](#) and described in [Table 16-602](#).

**Figure 16-588. QUEUE\_69\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

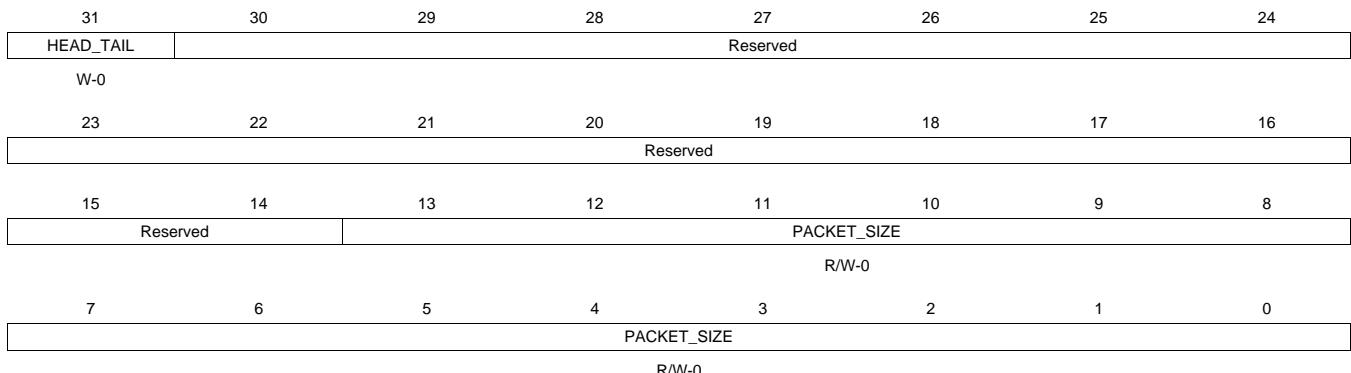
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-602. QUEUE\_69\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.313 QUEUE\_69\_C Register (offset = 2458h) [reset = 0h]**

QUEUE\_69\_C is shown in [Figure 16-589](#) and described in [Table 16-603](#).

**Figure 16-589. QUEUE\_69\_C Register**


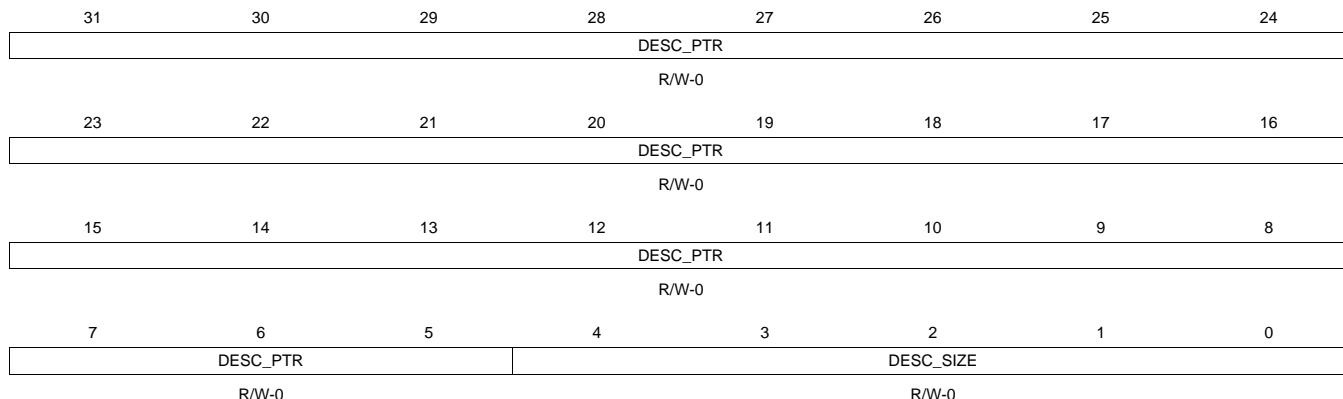
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-603. QUEUE\_69\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.314 QUEUE\_69\_D Register (offset = 245Ch) [reset = 0h]**

 QUEUE\_69\_D is shown in [Figure 16-590](#) and described in [Table 16-604](#).

**Figure 16-590. QUEUE\_69\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-604. QUEUE\_69\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.315 QUEUE\_70\_A Register (offset = 2460h) [reset = 0h]**

QUEUE\_70\_A is shown in [Figure 16-591](#) and described in [Table 16-605](#).

**Figure 16-591. QUEUE\_70\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-605. QUEUE\_70\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.316 QUEUE\_70\_B Register (offset = 2464h) [reset = 0h]**

QUEUE\_70\_B is shown in [Figure 16-592](#) and described in [Table 16-606](#).

**Figure 16-592. QUEUE\_70\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

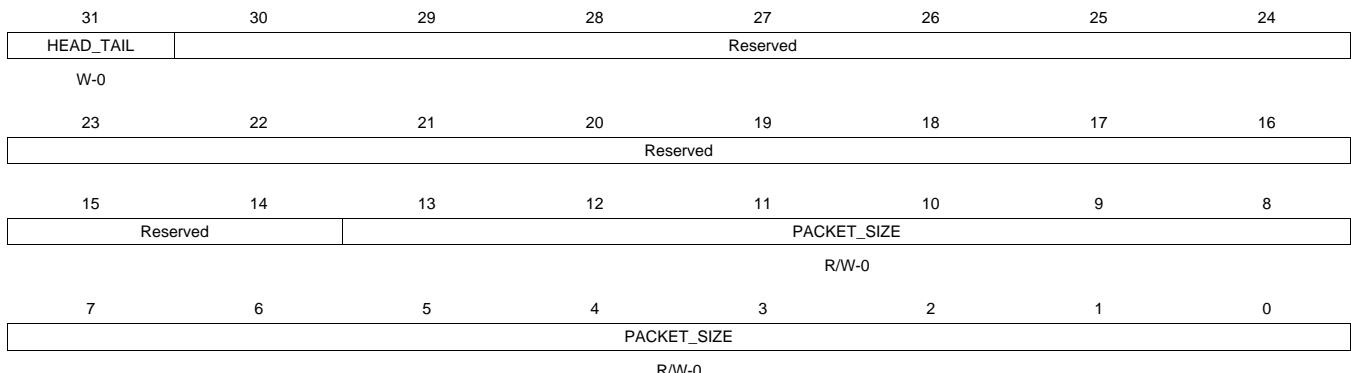
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-606. QUEUE\_70\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.317 QUEUE\_70\_C Register (offset = 2468h) [reset = 0h]**

QUEUE\_70\_C is shown in [Figure 16-593](#) and described in [Table 16-607](#).

**Figure 16-593. QUEUE\_70\_C Register**


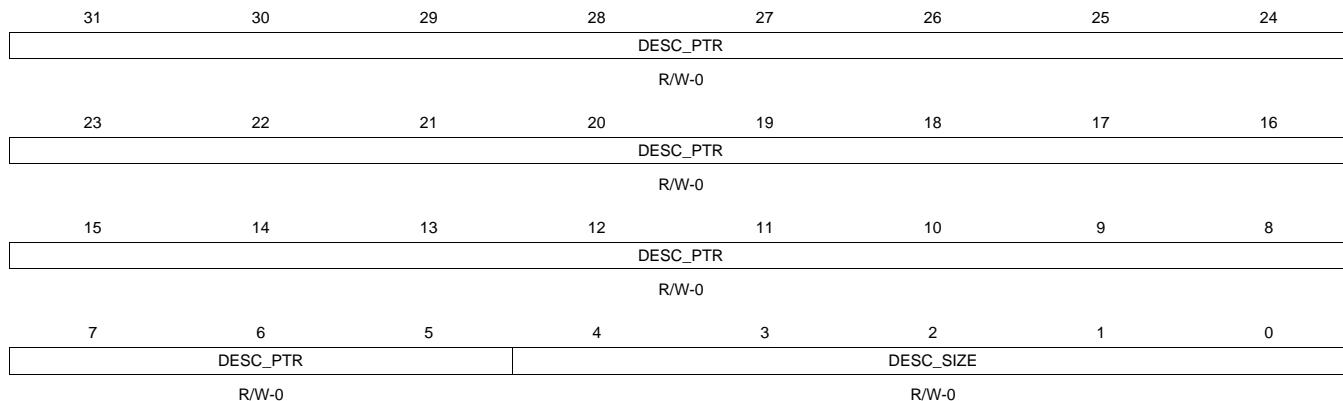
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-607. QUEUE\_70\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.318 QUEUE\_70\_D Register (offset = 246Ch) [reset = 0h]**

QUEUE\_70\_D is shown in [Figure 16-594](#) and described in [Table 16-608](#).

**Figure 16-594. QUEUE\_70\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-608. QUEUE\_70\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.319 QUEUE\_71\_A Register (offset = 2470h) [reset = 0h]**

QUEUE\_71\_A is shown in [Figure 16-595](#) and described in [Table 16-609](#).

**Figure 16-595. QUEUE\_71\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-609. QUEUE\_71\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.320 QUEUE\_71\_B Register (offset = 2474h) [reset = 0h]**

QUEUE\_71\_B is shown in [Figure 16-596](#) and described in [Table 16-610](#).

**Figure 16-596. QUEUE\_71\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

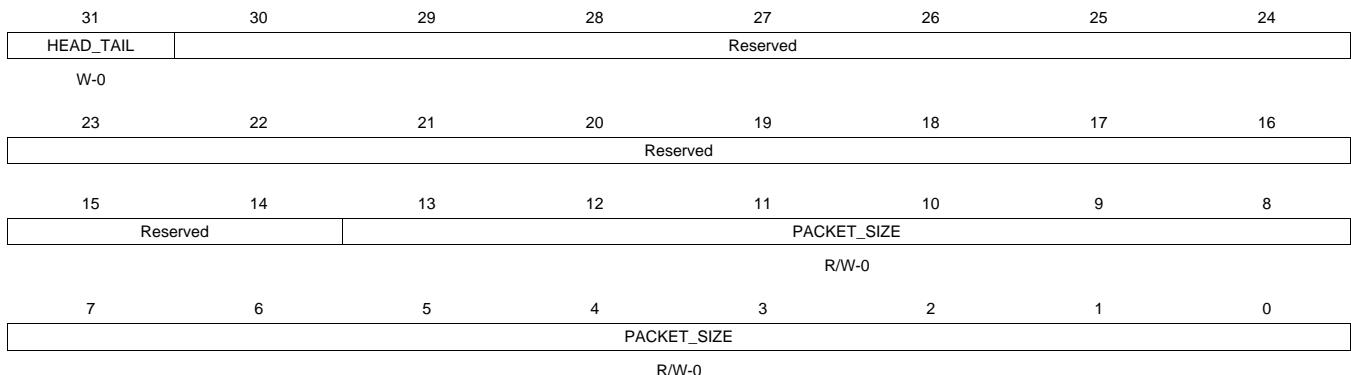
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-610. QUEUE\_71\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.321 QUEUE\_71\_C Register (offset = 2478h) [reset = 0h]**

QUEUE\_71\_C is shown in [Figure 16-597](#) and described in [Table 16-611](#).

**Figure 16-597. QUEUE\_71\_C Register**


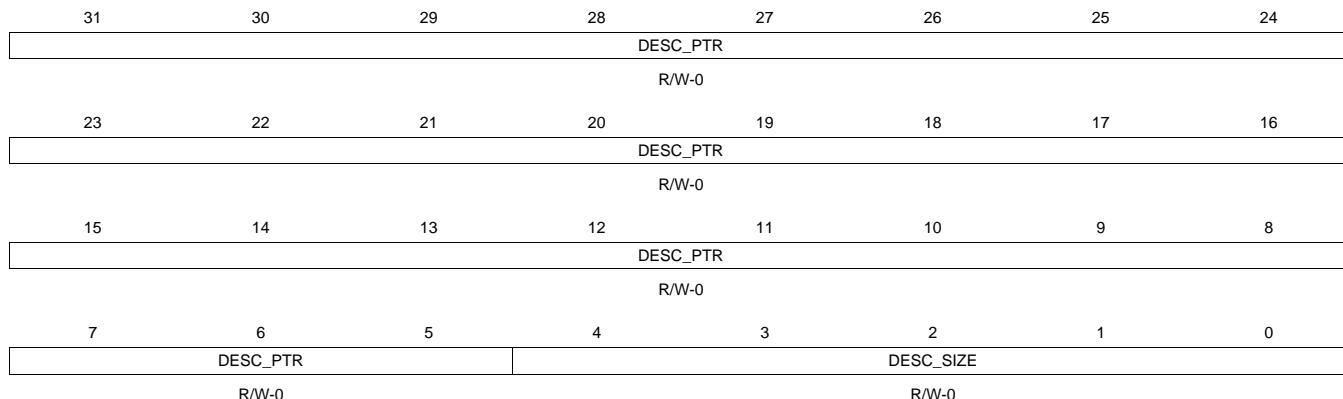
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-611. QUEUE\_71\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.322 QUEUE\_71\_D Register (offset = 247Ch) [reset = 0h]**

 QUEUE\_71\_D is shown in [Figure 16-598](#) and described in [Table 16-612](#).

**Figure 16-598. QUEUE\_71\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-612. QUEUE\_71\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.323 QUEUE\_72\_A Register (offset = 2480h) [reset = 0h]**

QUEUE\_72\_A is shown in [Figure 16-599](#) and described in [Table 16-613](#).

**Figure 16-599. QUEUE\_72\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-613. QUEUE\_72\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.324 QUEUE\_72\_B Register (offset = 2484h) [reset = 0h]**

QUEUE\_72\_B is shown in [Figure 16-600](#) and described in [Table 16-614](#).

**Figure 16-600. QUEUE\_72\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

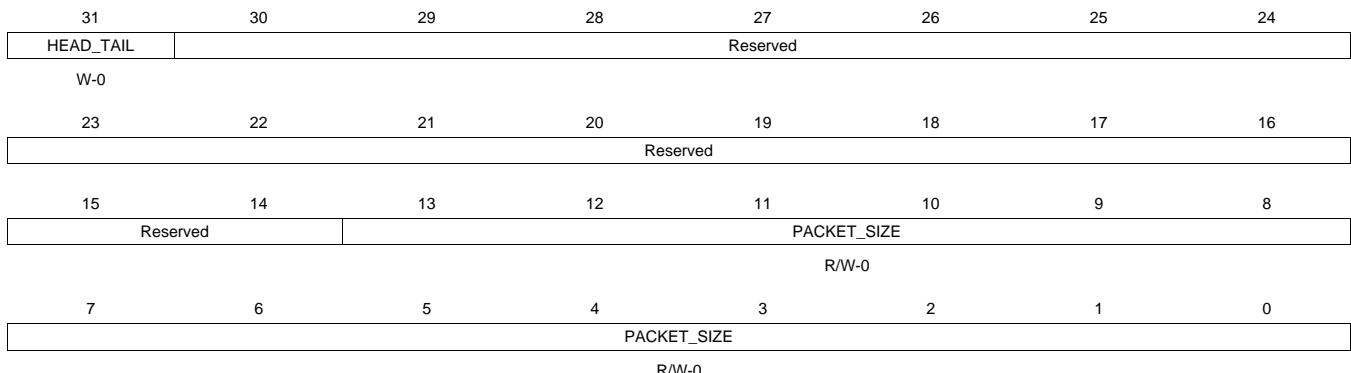
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-614. QUEUE\_72\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.325 QUEUE\_72\_C Register (offset = 2488h) [reset = 0h]**

QUEUE\_72\_C is shown in [Figure 16-601](#) and described in [Table 16-615](#).

**Figure 16-601. QUEUE\_72\_C Register**


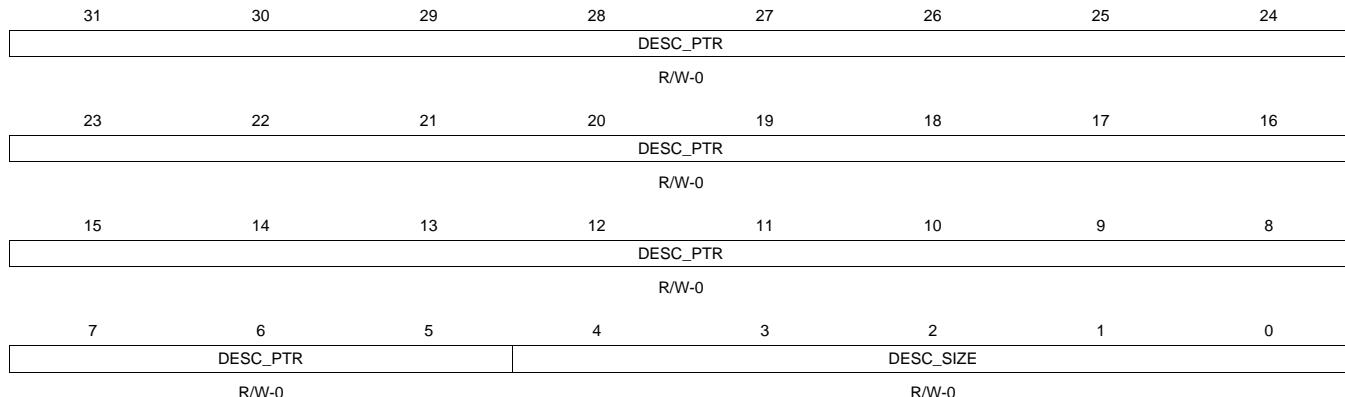
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-615. QUEUE\_72\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.326 QUEUE\_72\_D Register (offset = 248Ch) [reset = 0h]**

 QUEUE\_72\_D is shown in [Figure 16-602](#) and described in [Table 16-616](#).

**Figure 16-602. QUEUE\_72\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-616. QUEUE\_72\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.327 QUEUE\_73\_A Register (offset = 2490h) [reset = 0h]**

QUEUE\_73\_A is shown in [Figure 16-603](#) and described in [Table 16-617](#).

**Figure 16-603. QUEUE\_73\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-617. QUEUE\_73\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.328 QUEUE\_73\_B Register (offset = 2494h) [reset = 0h]**

QUEUE\_73\_B is shown in [Figure 16-604](#) and described in [Table 16-618](#).

**Figure 16-604. QUEUE\_73\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

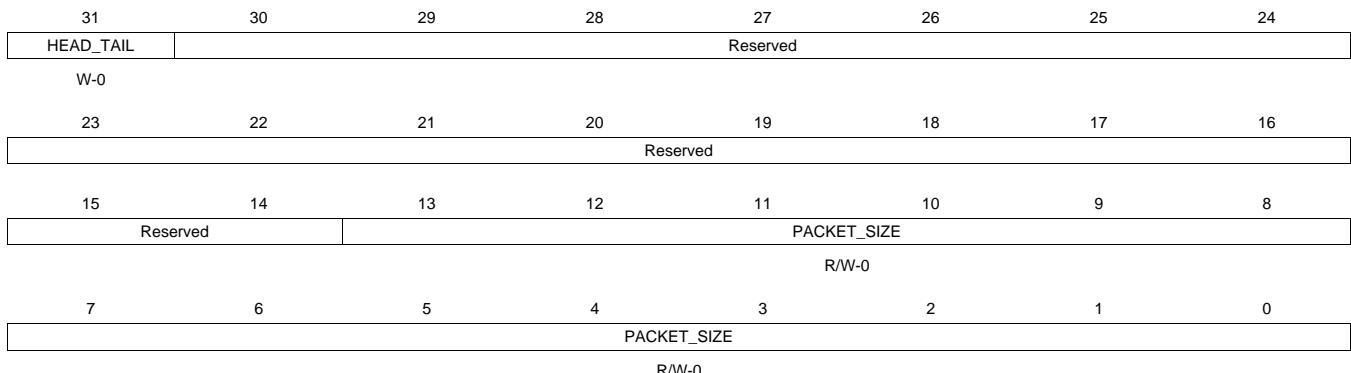
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-618. QUEUE\_73\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.329 QUEUE\_73\_C Register (offset = 2498h) [reset = 0h]**

QUEUE\_73\_C is shown in [Figure 16-605](#) and described in [Table 16-619](#).

**Figure 16-605. QUEUE\_73\_C Register**


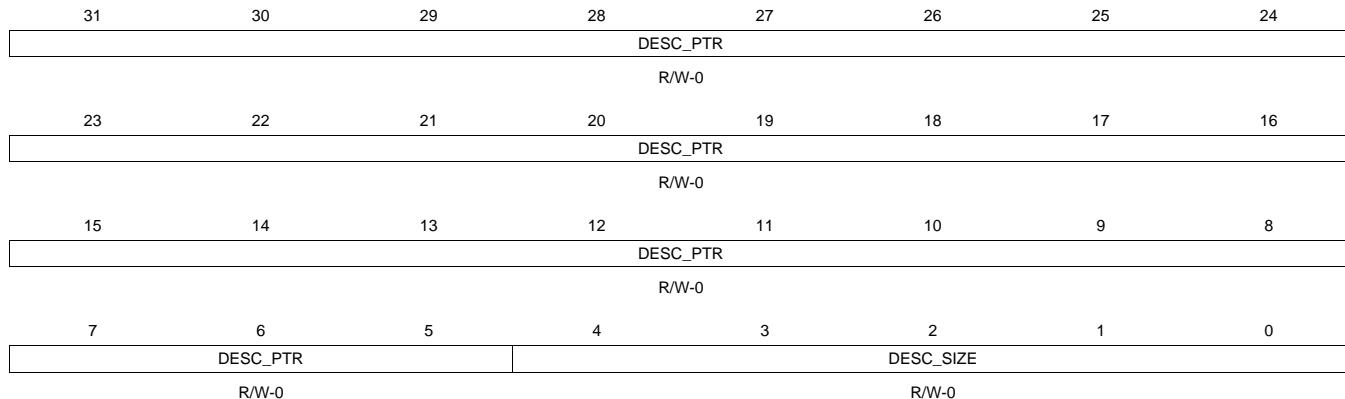
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-619. QUEUE\_73\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.330 QUEUE\_73\_D Register (offset = 249Ch) [reset = 0h]**

 QUEUE\_73\_D is shown in [Figure 16-606](#) and described in [Table 16-620](#).

**Figure 16-606. QUEUE\_73\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-620. QUEUE\_73\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.331 QUEUE\_74\_A Register (offset = 24A0h) [reset = 0h]**

QUEUE\_74\_A is shown in [Figure 16-607](#) and described in [Table 16-621](#).

**Figure 16-607. QUEUE\_74\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-621. QUEUE\_74\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.332 QUEUE\_74\_B Register (offset = 24A4h) [reset = 0h]**

QUEUE\_74\_B is shown in [Figure 16-608](#) and described in [Table 16-622](#).

**Figure 16-608. QUEUE\_74\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

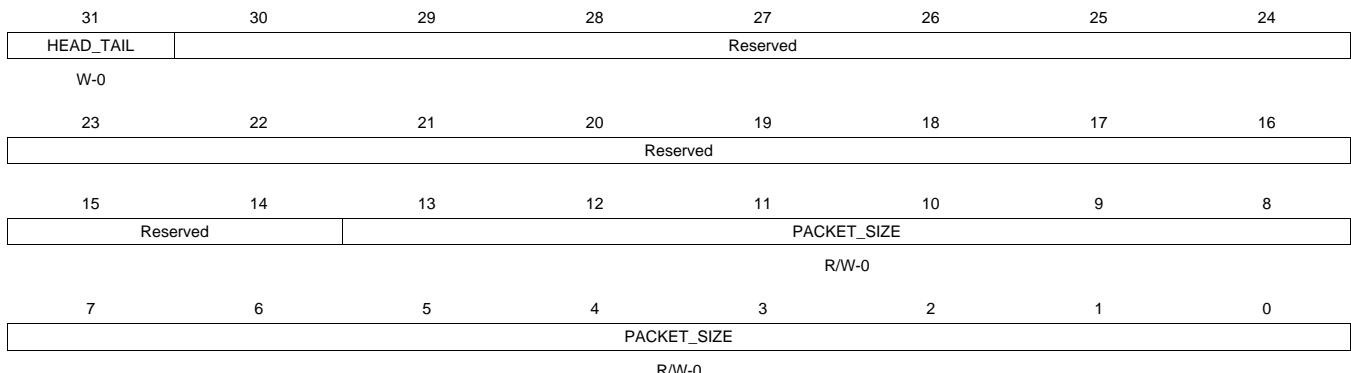
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-622. QUEUE\_74\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.333 QUEUE\_74\_C Register (offset = 24A8h) [reset = 0h]**

QUEUE\_74\_C is shown in [Figure 16-609](#) and described in [Table 16-623](#).

**Figure 16-609. QUEUE\_74\_C Register**


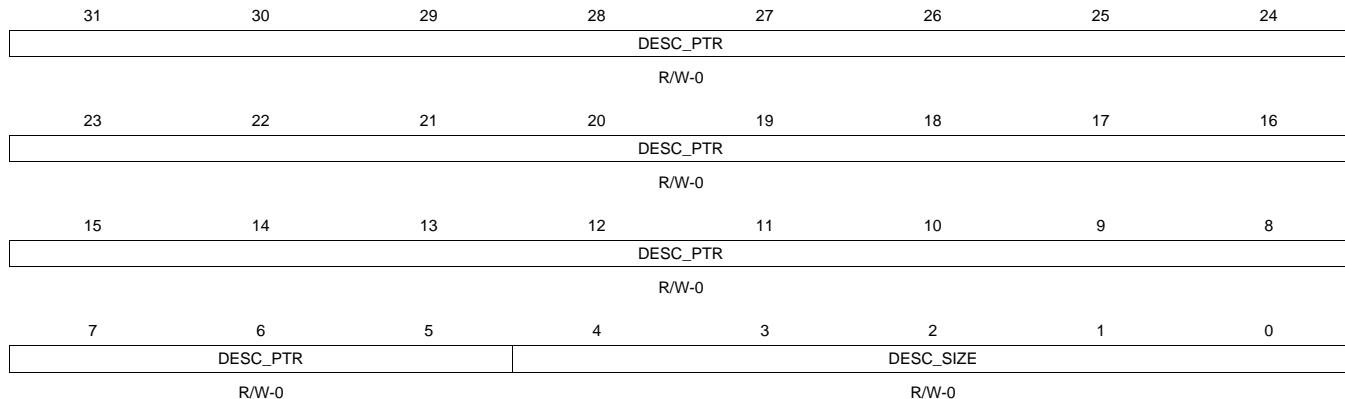
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-623. QUEUE\_74\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.334 QUEUE\_74\_D Register (offset = 24ACh) [reset = 0h]**

 QUEUE\_74\_D is shown in [Figure 16-610](#) and described in [Table 16-624](#).

**Figure 16-610. QUEUE\_74\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-624. QUEUE\_74\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.335 QUEUE\_75\_A Register (offset = 24B0h) [reset = 0h]**

QUEUE\_75\_A is shown in [Figure 16-611](#) and described in [Table 16-625](#).

**Figure 16-611. QUEUE\_75\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-625. QUEUE\_75\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.336 QUEUE\_75\_B Register (offset = 24B4h) [reset = 0h]**

QUEUE\_75\_B is shown in [Figure 16-612](#) and described in [Table 16-626](#).

**Figure 16-612. QUEUE\_75\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

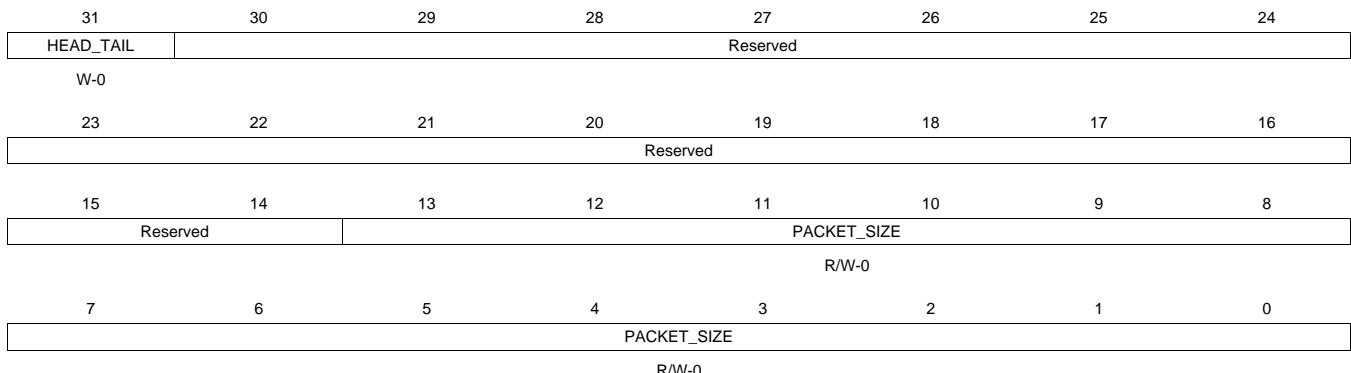
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-626. QUEUE\_75\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.337 QUEUE\_75\_C Register (offset = 24B8h) [reset = 0h]**

QUEUE\_75\_C is shown in [Figure 16-613](#) and described in [Table 16-627](#).

**Figure 16-613. QUEUE\_75\_C Register**


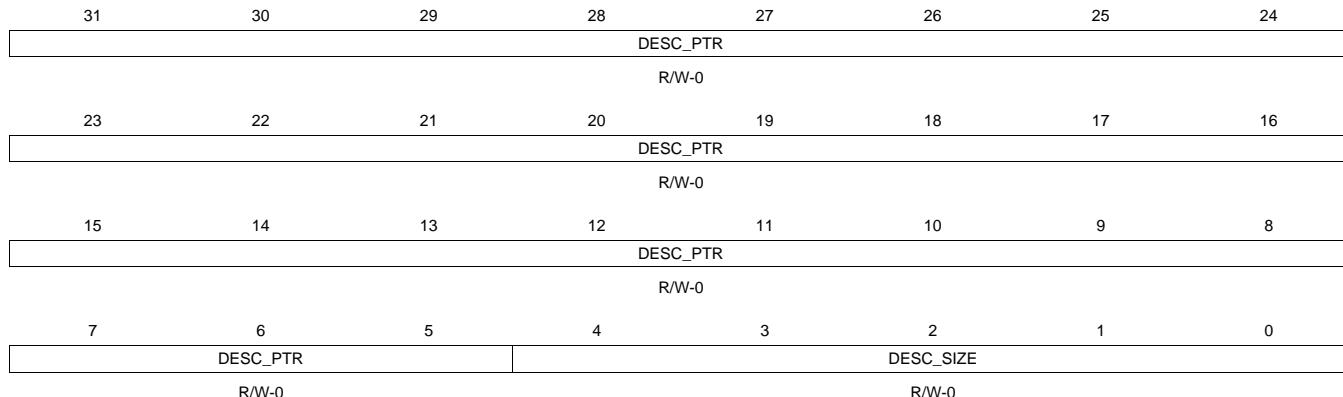
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-627. QUEUE\_75\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.338 QUEUE\_75\_D Register (offset = 24BCh) [reset = 0h]**

 QUEUE\_75\_D is shown in [Figure 16-614](#) and described in [Table 16-628](#).

**Figure 16-614. QUEUE\_75\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-628. QUEUE\_75\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.339 QUEUE\_76\_A Register (offset = 24C0h) [reset = 0h]**

QUEUE\_76\_A is shown in [Figure 16-615](#) and described in [Table 16-629](#).

**Figure 16-615. QUEUE\_76\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-629. QUEUE\_76\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.340 QUEUE\_76\_B Register (offset = 24C4h) [reset = 0h]**

QUEUE\_76\_B is shown in [Figure 16-616](#) and described in [Table 16-630](#).

**Figure 16-616. QUEUE\_76\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

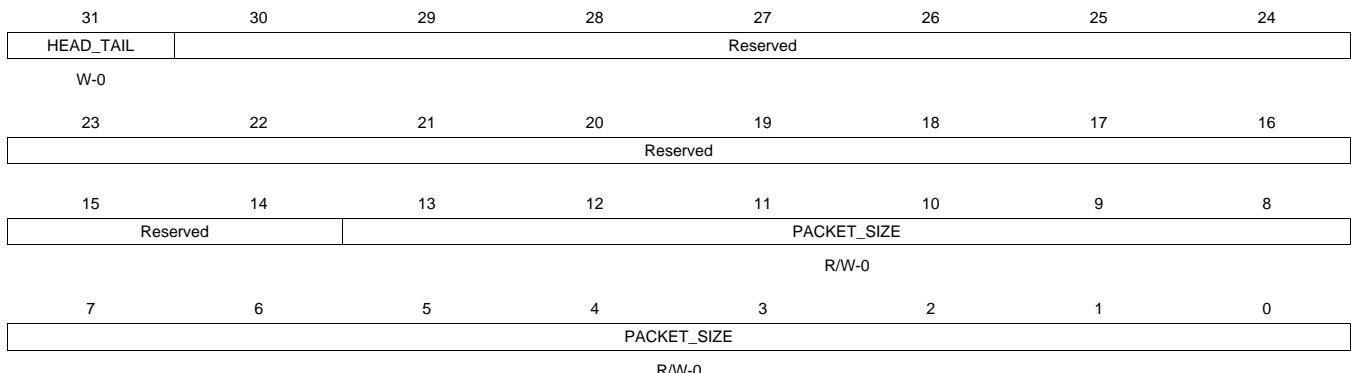
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-630. QUEUE\_76\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.341 QUEUE\_76\_C Register (offset = 24C8h) [reset = 0h]**

QUEUE\_76\_C is shown in [Figure 16-617](#) and described in [Table 16-631](#).

**Figure 16-617. QUEUE\_76\_C Register**


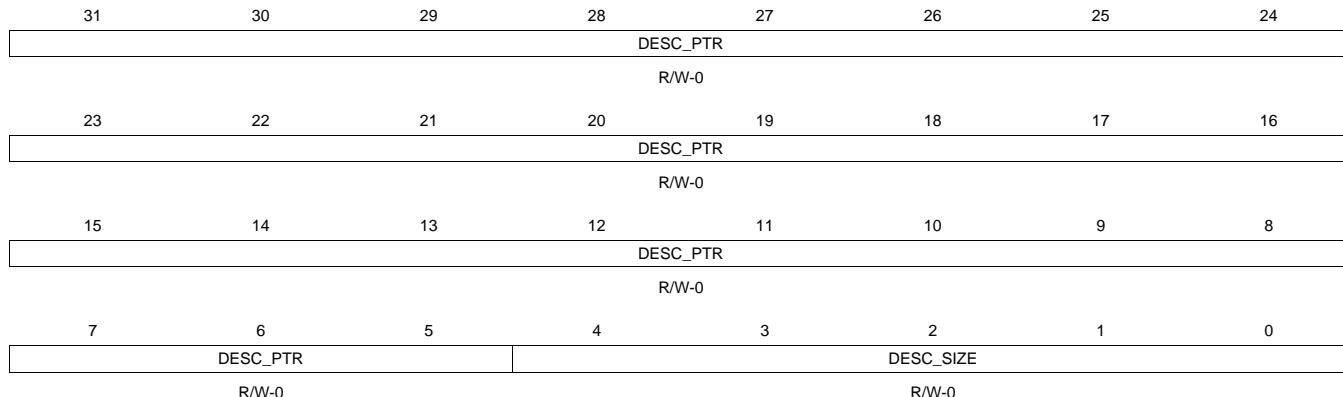
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-631. QUEUE\_76\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.342 QUEUE\_76\_D Register (offset = 24CCh) [reset = 0h]**

 QUEUE\_76\_D is shown in [Figure 16-618](#) and described in [Table 16-632](#).

**Figure 16-618. QUEUE\_76\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-632. QUEUE\_76\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.343 QUEUE\_77\_A Register (offset = 24D0h) [reset = 0h]**

QUEUE\_77\_A is shown in [Figure 16-619](#) and described in [Table 16-633](#).

**Figure 16-619. QUEUE\_77\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-633. QUEUE\_77\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.344 QUEUE\_77\_B Register (offset = 24D4h) [reset = 0h]**

QUEUE\_77\_B is shown in [Figure 16-620](#) and described in [Table 16-634](#).

**Figure 16-620. QUEUE\_77\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

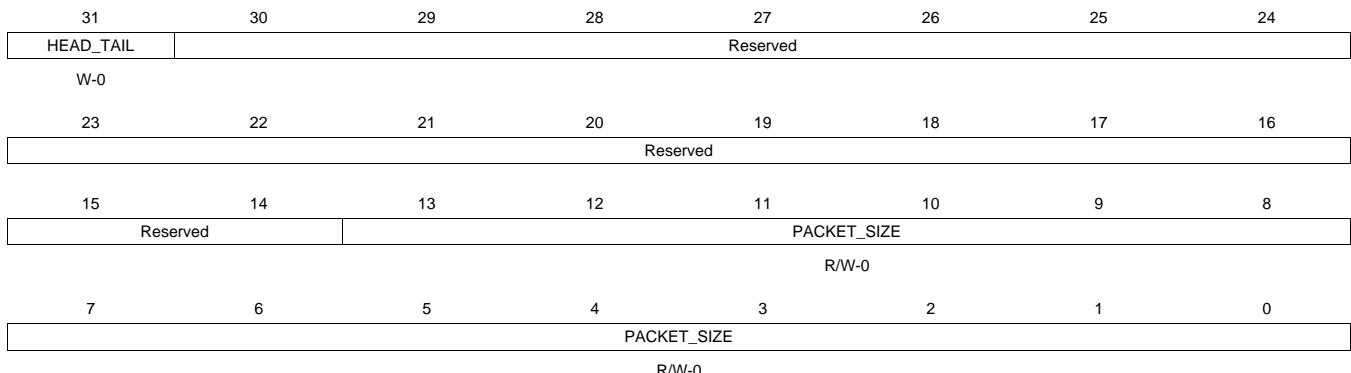
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-634. QUEUE\_77\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.345 QUEUE\_77\_C Register (offset = 24D8h) [reset = 0h]**

QUEUE\_77\_C is shown in [Figure 16-621](#) and described in [Table 16-635](#).

**Figure 16-621. QUEUE\_77\_C Register**


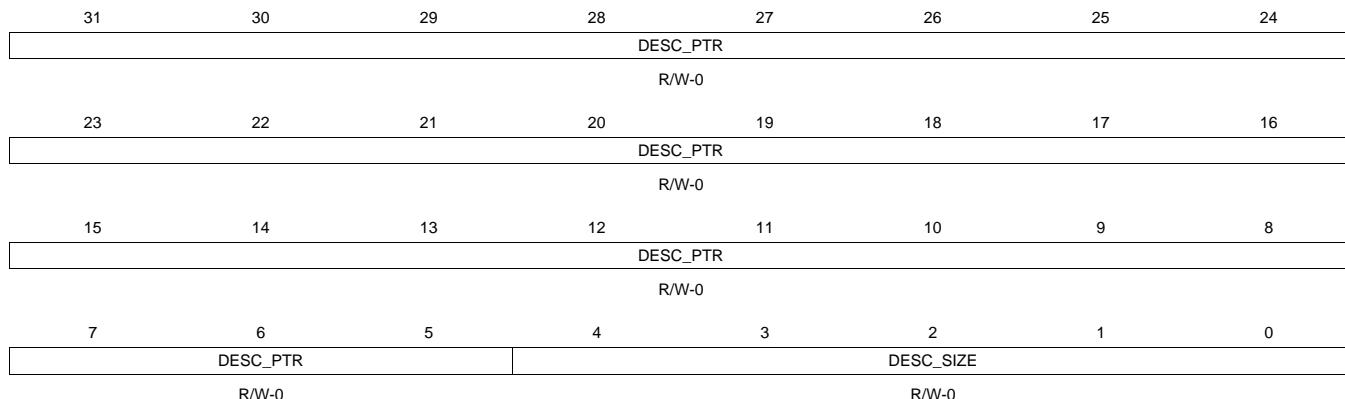
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-635. QUEUE\_77\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.346 QUEUE\_77\_D Register (offset = 24DCh) [reset = 0h]**

 QUEUE\_77\_D is shown in [Figure 16-622](#) and described in [Table 16-636](#).

**Figure 16-622. QUEUE\_77\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-636. QUEUE\_77\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.347 QUEUE\_78\_A Register (offset = 24E0h) [reset = 0h]**

QUEUE\_78\_A is shown in [Figure 16-623](#) and described in [Table 16-637](#).

**Figure 16-623. QUEUE\_78\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-637. QUEUE\_78\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.348 QUEUE\_78\_B Register (offset = 24E4h) [reset = 0h]**

QUEUE\_78\_B is shown in [Figure 16-624](#) and described in [Table 16-638](#).

**Figure 16-624. QUEUE\_78\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

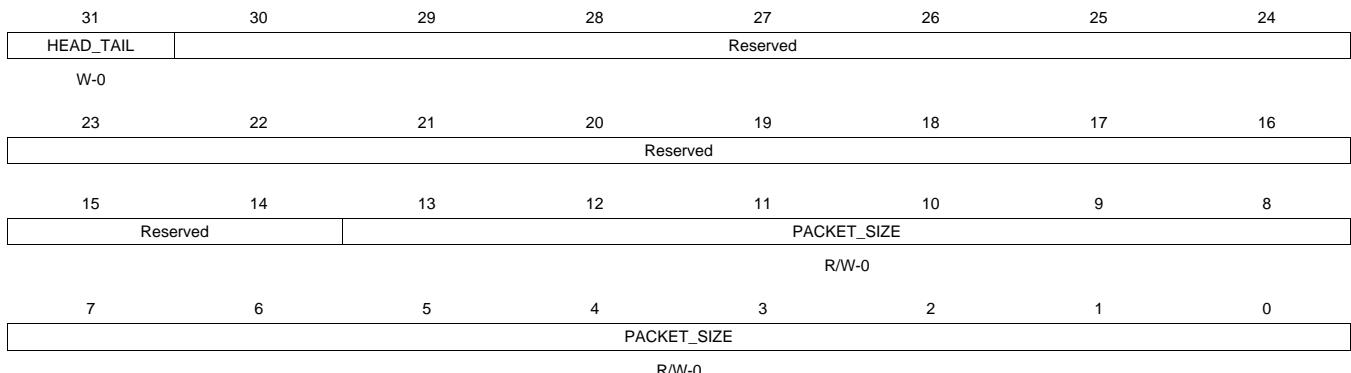
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-638. QUEUE\_78\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.349 QUEUE\_78\_C Register (offset = 24E8h) [reset = 0h]**

QUEUE\_78\_C is shown in [Figure 16-625](#) and described in [Table 16-639](#).

**Figure 16-625. QUEUE\_78\_C Register**


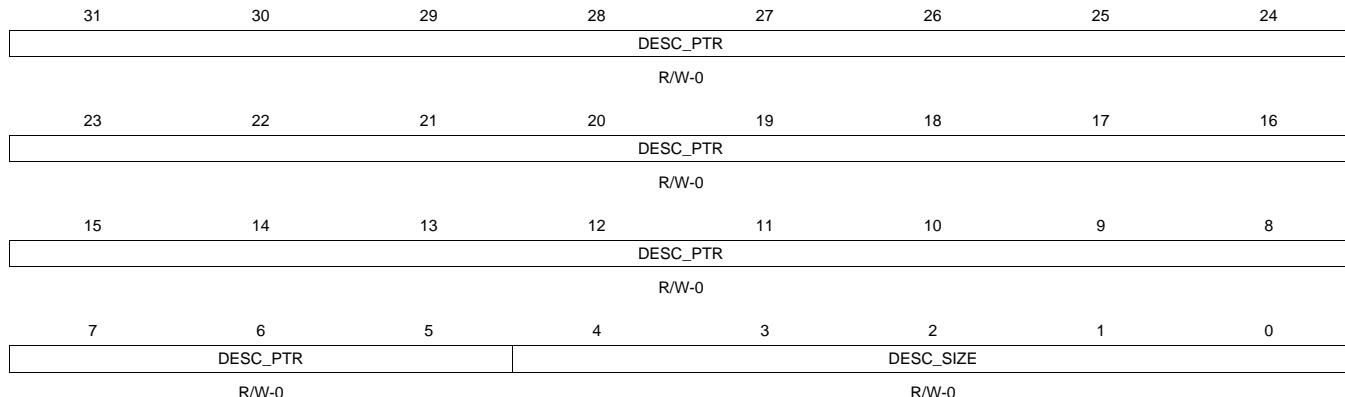
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-639. QUEUE\_78\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.350 QUEUE\_78\_D Register (offset = 24ECh) [reset = 0h]**

 QUEUE\_78\_D is shown in [Figure 16-626](#) and described in [Table 16-640](#).

**Figure 16-626. QUEUE\_78\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-640. QUEUE\_78\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.351 QUEUE\_79\_A Register (offset = 24F0h) [reset = 0h]**

QUEUE\_79\_A is shown in [Figure 16-627](#) and described in [Table 16-641](#).

**Figure 16-627. QUEUE\_79\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-641. QUEUE\_79\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.352 QUEUE\_79\_B Register (offset = 24F4h) [reset = 0h]**

QUEUE\_79\_B is shown in [Figure 16-628](#) and described in [Table 16-642](#).

**Figure 16-628. QUEUE\_79\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

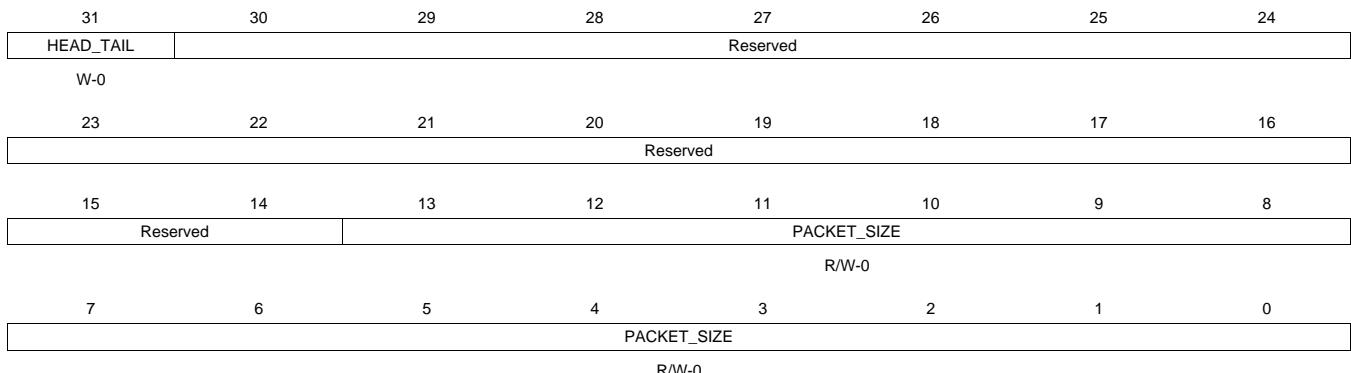
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-642. QUEUE\_79\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.353 QUEUE\_79\_C Register (offset = 24F8h) [reset = 0h]**

QUEUE\_79\_C is shown in [Figure 16-629](#) and described in [Table 16-643](#).

**Figure 16-629. QUEUE\_79\_C Register**


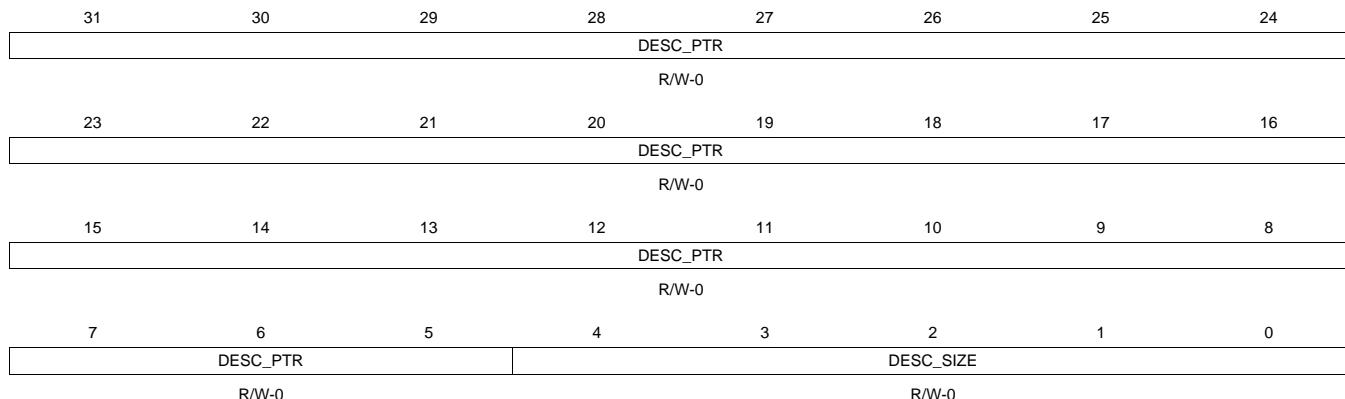
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-643. QUEUE\_79\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.354 QUEUE\_79\_D Register (offset = 24FCh) [reset = 0h]**

 QUEUE\_79\_D is shown in [Figure 16-630](#) and described in [Table 16-644](#).

**Figure 16-630. QUEUE\_79\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-644. QUEUE\_79\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.355 QUEUE\_80\_A Register (offset = 2500h) [reset = 0h]**

QUEUE\_80\_A is shown in [Figure 16-631](#) and described in [Table 16-645](#).

**Figure 16-631. QUEUE\_80\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-645. QUEUE\_80\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.356 QUEUE\_80\_B Register (offset = 2504h) [reset = 0h]**

QUEUE\_80\_B is shown in [Figure 16-632](#) and described in [Table 16-646](#).

**Figure 16-632. QUEUE\_80\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

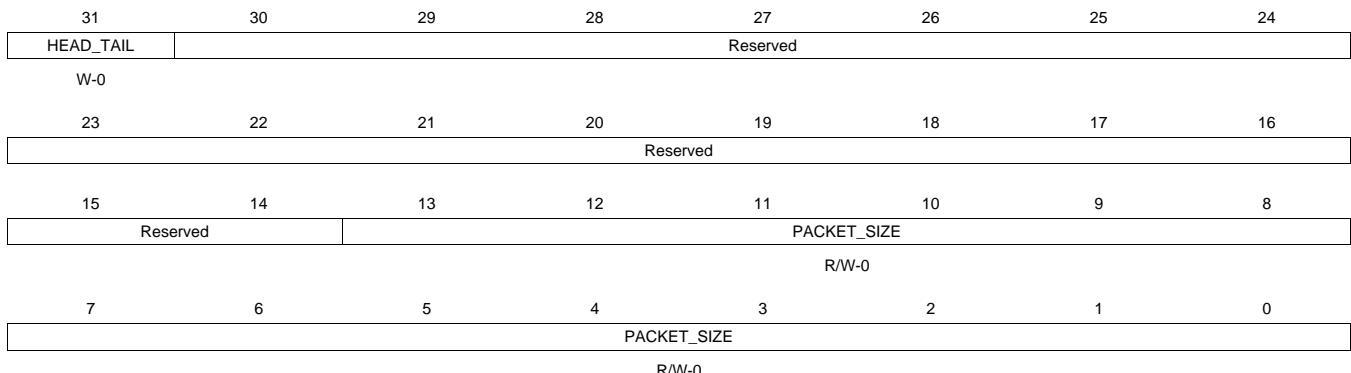
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-646. QUEUE\_80\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.357 QUEUE\_80\_C Register (offset = 2508h) [reset = 0h]**

QUEUE\_80\_C is shown in [Figure 16-633](#) and described in [Table 16-647](#).

**Figure 16-633. QUEUE\_80\_C Register**


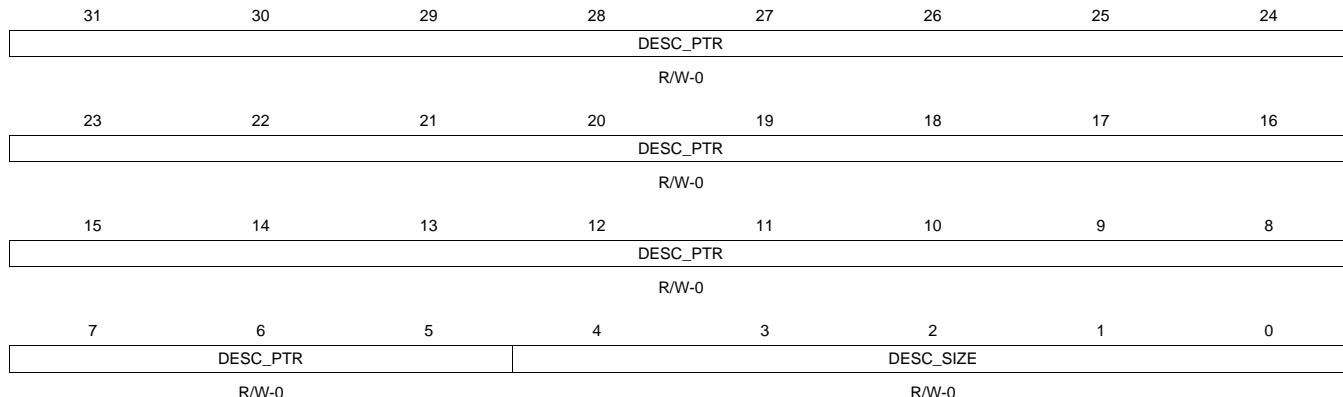
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-647. QUEUE\_80\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.358 QUEUE\_80\_D Register (offset = 250Ch) [reset = 0h]**

 QUEUE\_80\_D is shown in [Figure 16-634](#) and described in [Table 16-648](#).

**Figure 16-634. QUEUE\_80\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-648. QUEUE\_80\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.359 QUEUE\_81\_A Register (offset = 2510h) [reset = 0h]**

QUEUE\_81\_A is shown in [Figure 16-635](#) and described in [Table 16-649](#).

**Figure 16-635. QUEUE\_81\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-649. QUEUE\_81\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.360 QUEUE\_81\_B Register (offset = 2514h) [reset = 0h]**

QUEUE\_81\_B is shown in [Figure 16-636](#) and described in [Table 16-650](#).

**Figure 16-636. QUEUE\_81\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

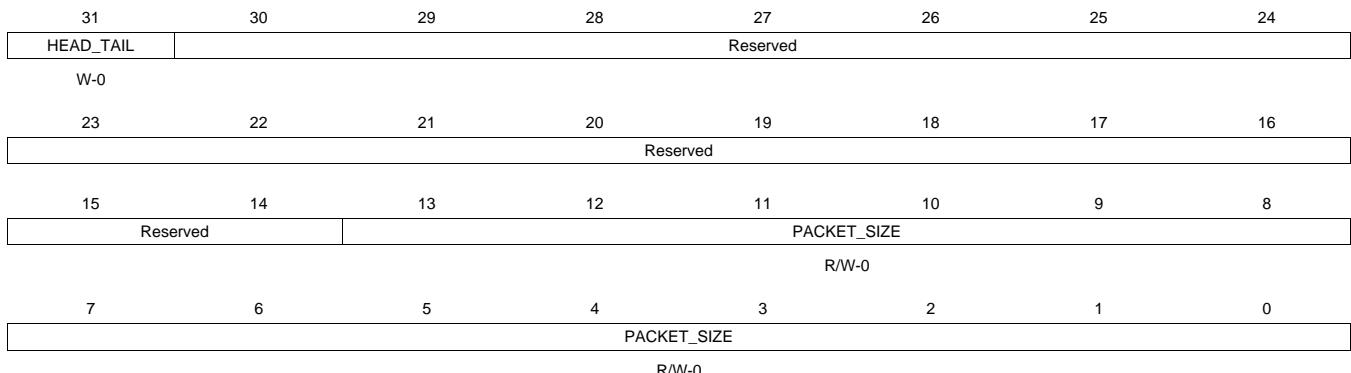
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-650. QUEUE\_81\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.361 QUEUE\_81\_C Register (offset = 2518h) [reset = 0h]**

QUEUE\_81\_C is shown in [Figure 16-637](#) and described in [Table 16-651](#).

**Figure 16-637. QUEUE\_81\_C Register**


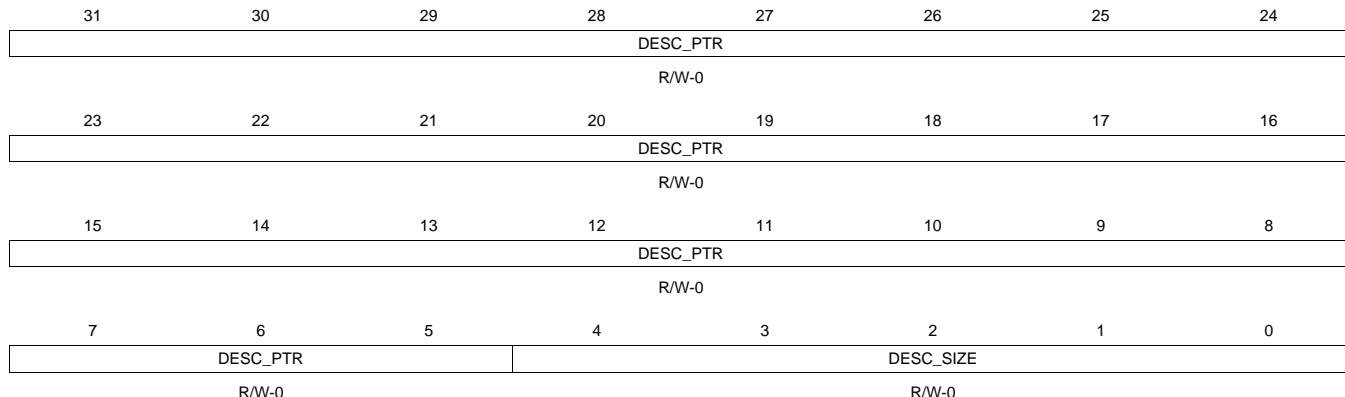
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-651. QUEUE\_81\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.362 QUEUE\_81\_D Register (offset = 251Ch) [reset = 0h]**

 QUEUE\_81\_D is shown in [Figure 16-638](#) and described in [Table 16-652](#).

**Figure 16-638. QUEUE\_81\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-652. QUEUE\_81\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.363 QUEUE\_82\_A Register (offset = 2520h) [reset = 0h]**

QUEUE\_82\_A is shown in [Figure 16-639](#) and described in [Table 16-653](#).

**Figure 16-639. QUEUE\_82\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-653. QUEUE\_82\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.364 QUEUE\_82\_B Register (offset = 2524h) [reset = 0h]**

QUEUE\_82\_B is shown in [Figure 16-640](#) and described in [Table 16-654](#).

**Figure 16-640. QUEUE\_82\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

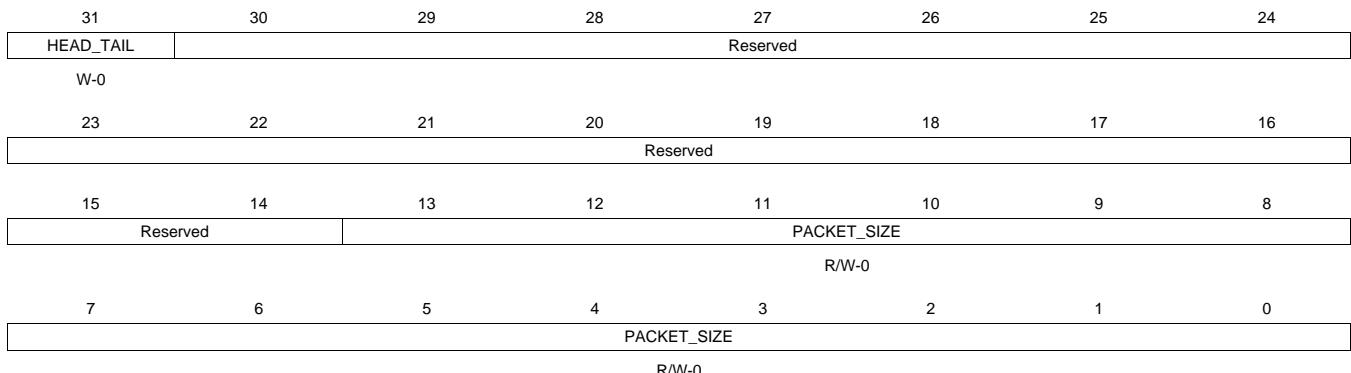
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-654. QUEUE\_82\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.365 QUEUE\_82\_C Register (offset = 2528h) [reset = 0h]**

QUEUE\_82\_C is shown in [Figure 16-641](#) and described in [Table 16-655](#).

**Figure 16-641. QUEUE\_82\_C Register**


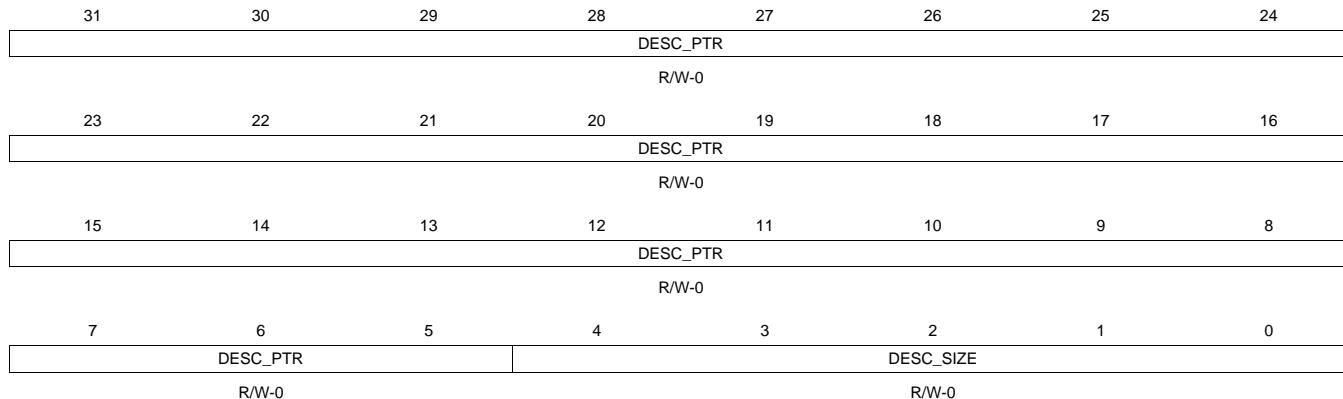
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-655. QUEUE\_82\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.366 QUEUE\_82\_D Register (offset = 252Ch) [reset = 0h]**

QUEUE\_82\_D is shown in [Figure 16-642](#) and described in [Table 16-656](#).

**Figure 16-642. QUEUE\_82\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-656. QUEUE\_82\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.367 QUEUE\_83\_A Register (offset = 2530h) [reset = 0h]**

QUEUE\_83\_A is shown in [Figure 16-643](#) and described in [Table 16-657](#).

**Figure 16-643. QUEUE\_83\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-657. QUEUE\_83\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.368 QUEUE\_83\_B Register (offset = 2534h) [reset = 0h]**

QUEUE\_83\_B is shown in [Figure 16-644](#) and described in [Table 16-658](#).

**Figure 16-644. QUEUE\_83\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

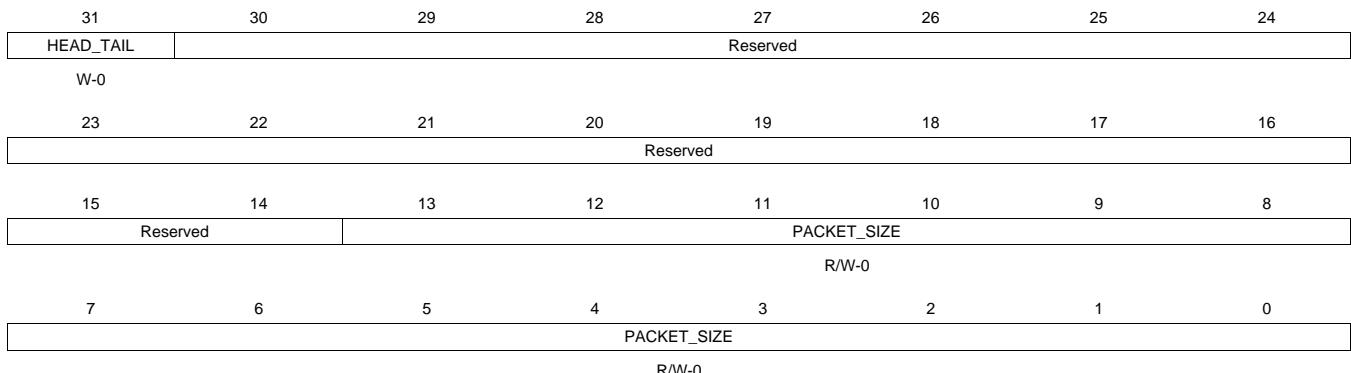
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-658. QUEUE\_83\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.369 QUEUE\_83\_C Register (offset = 2538h) [reset = 0h]**

QUEUE\_83\_C is shown in [Figure 16-645](#) and described in [Table 16-659](#).

**Figure 16-645. QUEUE\_83\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

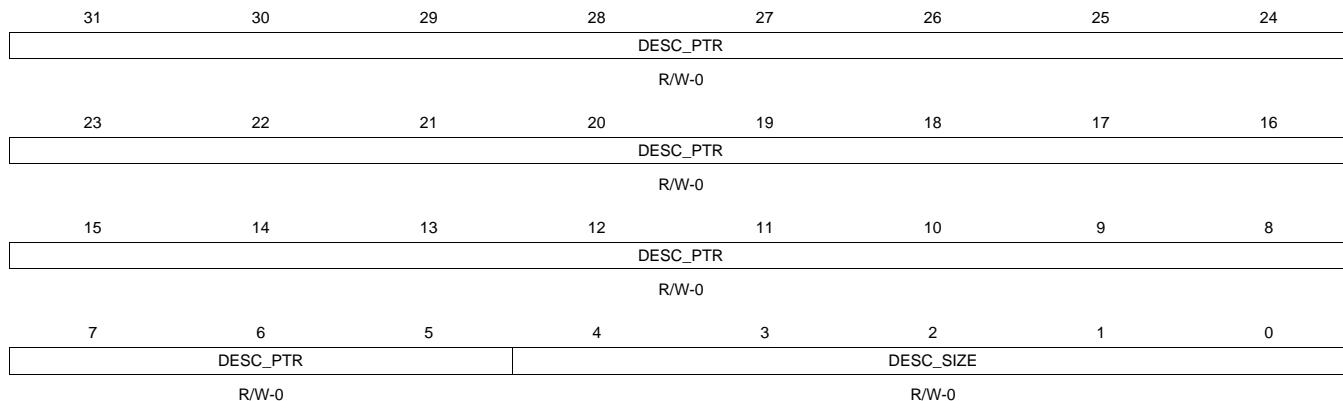
**Table 16-659. QUEUE\_83\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.370 QUEUE\_83\_D Register (offset = 253Ch) [reset = 0h]

QUEUE\_83\_D is shown in [Figure 16-646](#) and described in [Table 16-660](#).

**Figure 16-646. QUEUE\_83\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-660. QUEUE\_83\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.371 QUEUE\_84\_A Register (offset = 2540h) [reset = 0h]**

QUEUE\_84\_A is shown in [Figure 16-647](#) and described in [Table 16-661](#).

**Figure 16-647. QUEUE\_84\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-661. QUEUE\_84\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.372 QUEUE\_84\_B Register (offset = 2544h) [reset = 0h]**

QUEUE\_84\_B is shown in [Figure 16-648](#) and described in [Table 16-662](#).

**Figure 16-648. QUEUE\_84\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

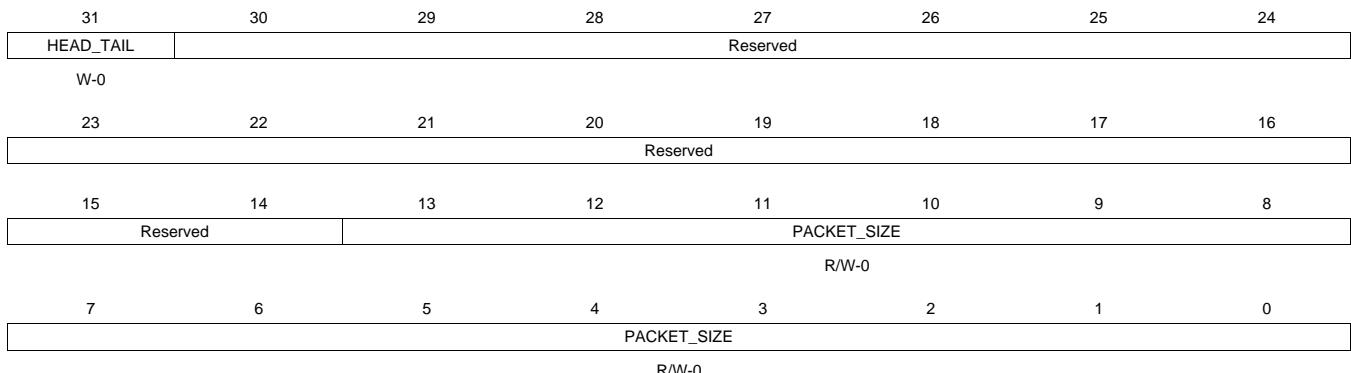
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-662. QUEUE\_84\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.373 QUEUE\_84\_C Register (offset = 2548h) [reset = 0h]**

QUEUE\_84\_C is shown in [Figure 16-649](#) and described in [Table 16-663](#).

**Figure 16-649. QUEUE\_84\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

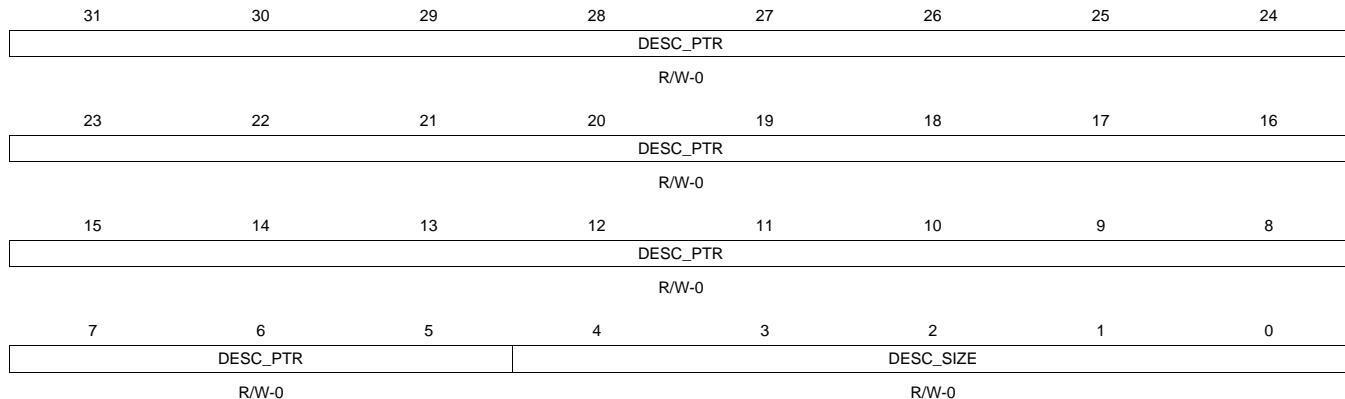
**Table 16-663. QUEUE\_84\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.374 QUEUE\_84\_D Register (offset = 254Ch) [reset = 0h]

QUEUE\_84\_D is shown in [Figure 16-650](#) and described in [Table 16-664](#).

**Figure 16-650. QUEUE\_84\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-664. QUEUE\_84\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.375 QUEUE\_85\_A Register (offset = 2550h) [reset = 0h]**

QUEUE\_85\_A is shown in [Figure 16-651](#) and described in [Table 16-665](#).

**Figure 16-651. QUEUE\_85\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-665. QUEUE\_85\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.376 QUEUE\_85\_B Register (offset = 2554h) [reset = 0h]**

QUEUE\_85\_B is shown in [Figure 16-652](#) and described in [Table 16-666](#).

**Figure 16-652. QUEUE\_85\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

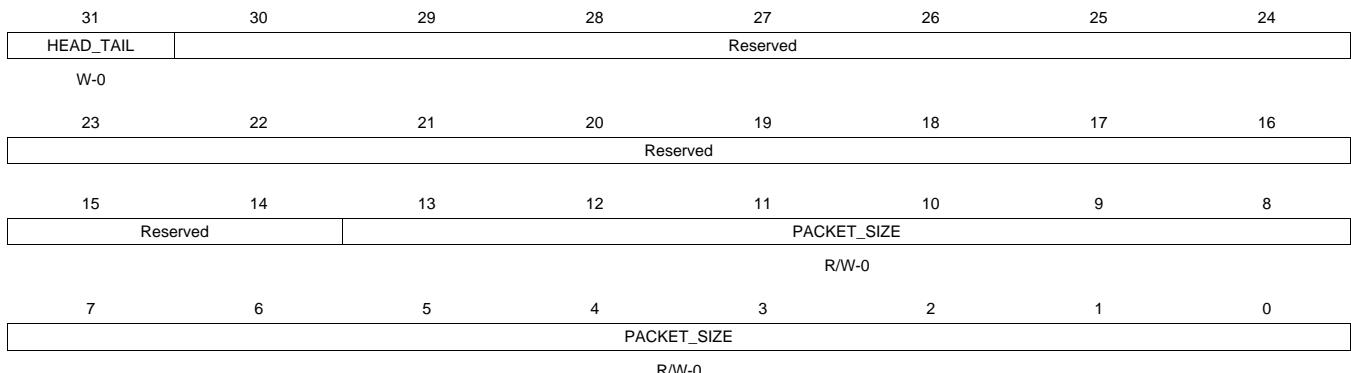
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-666. QUEUE\_85\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.377 QUEUE\_85\_C Register (offset = 2558h) [reset = 0h]**

QUEUE\_85\_C is shown in [Figure 16-653](#) and described in [Table 16-667](#).

**Figure 16-653. QUEUE\_85\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

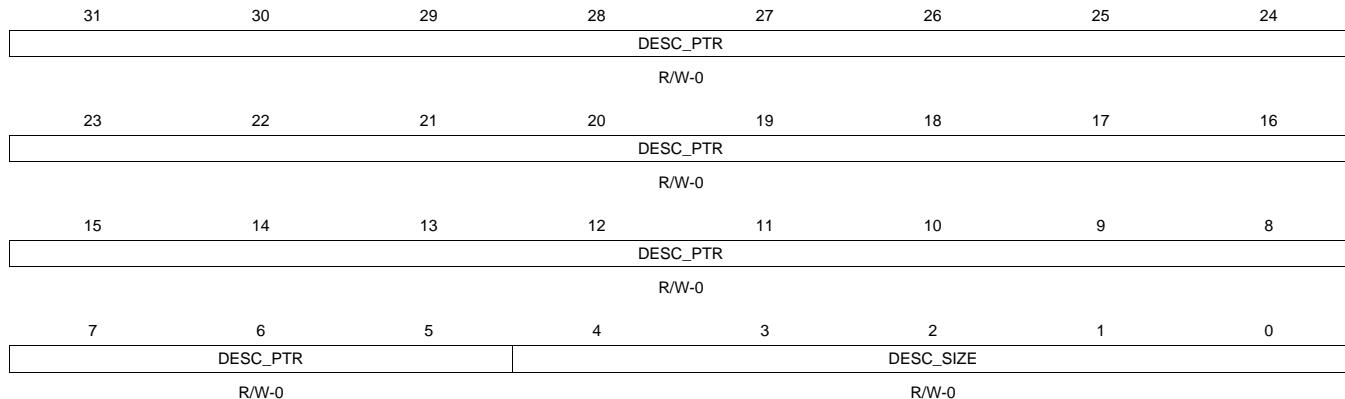
**Table 16-667. QUEUE\_85\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.378 QUEUE\_85\_D Register (offset = 255Ch) [reset = 0h]

QUEUE\_85\_D is shown in [Figure 16-654](#) and described in [Table 16-668](#).

**Figure 16-654. QUEUE\_85\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-668. QUEUE\_85\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.379 QUEUE\_86\_A Register (offset = 2560h) [reset = 0h]**

QUEUE\_86\_A is shown in [Figure 16-655](#) and described in [Table 16-669](#).

**Figure 16-655. QUEUE\_86\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-669. QUEUE\_86\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.380 QUEUE\_86\_B Register (offset = 2564h) [reset = 0h]**

QUEUE\_86\_B is shown in [Figure 16-656](#) and described in [Table 16-670](#).

**Figure 16-656. QUEUE\_86\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

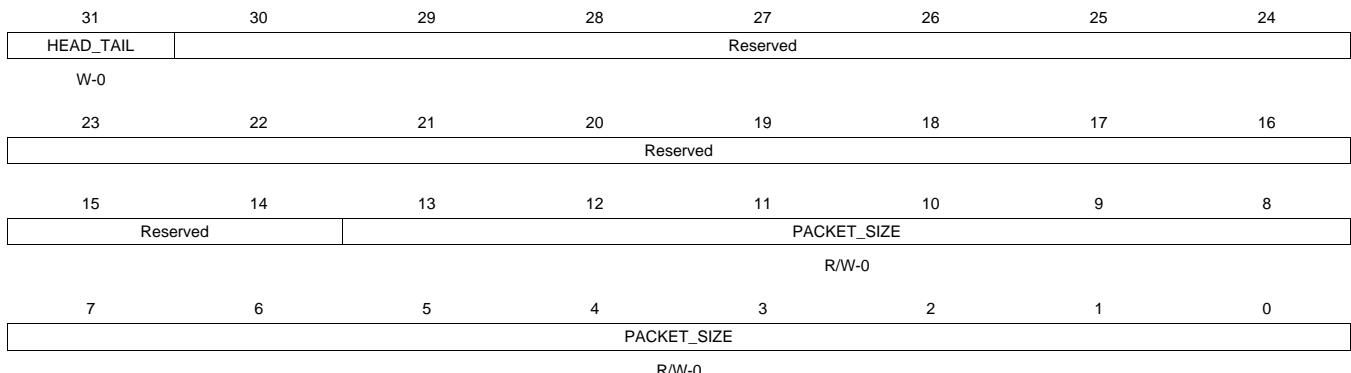
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-670. QUEUE\_86\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.381 QUEUE\_86\_C Register (offset = 2568h) [reset = 0h]**

QUEUE\_86\_C is shown in [Figure 16-657](#) and described in [Table 16-671](#).

**Figure 16-657. QUEUE\_86\_C Register**


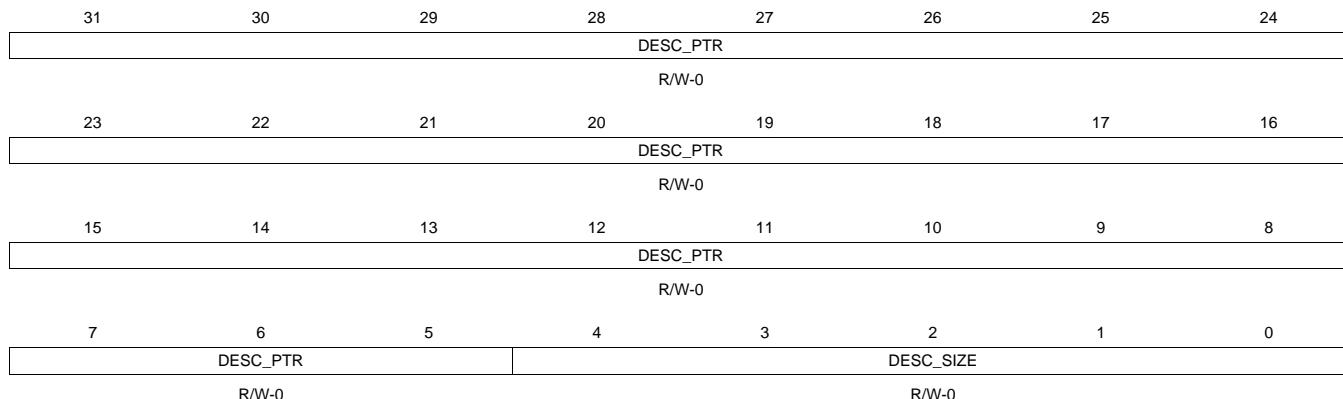
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-671. QUEUE\_86\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.382 QUEUE\_86\_D Register (offset = 256Ch) [reset = 0h]**

QUEUE\_86\_D is shown in [Figure 16-658](#) and described in [Table 16-672](#).

**Figure 16-658. QUEUE\_86\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-672. QUEUE\_86\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.383 QUEUE\_87\_A Register (offset = 2570h) [reset = 0h]**

QUEUE\_87\_A is shown in [Figure 16-659](#) and described in [Table 16-673](#).

**Figure 16-659. QUEUE\_87\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-673. QUEUE\_87\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.384 QUEUE\_87\_B Register (offset = 2574h) [reset = 0h]**

QUEUE\_87\_B is shown in [Figure 16-660](#) and described in [Table 16-674](#).

**Figure 16-660. QUEUE\_87\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

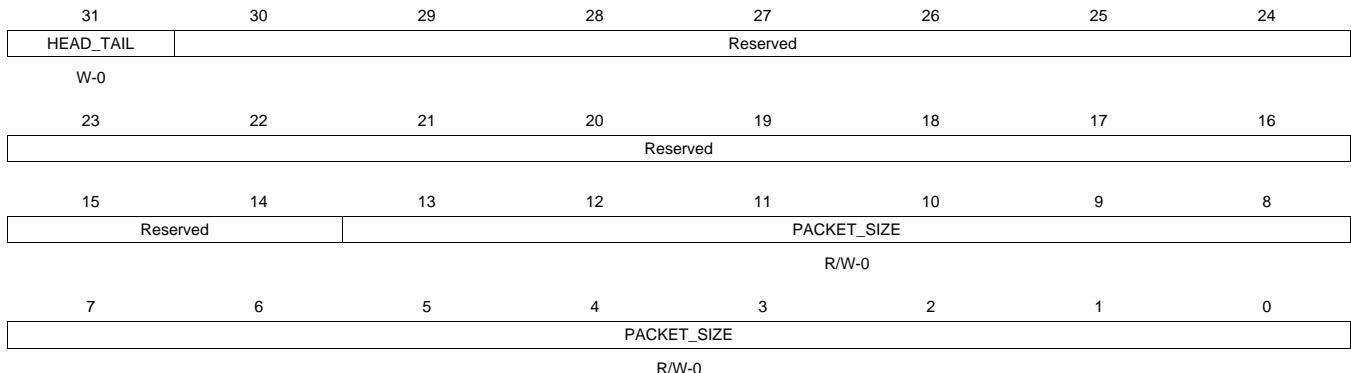
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-674. QUEUE\_87\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.385 QUEUE\_87\_C Register (offset = 2578h) [reset = 0h]**

QUEUE\_87\_C is shown in [Figure 16-661](#) and described in [Table 16-675](#).

**Figure 16-661. QUEUE\_87\_C Register**


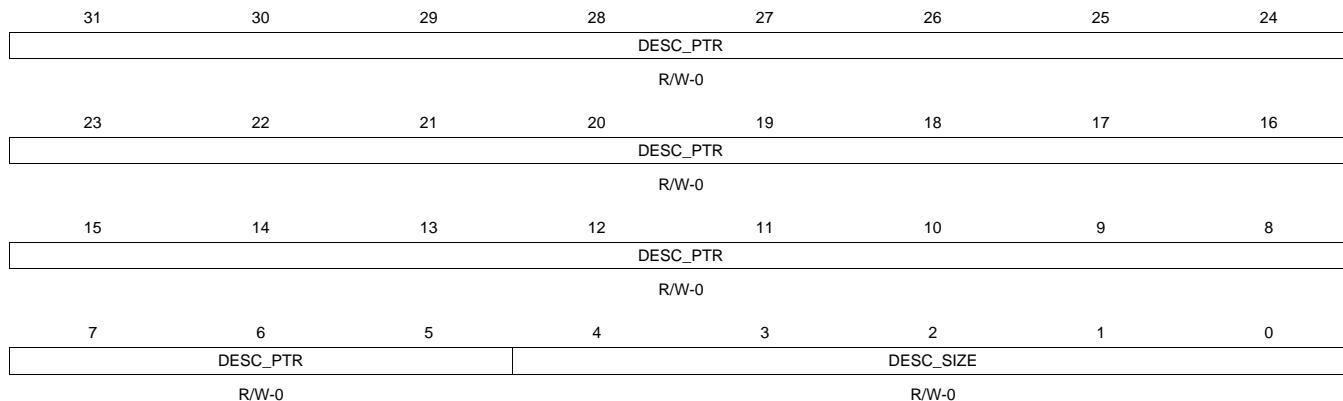
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-675. QUEUE\_87\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.386 QUEUE\_87\_D Register (offset = 257Ch) [reset = 0h]**

QUEUE\_87\_D is shown in [Figure 16-662](#) and described in [Table 16-676](#).

**Figure 16-662. QUEUE\_87\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-676. QUEUE\_87\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.387 QUEUE\_88\_A Register (offset = 2580h) [reset = 0h]**

QUEUE\_88\_A is shown in [Figure 16-663](#) and described in [Table 16-677](#).

**Figure 16-663. QUEUE\_88\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-677. QUEUE\_88\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.388 QUEUE\_88\_B Register (offset = 2584h) [reset = 0h]**

QUEUE\_88\_B is shown in [Figure 16-664](#) and described in [Table 16-678](#).

**Figure 16-664. QUEUE\_88\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

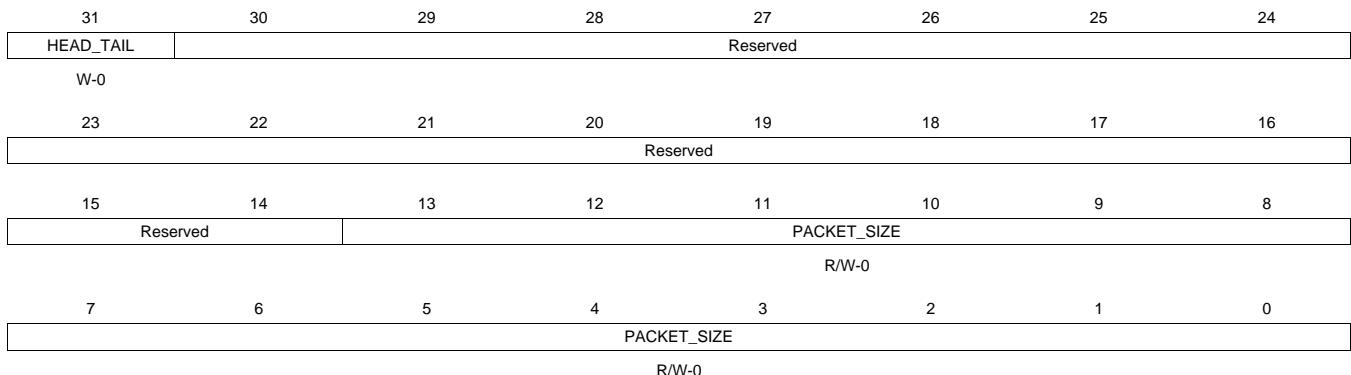
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-678. QUEUE\_88\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.389 QUEUE\_88\_C Register (offset = 2588h) [reset = 0h]**

QUEUE\_88\_C is shown in [Figure 16-665](#) and described in [Table 16-679](#).

**Figure 16-665. QUEUE\_88\_C Register**


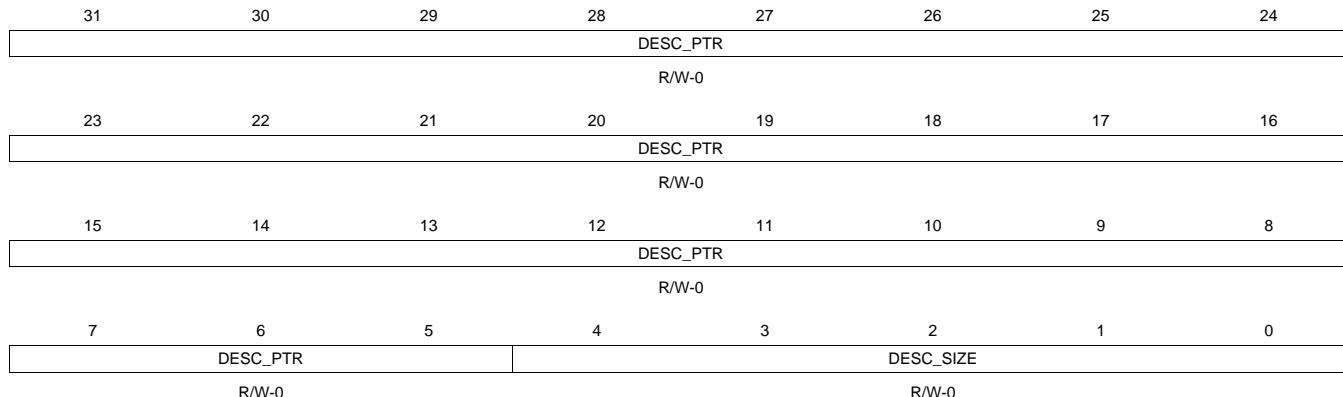
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-679. QUEUE\_88\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.390 QUEUE\_88\_D Register (offset = 258Ch) [reset = 0h]**

 QUEUE\_88\_D is shown in [Figure 16-666](#) and described in [Table 16-680](#).

**Figure 16-666. QUEUE\_88\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-680. QUEUE\_88\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.391 QUEUE\_89\_A Register (offset = 2590h) [reset = 0h]**

QUEUE\_89\_A is shown in [Figure 16-667](#) and described in [Table 16-681](#).

**Figure 16-667. QUEUE\_89\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-681. QUEUE\_89\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.392 QUEUE\_89\_B Register (offset = 2594h) [reset = 0h]**

QUEUE\_89\_B is shown in [Figure 16-668](#) and described in [Table 16-682](#).

**Figure 16-668. QUEUE\_89\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

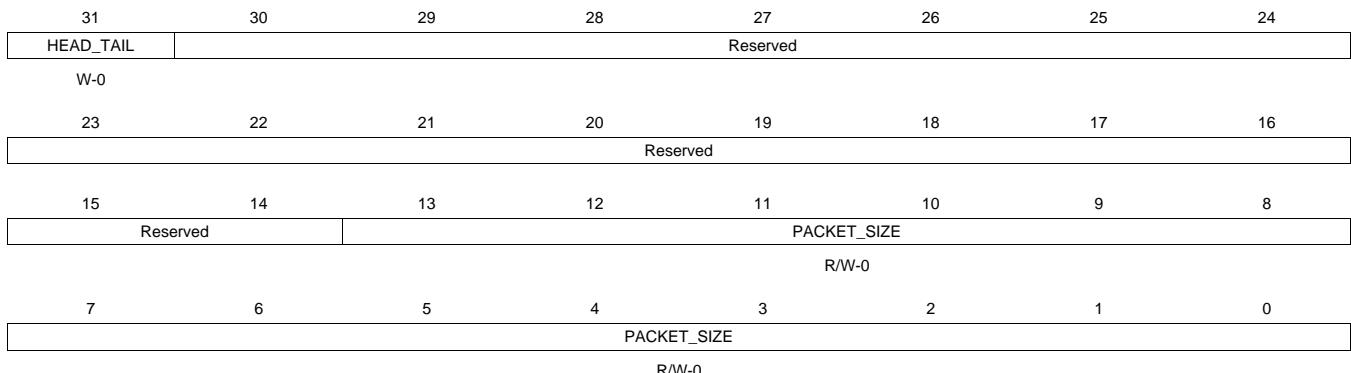
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-682. QUEUE\_89\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.393 QUEUE\_89\_C Register (offset = 2598h) [reset = 0h]**

QUEUE\_89\_C is shown in [Figure 16-669](#) and described in [Table 16-683](#).

**Figure 16-669. QUEUE\_89\_C Register**


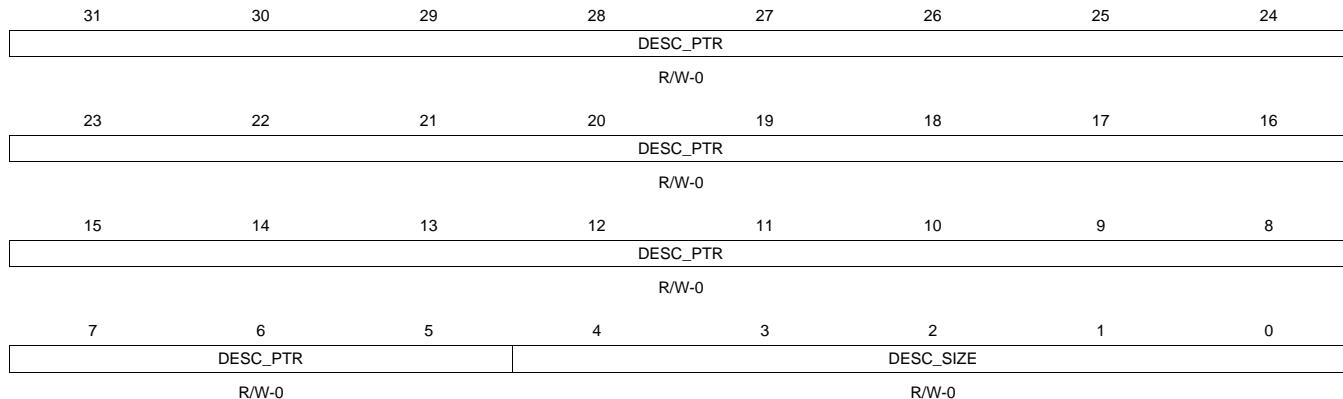
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-683. QUEUE\_89\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.394 QUEUE\_89\_D Register (offset = 259Ch) [reset = 0h]**

 QUEUE\_89\_D is shown in [Figure 16-670](#) and described in [Table 16-684](#).

**Figure 16-670. QUEUE\_89\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-684. QUEUE\_89\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.395 QUEUE\_90\_A Register (offset = 25A0h) [reset = 0h]**

QUEUE\_90\_A is shown in [Figure 16-671](#) and described in [Table 16-685](#).

**Figure 16-671. QUEUE\_90\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-685. QUEUE\_90\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.396 QUEUE\_90\_B Register (offset = 25A4h) [reset = 0h]**

QUEUE\_90\_B is shown in [Figure 16-672](#) and described in [Table 16-686](#).

**Figure 16-672. QUEUE\_90\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

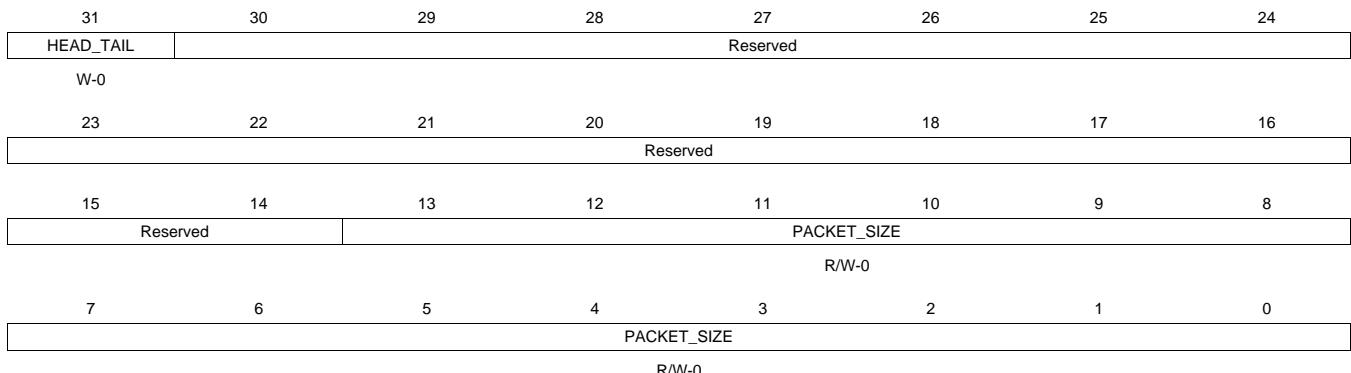
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-686. QUEUE\_90\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.397 QUEUE\_90\_C Register (offset = 25A8h) [reset = 0h]**

QUEUE\_90\_C is shown in [Figure 16-673](#) and described in [Table 16-687](#).

**Figure 16-673. QUEUE\_90\_C Register**


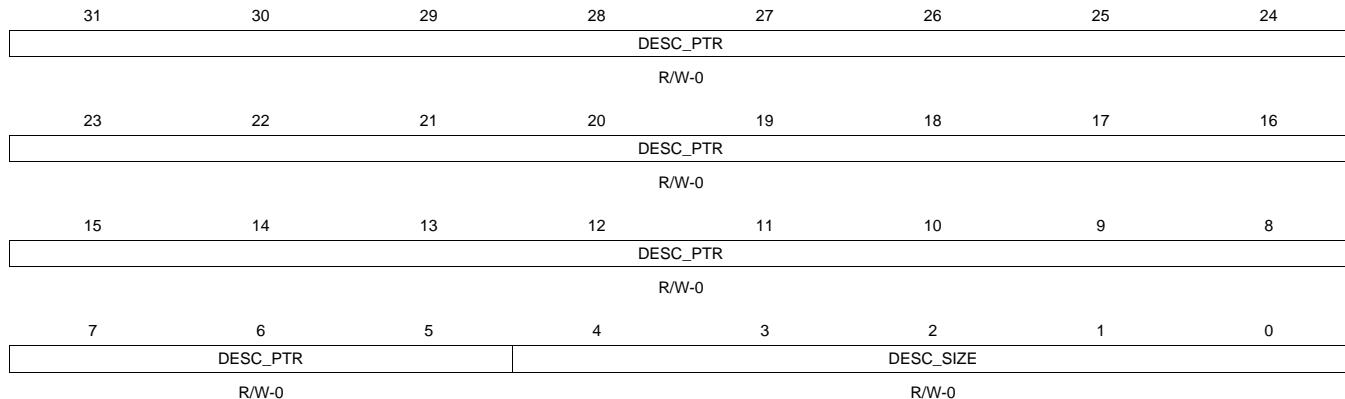
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-687. QUEUE\_90\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.398 QUEUE\_90\_D Register (offset = 25ACh) [reset = 0h]**

 QUEUE\_90\_D is shown in [Figure 16-674](#) and described in [Table 16-688](#).

**Figure 16-674. QUEUE\_90\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-688. QUEUE\_90\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.399 QUEUE\_91\_A Register (offset = 25B0h) [reset = 0h]**

QUEUE\_91\_A is shown in [Figure 16-675](#) and described in [Table 16-689](#).

**Figure 16-675. QUEUE\_91\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-689. QUEUE\_91\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.400 QUEUE\_91\_B Register (offset = 25B4h) [reset = 0h]**

QUEUE\_91\_B is shown in [Figure 16-676](#) and described in [Table 16-690](#).

**Figure 16-676. QUEUE\_91\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

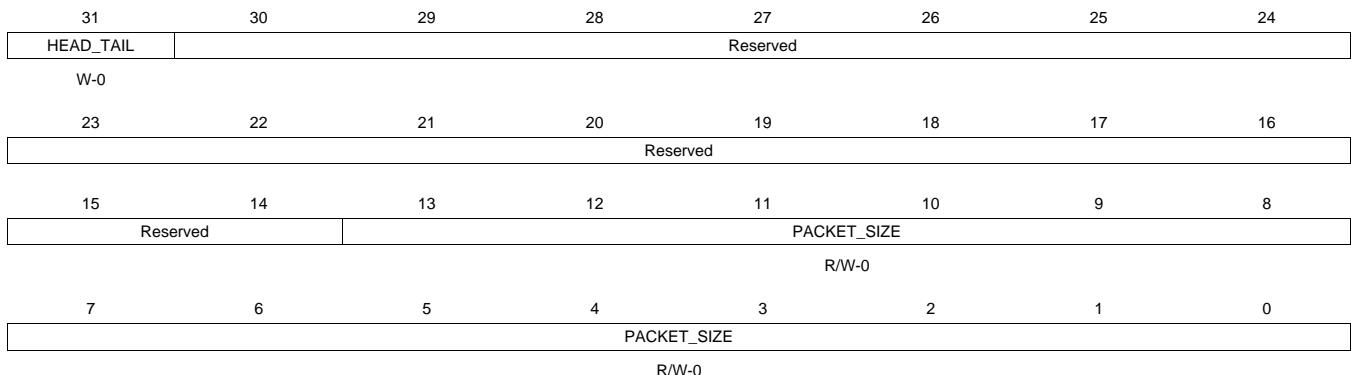
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-690. QUEUE\_91\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.401 QUEUE\_91\_C Register (offset = 25B8h) [reset = 0h]**

QUEUE\_91\_C is shown in [Figure 16-677](#) and described in [Table 16-691](#).

**Figure 16-677. QUEUE\_91\_C Register**


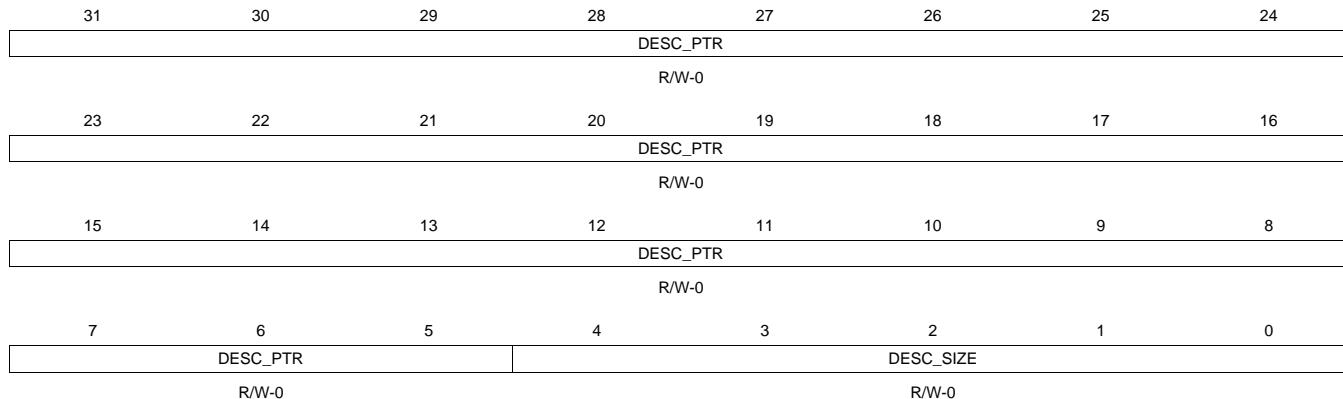
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-691. QUEUE\_91\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.402 QUEUE\_91\_D Register (offset = 25BCh) [reset = 0h]**

 QUEUE\_91\_D is shown in [Figure 16-678](#) and described in [Table 16-692](#).

**Figure 16-678. QUEUE\_91\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-692. QUEUE\_91\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.403 QUEUE\_92\_A Register (offset = 25C0h) [reset = 0h]**

QUEUE\_92\_A is shown in [Figure 16-679](#) and described in [Table 16-693](#).

**Figure 16-679. QUEUE\_92\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-693. QUEUE\_92\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.404 QUEUE\_92\_B Register (offset = 25C4h) [reset = 0h]**

QUEUE\_92\_B is shown in [Figure 16-680](#) and described in [Table 16-694](#).

**Figure 16-680. QUEUE\_92\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

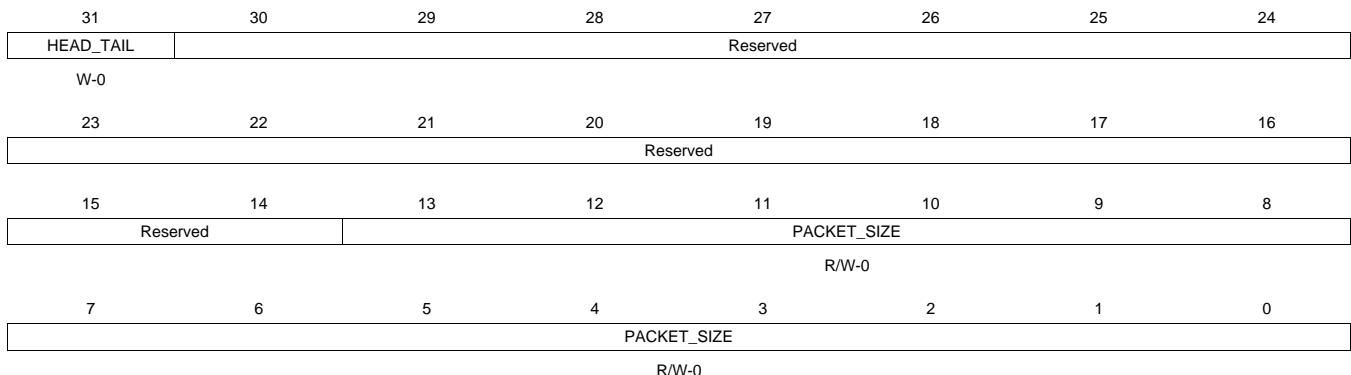
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-694. QUEUE\_92\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.405 QUEUE\_92\_C Register (offset = 25C8h) [reset = 0h]**

QUEUE\_92\_C is shown in [Figure 16-681](#) and described in [Table 16-695](#).

**Figure 16-681. QUEUE\_92\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

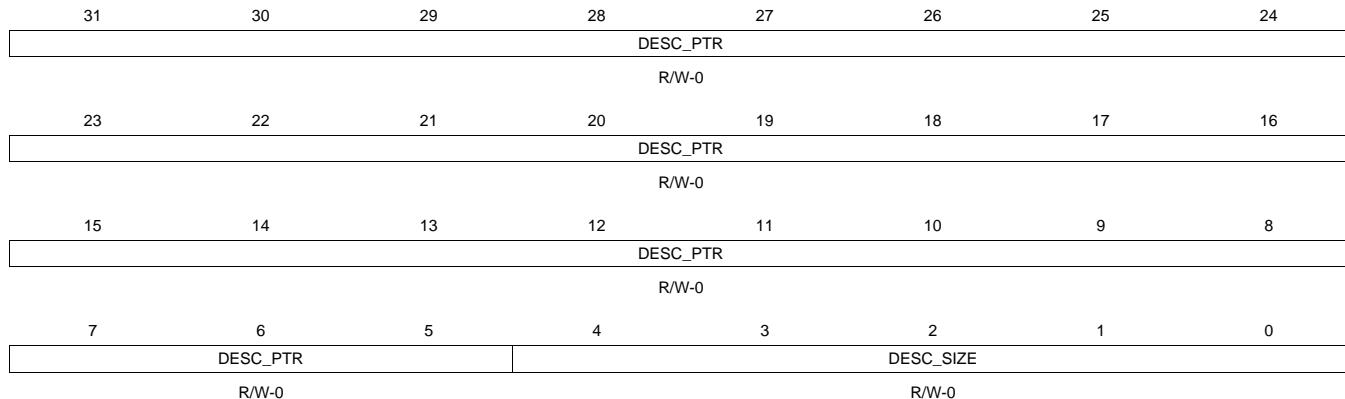
**Table 16-695. QUEUE\_92\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.406 QUEUE\_92\_D Register (offset = 25CCh) [reset = 0h]

QUEUE\_92\_D is shown in [Figure 16-682](#) and described in [Table 16-696](#).

**Figure 16-682. QUEUE\_92\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-696. QUEUE\_92\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.407 QUEUE\_93\_A Register (offset = 25D0h) [reset = 0h]**

QUEUE\_93\_A is shown in [Figure 16-683](#) and described in [Table 16-697](#).

**Figure 16-683. QUEUE\_93\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-697. QUEUE\_93\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.408 QUEUE\_93\_B Register (offset = 25D4h) [reset = 0h]**

QUEUE\_93\_B is shown in [Figure 16-684](#) and described in [Table 16-698](#).

**Figure 16-684. QUEUE\_93\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

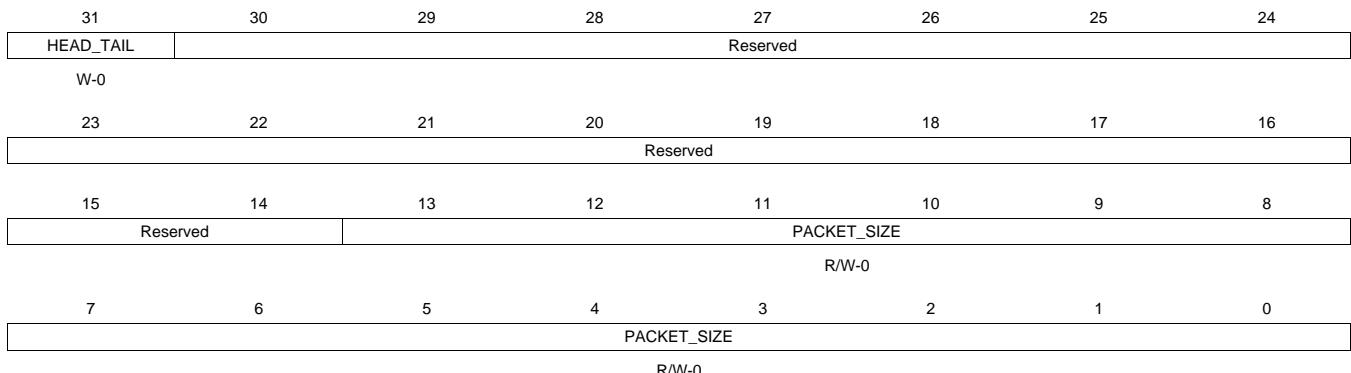
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-698. QUEUE\_93\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.409 QUEUE\_93\_C Register (offset = 25D8h) [reset = 0h]**

QUEUE\_93\_C is shown in [Figure 16-685](#) and described in [Table 16-699](#).

**Figure 16-685. QUEUE\_93\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

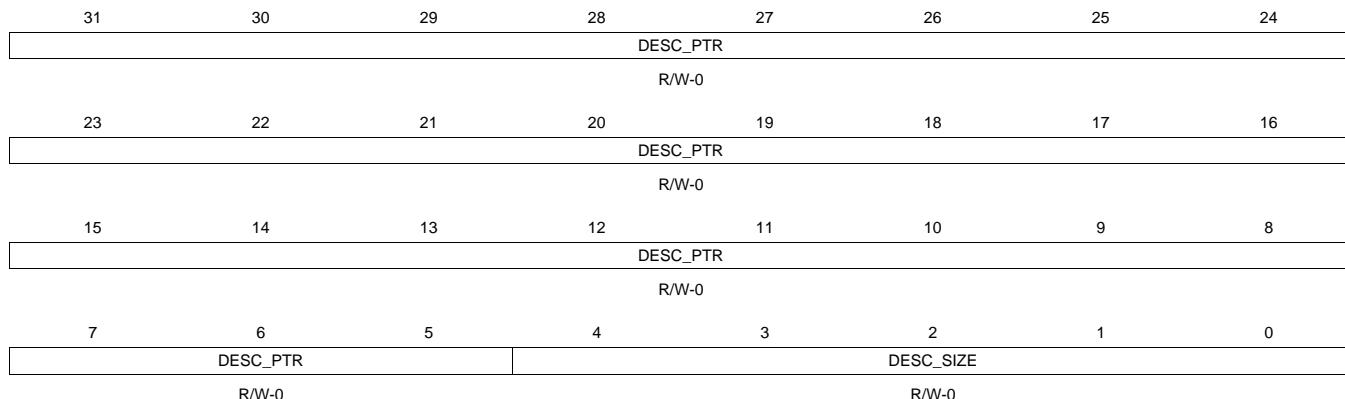
**Table 16-699. QUEUE\_93\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.410 QUEUE\_93\_D Register (offset = 25DCh) [reset = 0h]

QUEUE\_93\_D is shown in [Figure 16-686](#) and described in [Table 16-700](#).

**Figure 16-686. QUEUE\_93\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-700. QUEUE\_93\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.411 QUEUE\_94\_A Register (offset = 25E0h) [reset = 0h]**

QUEUE\_94\_A is shown in [Figure 16-687](#) and described in [Table 16-701](#).

**Figure 16-687. QUEUE\_94\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-701. QUEUE\_94\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.412 QUEUE\_94\_B Register (offset = 25E4h) [reset = 0h]**

QUEUE\_94\_B is shown in [Figure 16-688](#) and described in [Table 16-702](#).

**Figure 16-688. QUEUE\_94\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

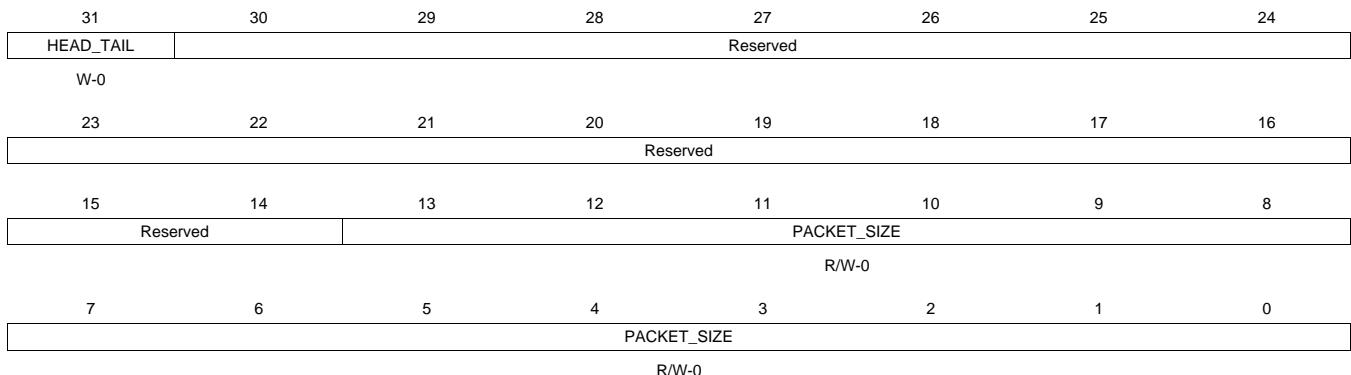
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-702. QUEUE\_94\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.413 QUEUE\_94\_C Register (offset = 25E8h) [reset = 0h]**

QUEUE\_94\_C is shown in [Figure 16-689](#) and described in [Table 16-703](#).

**Figure 16-689. QUEUE\_94\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

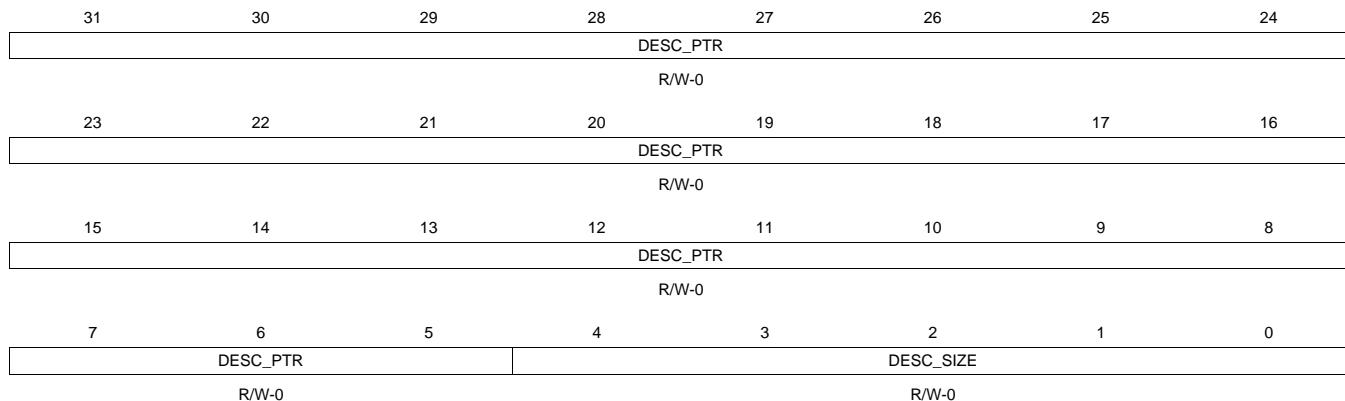
**Table 16-703. QUEUE\_94\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.414 QUEUE\_94\_D Register (offset = 25ECh) [reset = 0h]

QUEUE\_94\_D is shown in [Figure 16-690](#) and described in [Table 16-704](#).

**Figure 16-690. QUEUE\_94\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-704. QUEUE\_94\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.415 QUEUE\_95\_A Register (offset = 25F0h) [reset = 0h]

QUEUE\_95\_A is shown in [Figure 16-691](#) and described in [Table 16-705](#).

**Figure 16-691. QUEUE\_95\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-705. QUEUE\_95\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.416 QUEUE\_95\_B Register (offset = 25F4h) [reset = 0h]**

QUEUE\_95\_B is shown in [Figure 16-692](#) and described in [Table 16-706](#).

**Figure 16-692. QUEUE\_95\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

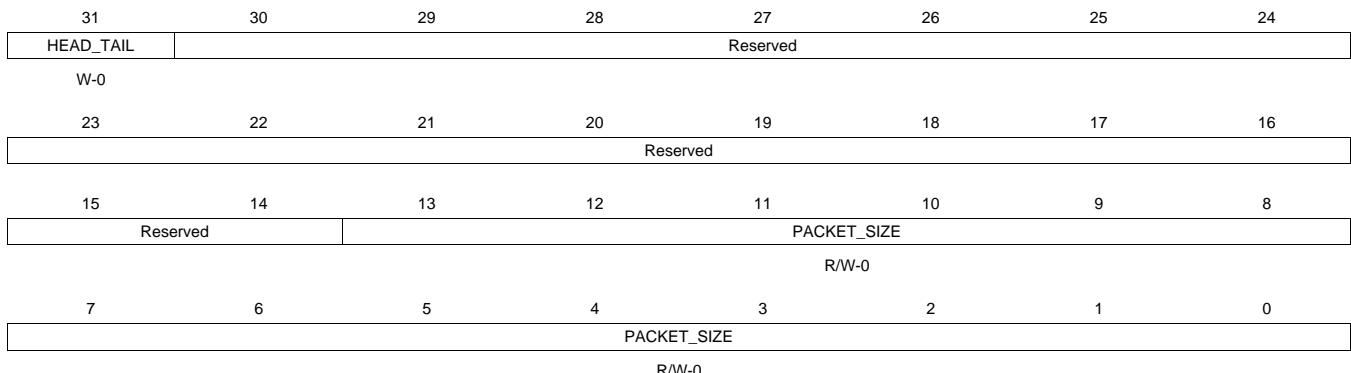
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-706. QUEUE\_95\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.417 QUEUE\_95\_C Register (offset = 25F8h) [reset = 0h]**

QUEUE\_95\_C is shown in [Figure 16-693](#) and described in [Table 16-707](#).

**Figure 16-693. QUEUE\_95\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

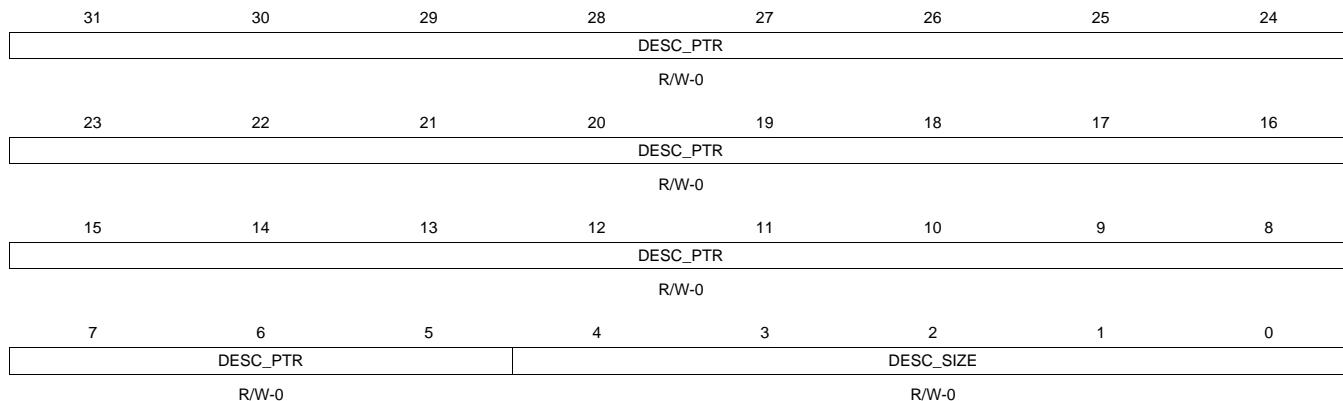
**Table 16-707. QUEUE\_95\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.418 QUEUE\_95\_D Register (offset = 25FCh) [reset = 0h]

QUEUE\_95\_D is shown in [Figure 16-694](#) and described in [Table 16-708](#).

**Figure 16-694. QUEUE\_95\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-708. QUEUE\_95\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.419 QUEUE\_96\_A Register (offset = 2600h) [reset = 0h]**

QUEUE\_96\_A is shown in [Figure 16-695](#) and described in [Table 16-709](#).

**Figure 16-695. QUEUE\_96\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-709. QUEUE\_96\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.420 QUEUE\_96\_B Register (offset = 2604h) [reset = 0h]**

QUEUE\_96\_B is shown in [Figure 16-696](#) and described in [Table 16-710](#).

**Figure 16-696. QUEUE\_96\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

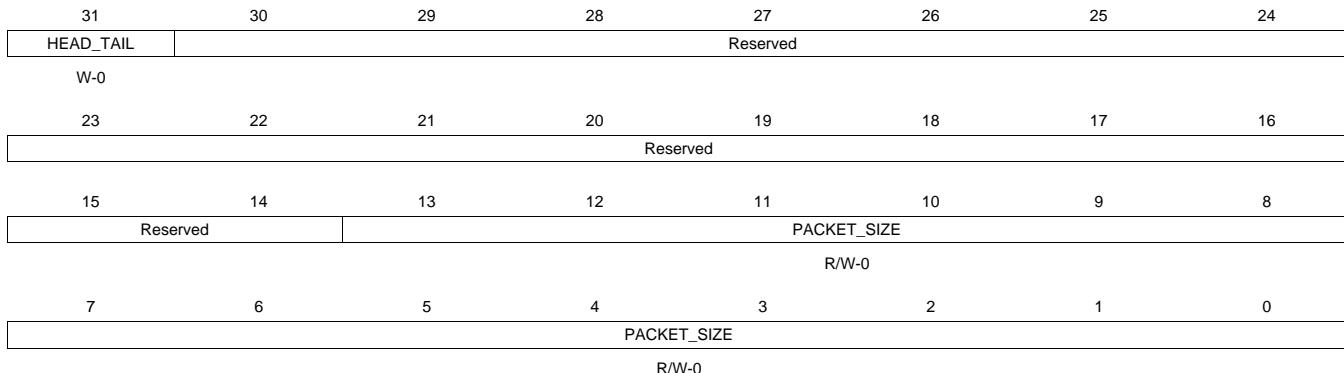
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-710. QUEUE\_96\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.421 QUEUE\_96\_C Register (offset = 2608h) [reset = 0h]**

QUEUE\_96\_C is shown in [Figure 16-697](#) and described in [Table 16-711](#).

**Figure 16-697. QUEUE\_96\_C Register**


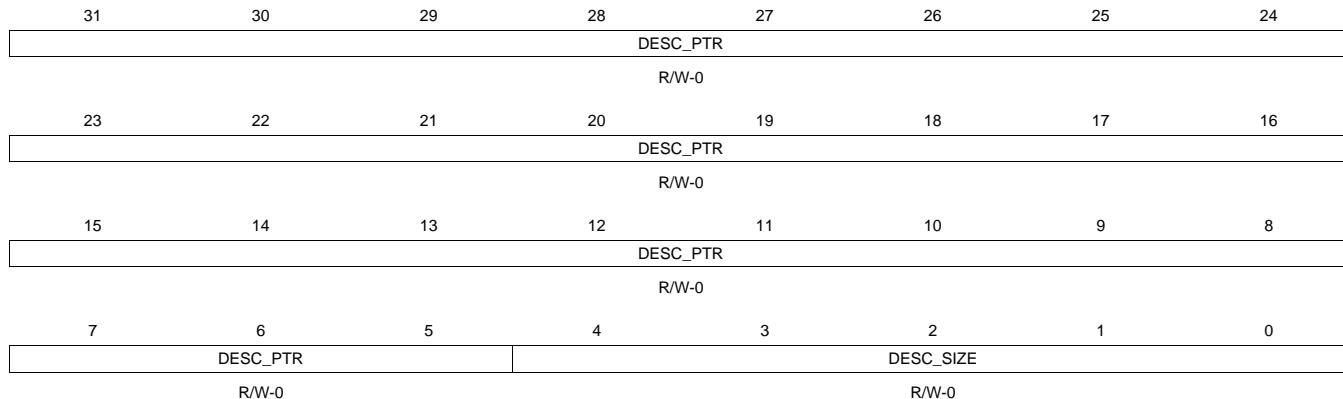
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-711. QUEUE\_96\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.422 QUEUE\_96\_D Register (offset = 260Ch) [reset = 0h]**

 QUEUE\_96\_D is shown in [Figure 16-698](#) and described in [Table 16-712](#).

**Figure 16-698. QUEUE\_96\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-712. QUEUE\_96\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.423 QUEUE\_97\_A Register (offset = 2610h) [reset = 0h]**

QUEUE\_97\_A is shown in [Figure 16-699](#) and described in [Table 16-713](#).

**Figure 16-699. QUEUE\_97\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-713. QUEUE\_97\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.424 QUEUE\_97\_B Register (offset = 2614h) [reset = 0h]**

QUEUE\_97\_B is shown in [Figure 16-700](#) and described in [Table 16-714](#).

**Figure 16-700. QUEUE\_97\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

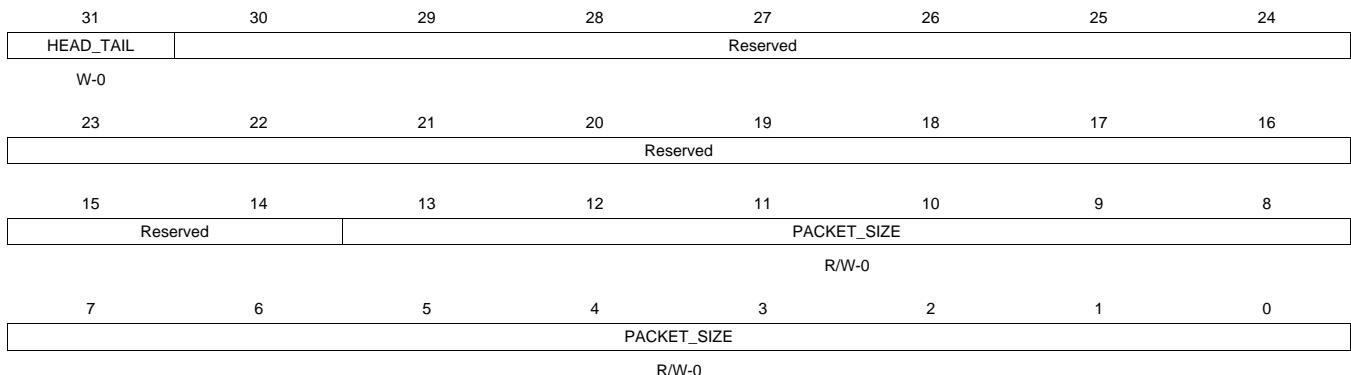
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-714. QUEUE\_97\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.425 QUEUE\_97\_C Register (offset = 2618h) [reset = 0h]**

QUEUE\_97\_C is shown in [Figure 16-701](#) and described in [Table 16-715](#).

**Figure 16-701. QUEUE\_97\_C Register**


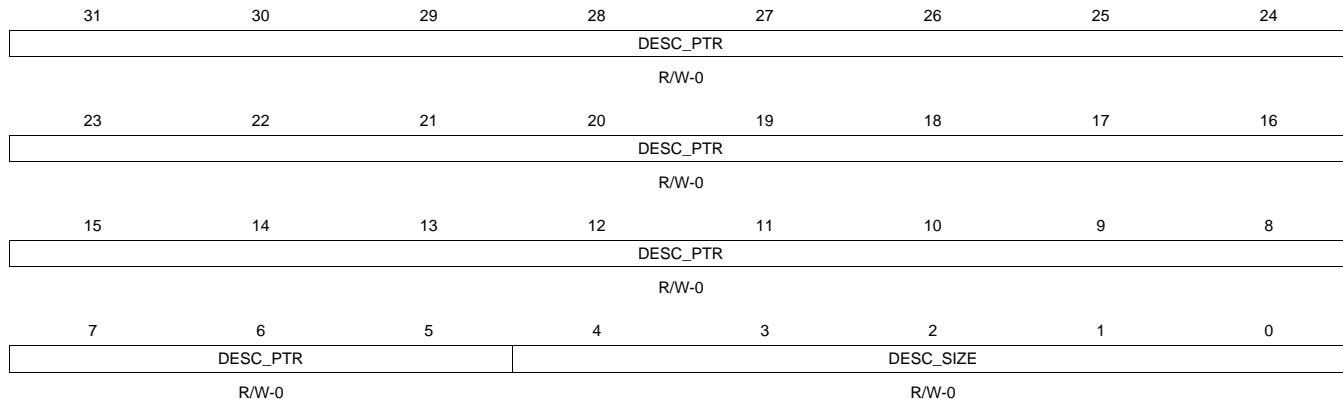
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-715. QUEUE\_97\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.426 QUEUE\_97\_D Register (offset = 261Ch) [reset = 0h]**

 QUEUE\_97\_D is shown in [Figure 16-702](#) and described in [Table 16-716](#).

**Figure 16-702. QUEUE\_97\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-716. QUEUE\_97\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.427 QUEUE\_98\_A Register (offset = 2620h) [reset = 0h]**

QUEUE\_98\_A is shown in [Figure 16-703](#) and described in [Table 16-717](#).

**Figure 16-703. QUEUE\_98\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-717. QUEUE\_98\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.428 QUEUE\_98\_B Register (offset = 2624h) [reset = 0h]**

QUEUE\_98\_B is shown in [Figure 16-704](#) and described in [Table 16-718](#).

**Figure 16-704. QUEUE\_98\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

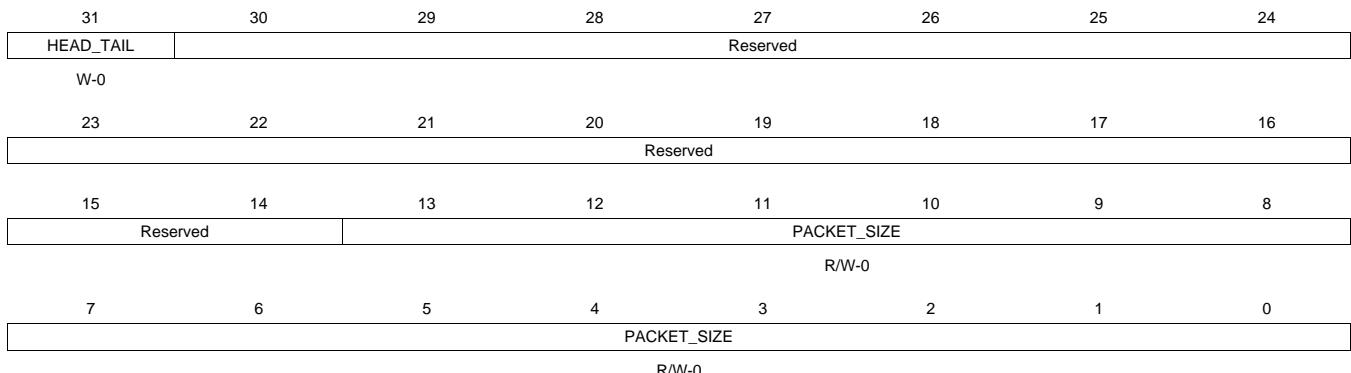
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-718. QUEUE\_98\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.429 QUEUE\_98\_C Register (offset = 2628h) [reset = 0h]**

QUEUE\_98\_C is shown in [Figure 16-705](#) and described in [Table 16-719](#).

**Figure 16-705. QUEUE\_98\_C Register**


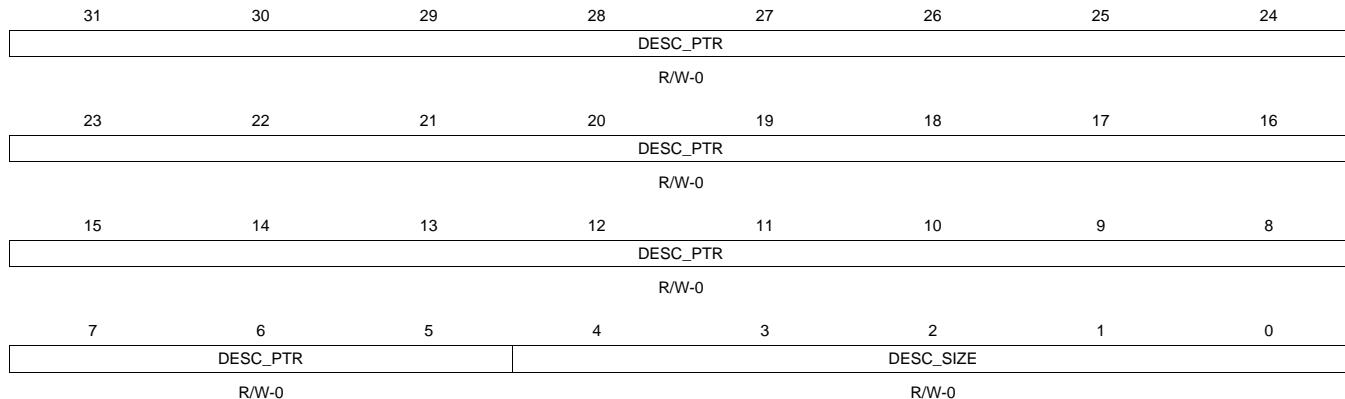
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-719. QUEUE\_98\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.430 QUEUE\_98\_D Register (offset = 262Ch) [reset = 0h]**

 QUEUE\_98\_D is shown in [Figure 16-706](#) and described in [Table 16-720](#).

**Figure 16-706. QUEUE\_98\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-720. QUEUE\_98\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.431 QUEUE\_99\_A Register (offset = 2630h) [reset = 0h]**

QUEUE\_99\_A is shown in [Figure 16-707](#) and described in [Table 16-721](#).

**Figure 16-707. QUEUE\_99\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-721. QUEUE\_99\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.432 QUEUE\_99\_B Register (offset = 2634h) [reset = 0h]**

QUEUE\_99\_B is shown in [Figure 16-708](#) and described in [Table 16-722](#).

**Figure 16-708. QUEUE\_99\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

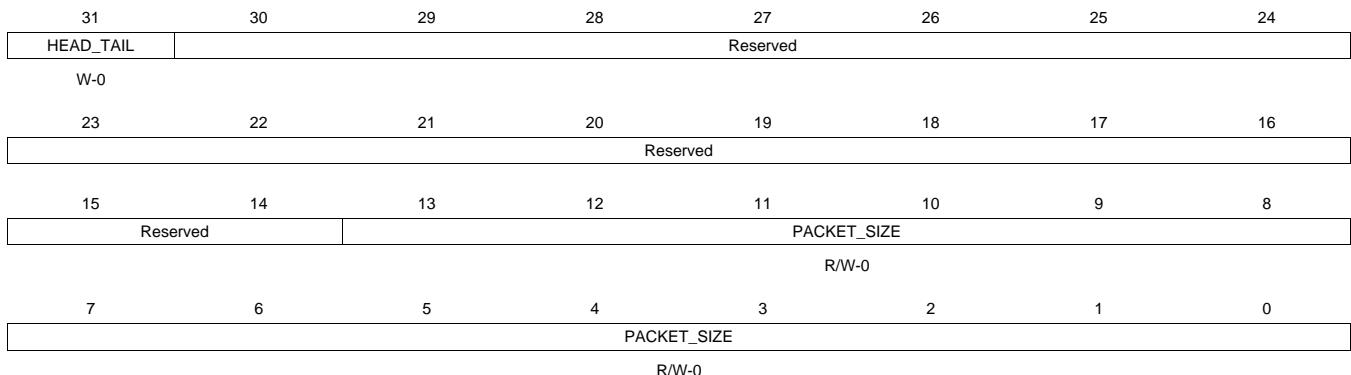
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-722. QUEUE\_99\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.433 QUEUE\_99\_C Register (offset = 2638h) [reset = 0h]**

QUEUE\_99\_C is shown in [Figure 16-709](#) and described in [Table 16-723](#).

**Figure 16-709. QUEUE\_99\_C Register**


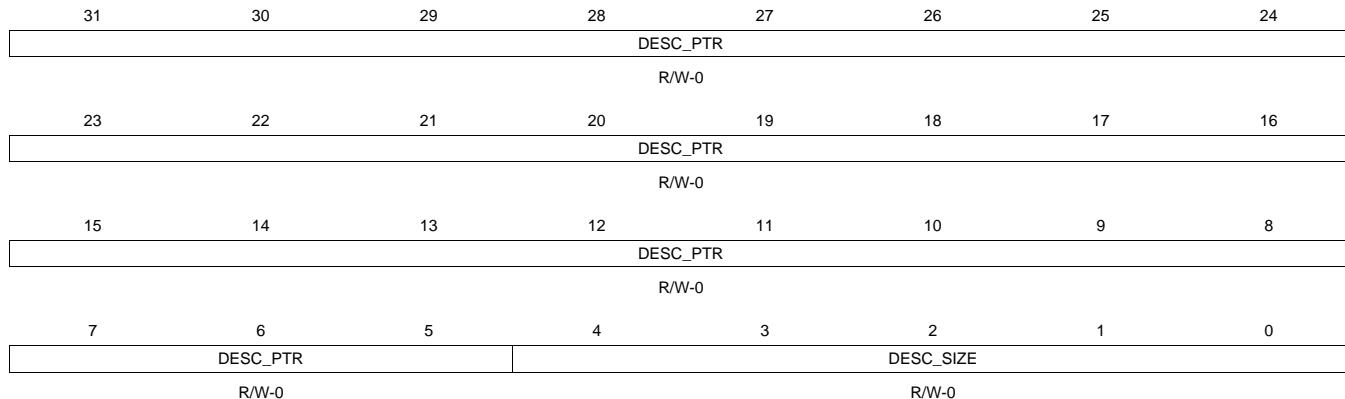
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-723. QUEUE\_99\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.434 QUEUE\_99\_D Register (offset = 263Ch) [reset = 0h]**

 QUEUE\_99\_D is shown in [Figure 16-710](#) and described in [Table 16-724](#).

**Figure 16-710. QUEUE\_99\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-724. QUEUE\_99\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.435 QUEUE\_100\_A Register (offset = 2640h) [reset = 0h]**

QUEUE\_100\_A is shown in [Figure 16-711](#) and described in [Table 16-725](#).

**Figure 16-711. QUEUE\_100\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-725. QUEUE\_100\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.436 QUEUE\_100\_B Register (offset = 2644h) [reset = 0h]**

QUEUE\_100\_B is shown in [Figure 16-712](#) and described in [Table 16-726](#).

**Figure 16-712. QUEUE\_100\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

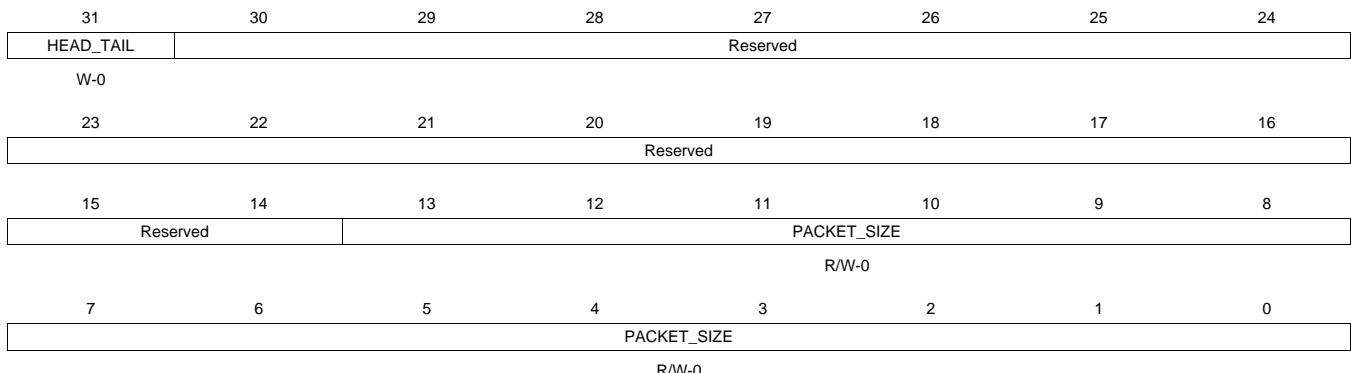
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-726. QUEUE\_100\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.437 QUEUE\_100\_C Register (offset = 2648h) [reset = 0h]**

QUEUE\_100\_C is shown in [Figure 16-713](#) and described in [Table 16-727](#).

**Figure 16-713. QUEUE\_100\_C Register**


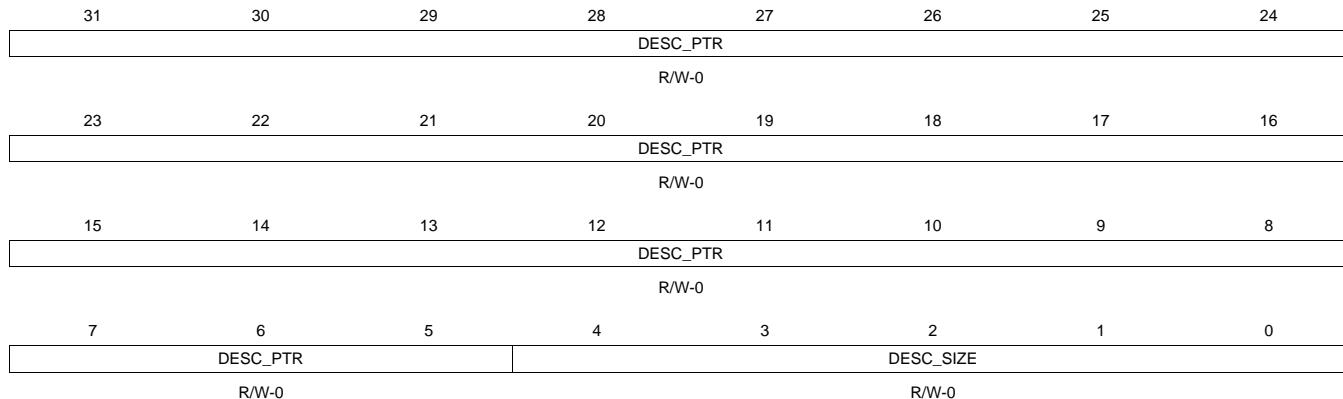
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-727. QUEUE\_100\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.438 QUEUE\_100\_D Register (offset = 264Ch) [reset = 0h]**

 QUEUE\_100\_D is shown in [Figure 16-714](#) and described in [Table 16-728](#).

**Figure 16-714. QUEUE\_100\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-728. QUEUE\_100\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.439 QUEUE\_101\_A Register (offset = 2650h) [reset = 0h]**

QUEUE\_101\_A is shown in [Figure 16-715](#) and described in [Table 16-729](#).

**Figure 16-715. QUEUE\_101\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-729. QUEUE\_101\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.440 QUEUE\_101\_B Register (offset = 2654h) [reset = 0h]**

QUEUE\_101\_B is shown in [Figure 16-716](#) and described in [Table 16-730](#).

**Figure 16-716. QUEUE\_101\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

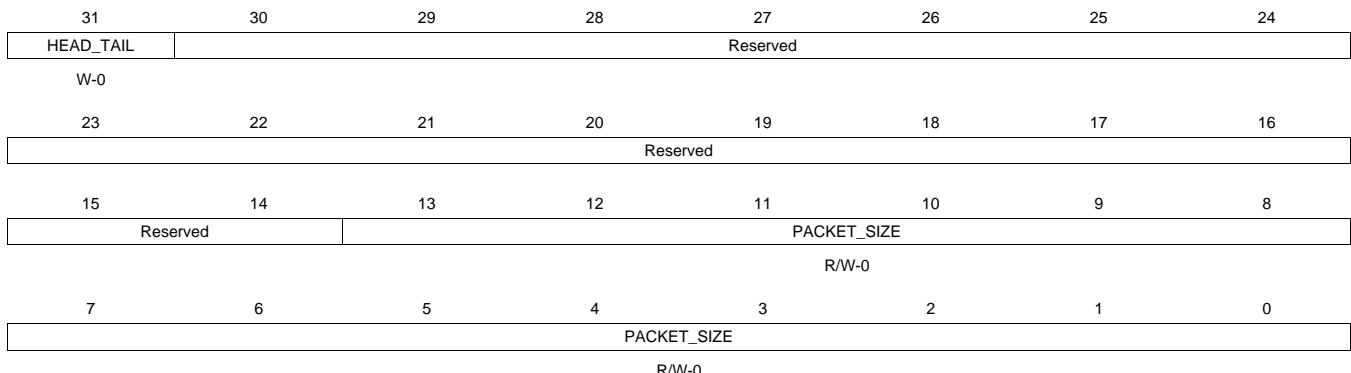
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-730. QUEUE\_101\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.441 QUEUE\_101\_C Register (offset = 2658h) [reset = 0h]**

QUEUE\_101\_C is shown in [Figure 16-717](#) and described in [Table 16-731](#).

**Figure 16-717. QUEUE\_101\_C Register**


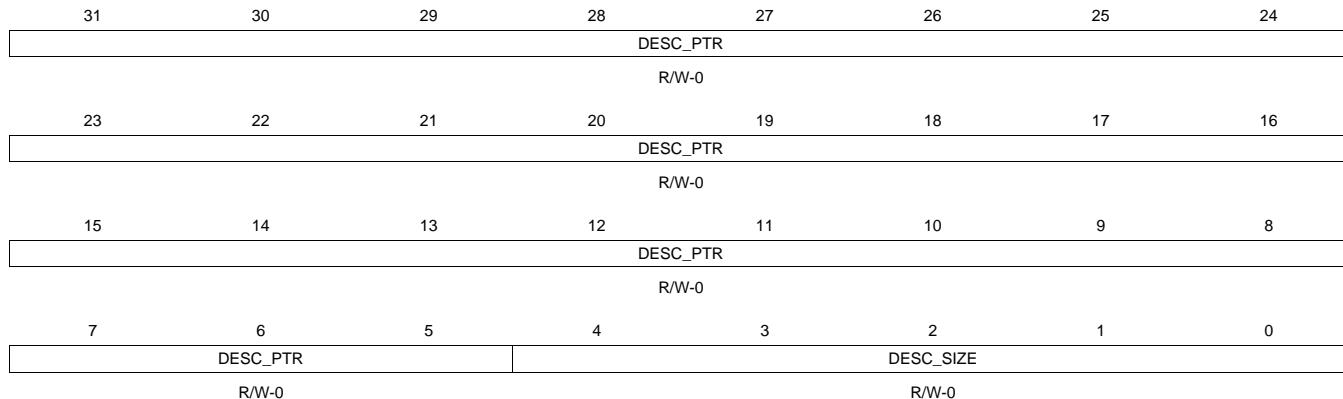
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-731. QUEUE\_101\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.442 QUEUE\_101\_D Register (offset = 265Ch) [reset = 0h]**

 QUEUE\_101\_D is shown in [Figure 16-718](#) and described in [Table 16-732](#).

**Figure 16-718. QUEUE\_101\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-732. QUEUE\_101\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.443 QUEUE\_102\_A Register (offset = 2660h) [reset = 0h]**

QUEUE\_102\_A is shown in [Figure 16-719](#) and described in [Table 16-733](#).

**Figure 16-719. QUEUE\_102\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-733. QUEUE\_102\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.444 QUEUE\_102\_B Register (offset = 2664h) [reset = 0h]**

QUEUE\_102\_B is shown in [Figure 16-720](#) and described in [Table 16-734](#).

**Figure 16-720. QUEUE\_102\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

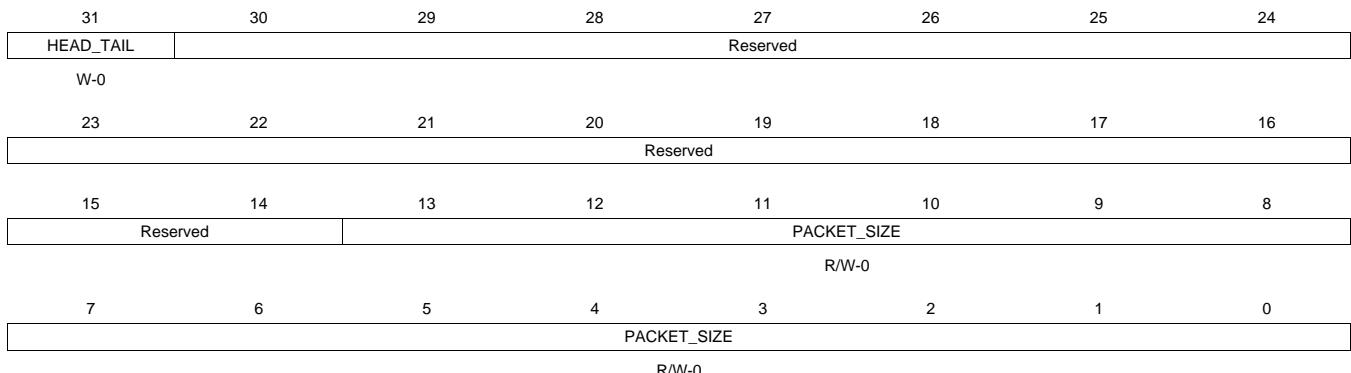
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-734. QUEUE\_102\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.445 QUEUE\_102\_C Register (offset = 2668h) [reset = 0h]**

QUEUE\_102\_C is shown in [Figure 16-721](#) and described in [Table 16-735](#).

**Figure 16-721. QUEUE\_102\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

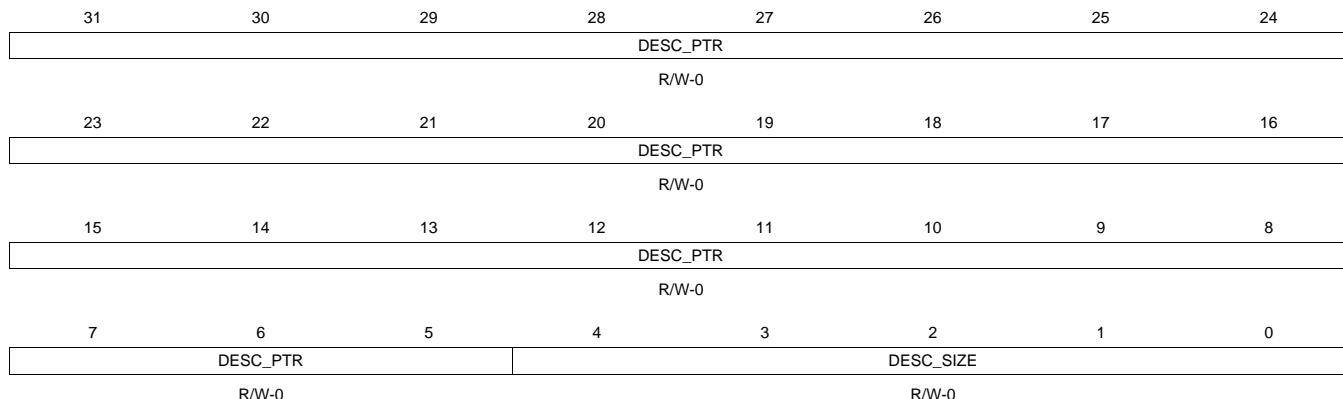
**Table 16-735. QUEUE\_102\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.446 QUEUE\_102\_D Register (offset = 266Ch) [reset = 0h]

QUEUE\_102\_D is shown in [Figure 16-722](#) and described in [Table 16-736](#).

**Figure 16-722. QUEUE\_102\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-736. QUEUE\_102\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.447 QUEUE\_103\_A Register (offset = 2670h) [reset = 0h]**

QUEUE\_103\_A is shown in [Figure 16-723](#) and described in [Table 16-737](#).

**Figure 16-723. QUEUE\_103\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-737. QUEUE\_103\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.448 QUEUE\_103\_B Register (offset = 2674h) [reset = 0h]**

QUEUE\_103\_B is shown in [Figure 16-724](#) and described in [Table 16-738](#).

**Figure 16-724. QUEUE\_103\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

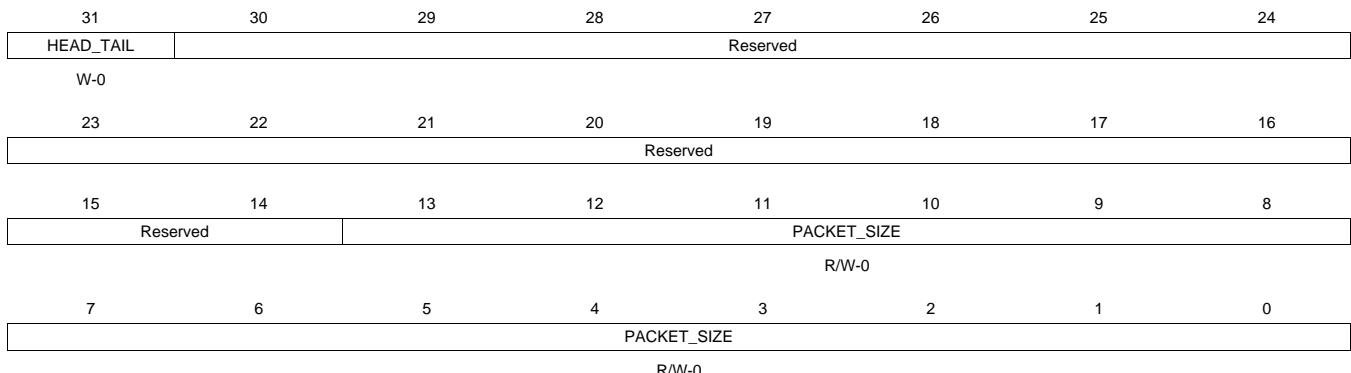
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-738. QUEUE\_103\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.449 QUEUE\_103\_C Register (offset = 2678h) [reset = 0h]**

QUEUE\_103\_C is shown in [Figure 16-725](#) and described in [Table 16-739](#).

**Figure 16-725. QUEUE\_103\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

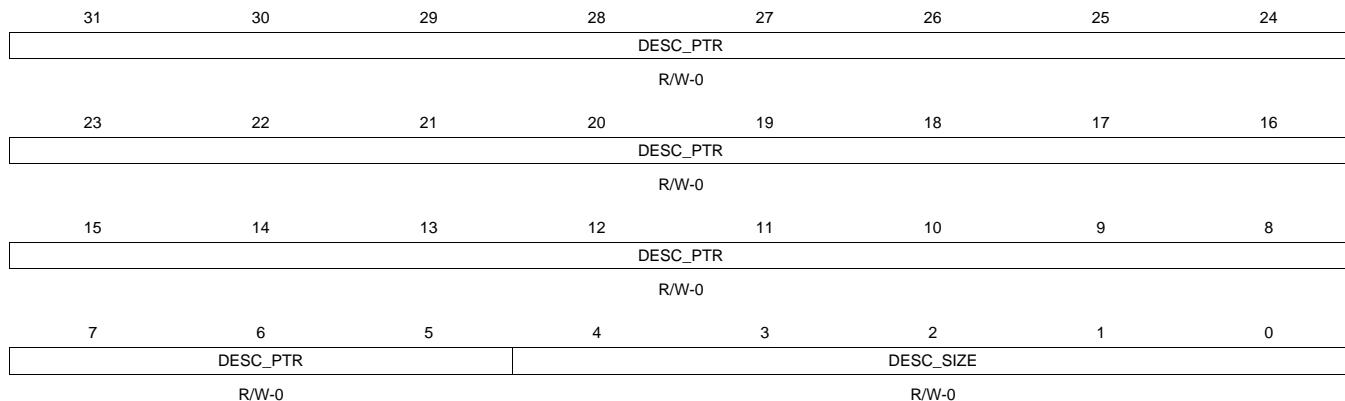
**Table 16-739. QUEUE\_103\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.450 QUEUE\_103\_D Register (offset = 267Ch) [reset = 0h]

QUEUE\_103\_D is shown in [Figure 16-726](#) and described in [Table 16-740](#).

**Figure 16-726. QUEUE\_103\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-740. QUEUE\_103\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.451 QUEUE\_104\_A Register (offset = 2680h) [reset = 0h]**

QUEUE\_104\_A is shown in [Figure 16-727](#) and described in [Table 16-741](#).

**Figure 16-727. QUEUE\_104\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-741. QUEUE\_104\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.452 QUEUE\_104\_B Register (offset = 2684h) [reset = 0h]**

QUEUE\_104\_B is shown in [Figure 16-728](#) and described in [Table 16-742](#).

**Figure 16-728. QUEUE\_104\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

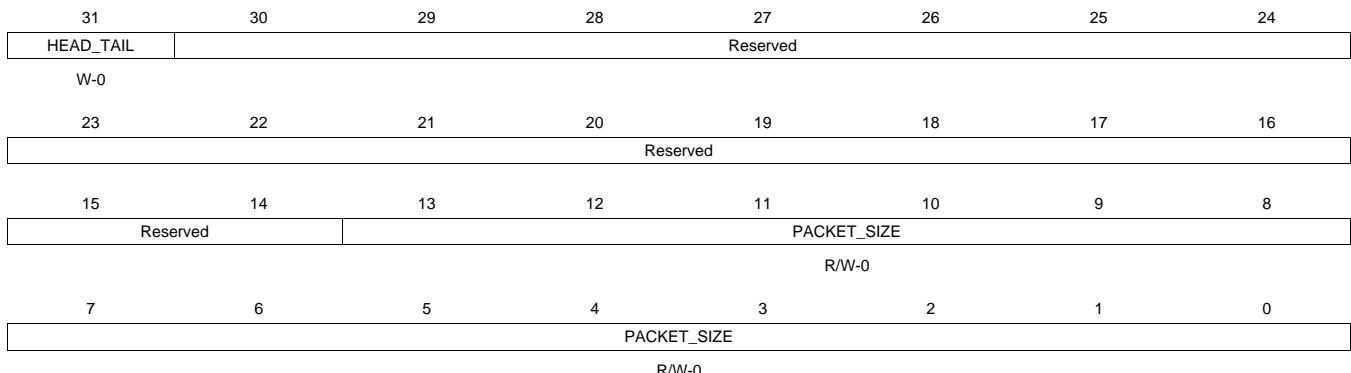
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-742. QUEUE\_104\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.453 QUEUE\_104\_C Register (offset = 2688h) [reset = 0h]**

QUEUE\_104\_C is shown in [Figure 16-729](#) and described in [Table 16-743](#).

**Figure 16-729. QUEUE\_104\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

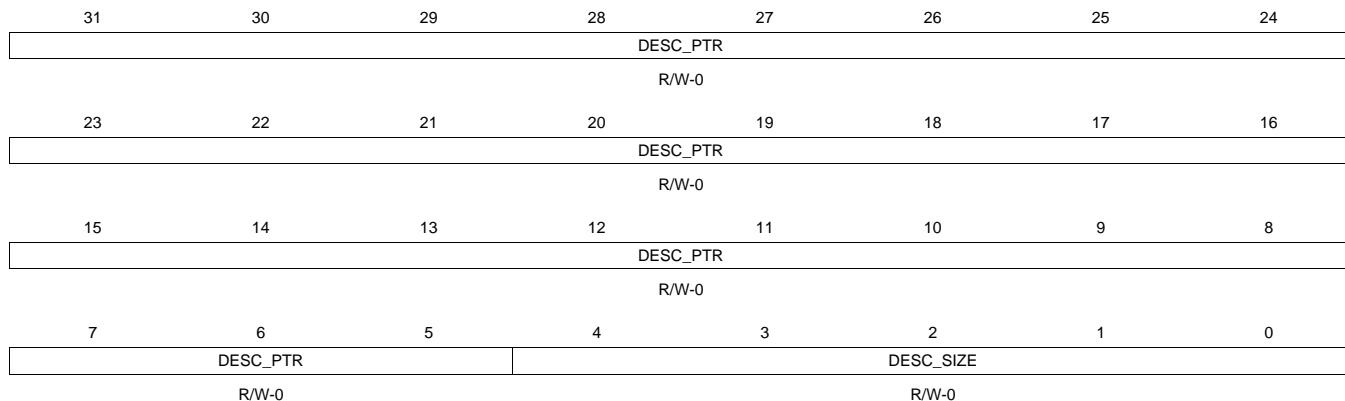
**Table 16-743. QUEUE\_104\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.454 QUEUE\_104\_D Register (offset = 268Ch) [reset = 0h]

QUEUE\_104\_D is shown in [Figure 16-730](#) and described in [Table 16-744](#).

**Figure 16-730. QUEUE\_104\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-744. QUEUE\_104\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.455 QUEUE\_105\_A Register (offset = 2690h) [reset = 0h]**

QUEUE\_105\_A is shown in [Figure 16-731](#) and described in [Table 16-745](#).

**Figure 16-731. QUEUE\_105\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-745. QUEUE\_105\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.456 QUEUE\_105\_B Register (offset = 2694h) [reset = 0h]**

QUEUE\_105\_B is shown in [Figure 16-732](#) and described in [Table 16-746](#).

**Figure 16-732. QUEUE\_105\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

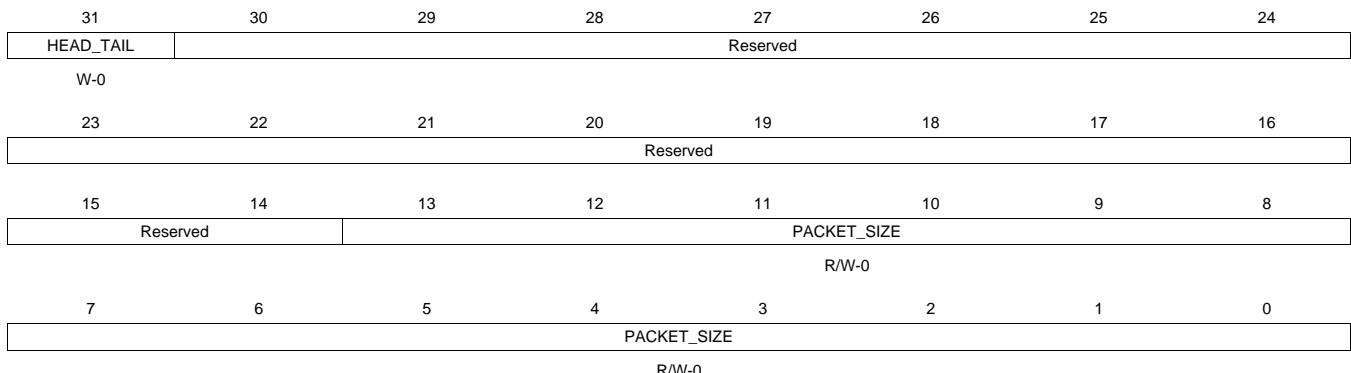
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-746. QUEUE\_105\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.457 QUEUE\_105\_C Register (offset = 2698h) [reset = 0h]**

 QUEUE\_105\_C is shown in [Figure 16-733](#) and described in [Table 16-747](#).

**Figure 16-733. QUEUE\_105\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

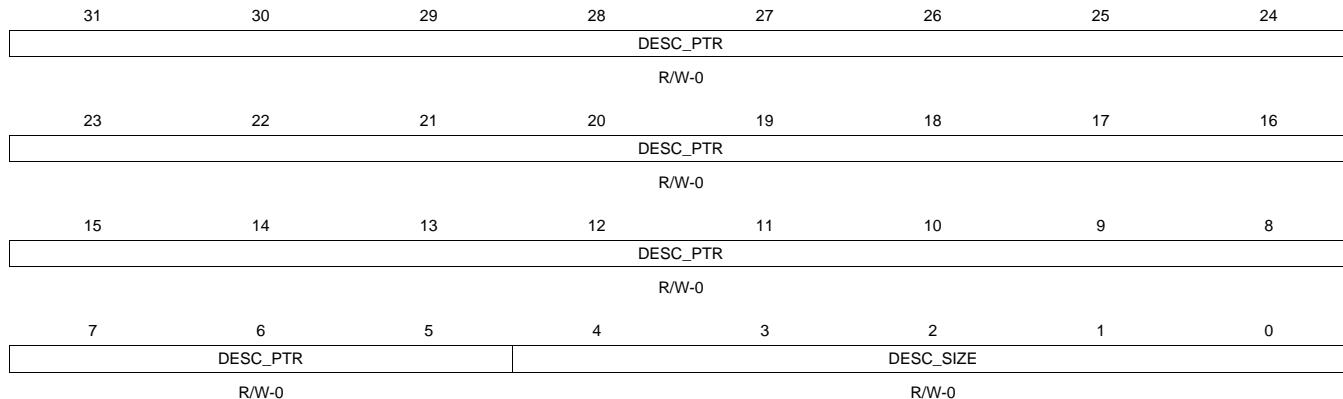
**Table 16-747. QUEUE\_105\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.458 QUEUE\_105\_D Register (offset = 269Ch) [reset = 0h]

QUEUE\_105\_D is shown in [Figure 16-734](#) and described in [Table 16-748](#).

**Figure 16-734. QUEUE\_105\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-748. QUEUE\_105\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.459 QUEUE\_106\_A Register (offset = 26A0h) [reset = 0h]**

QUEUE\_106\_A is shown in [Figure 16-735](#) and described in [Table 16-749](#).

**Figure 16-735. QUEUE\_106\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-749. QUEUE\_106\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.460 QUEUE\_106\_B Register (offset = 26A4h) [reset = 0h]**

QUEUE\_106\_B is shown in [Figure 16-736](#) and described in [Table 16-750](#).

**Figure 16-736. QUEUE\_106\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

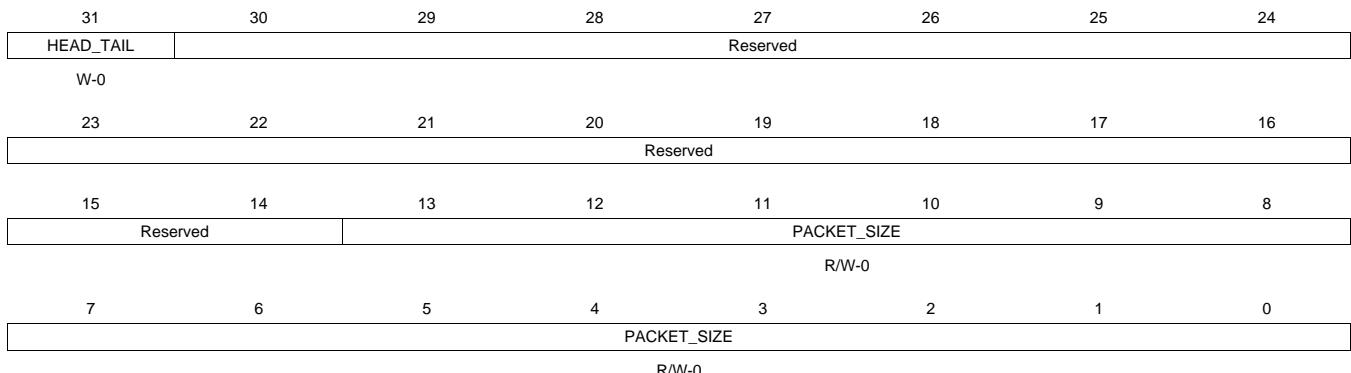
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-750. QUEUE\_106\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.461 QUEUE\_106\_C Register (offset = 26A8h) [reset = 0h]**

QUEUE\_106\_C is shown in [Figure 16-737](#) and described in [Table 16-751](#).

**Figure 16-737. QUEUE\_106\_C Register**


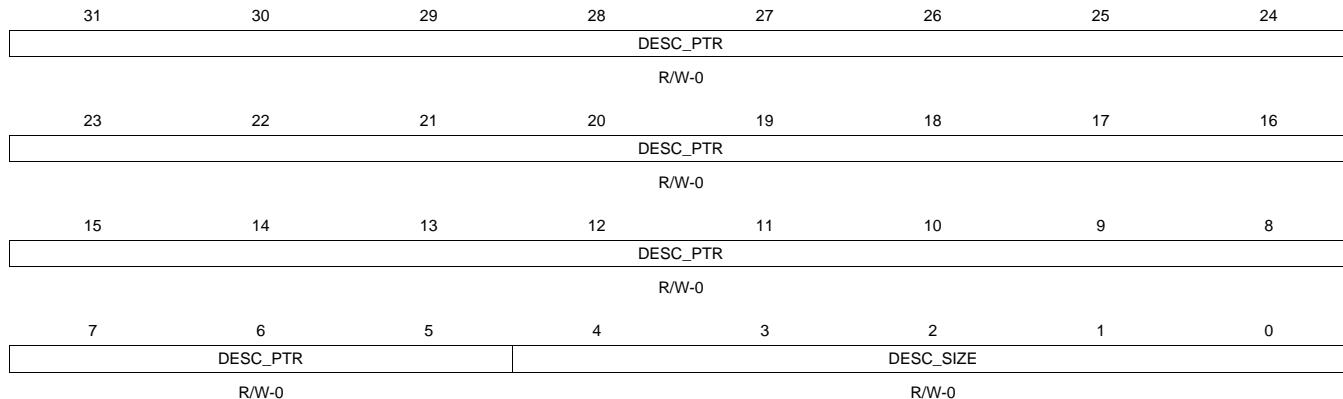
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-751. QUEUE\_106\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.462 QUEUE\_106\_D Register (offset = 26ACh) [reset = 0h]**

 QUEUE\_106\_D is shown in [Figure 16-738](#) and described in [Table 16-752](#).

**Figure 16-738. QUEUE\_106\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-752. QUEUE\_106\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.463 QUEUE\_107\_A Register (offset = 26B0h) [reset = 0h]**

QUEUE\_107\_A is shown in [Figure 16-739](#) and described in [Table 16-753](#).

**Figure 16-739. QUEUE\_107\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-753. QUEUE\_107\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.464 QUEUE\_107\_B Register (offset = 26B4h) [reset = 0h]**

QUEUE\_107\_B is shown in [Figure 16-740](#) and described in [Table 16-754](#).

**Figure 16-740. QUEUE\_107\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

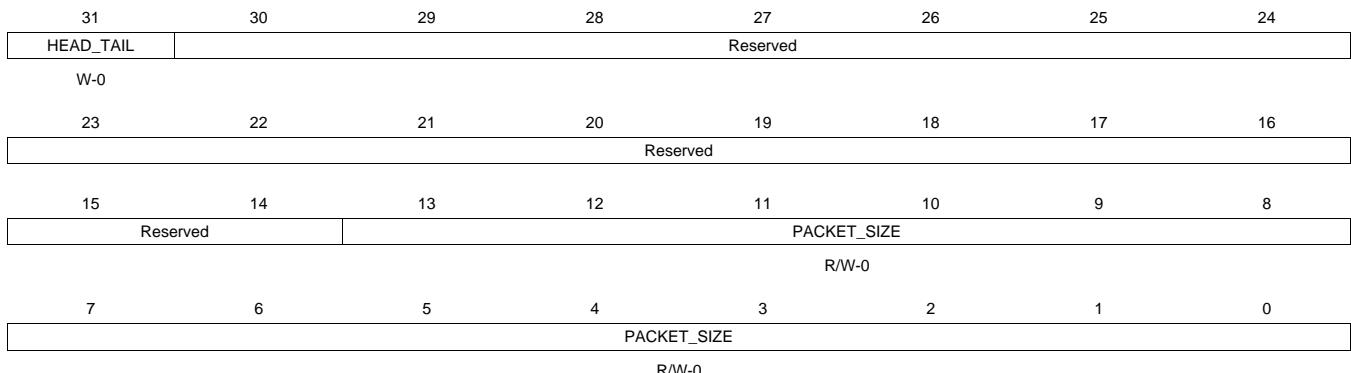
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-754. QUEUE\_107\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.465 QUEUE\_107\_C Register (offset = 26B8h) [reset = 0h]**

QUEUE\_107\_C is shown in [Figure 16-741](#) and described in [Table 16-755](#).

**Figure 16-741. QUEUE\_107\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

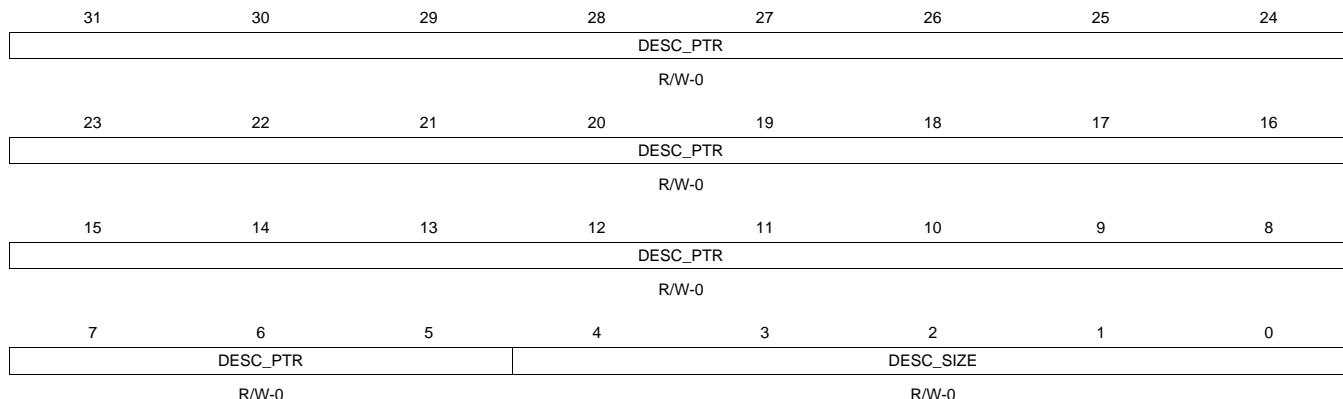
**Table 16-755. QUEUE\_107\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.466 QUEUE\_107\_D Register (offset = 26BCh) [reset = 0h]

QUEUE\_107\_D is shown in [Figure 16-742](#) and described in [Table 16-756](#).

**Figure 16-742. QUEUE\_107\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-756. QUEUE\_107\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.467 QUEUE\_108\_A Register (offset = 26C0h) [reset = 0h]**

QUEUE\_108\_A is shown in [Figure 16-743](#) and described in [Table 16-757](#).

**Figure 16-743. QUEUE\_108\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-757. QUEUE\_108\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.468 QUEUE\_108\_B Register (offset = 26C4h) [reset = 0h]**

QUEUE\_108\_B is shown in [Figure 16-744](#) and described in [Table 16-758](#).

**Figure 16-744. QUEUE\_108\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

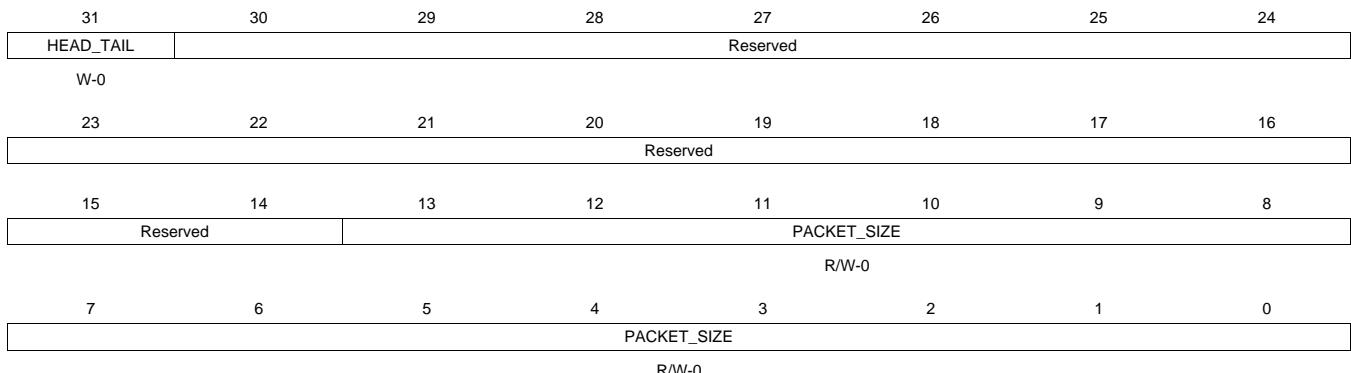
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-758. QUEUE\_108\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.469 QUEUE\_108\_C Register (offset = 26C8h) [reset = 0h]**

QUEUE\_108\_C is shown in [Figure 16-745](#) and described in [Table 16-759](#).

**Figure 16-745. QUEUE\_108\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

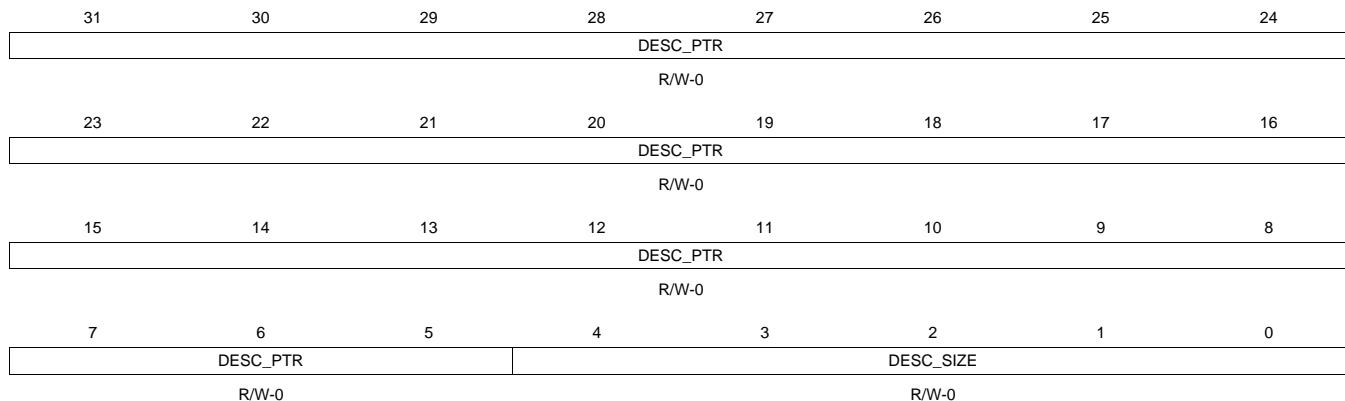
**Table 16-759. QUEUE\_108\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.470 QUEUE\_108\_D Register (offset = 26CCh) [reset = 0h]

QUEUE\_108\_D is shown in [Figure 16-746](#) and described in [Table 16-760](#).

**Figure 16-746. QUEUE\_108\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-760. QUEUE\_108\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.471 QUEUE\_109\_A Register (offset = 26D0h) [reset = 0h]**

QUEUE\_109\_A is shown in [Figure 16-747](#) and described in [Table 16-761](#).

**Figure 16-747. QUEUE\_109\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-761. QUEUE\_109\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>This count is incremented by 1 whenever a packet is added to the queue.<br>This count is decremented by 1 whenever a packet is popped from the queue. |

**16.5.7.472 QUEUE\_109\_B Register (offset = 26D4h) [reset = 0h]**

QUEUE\_109\_B is shown in [Figure 16-748](#) and described in [Table 16-762](#).

**Figure 16-748. QUEUE\_109\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

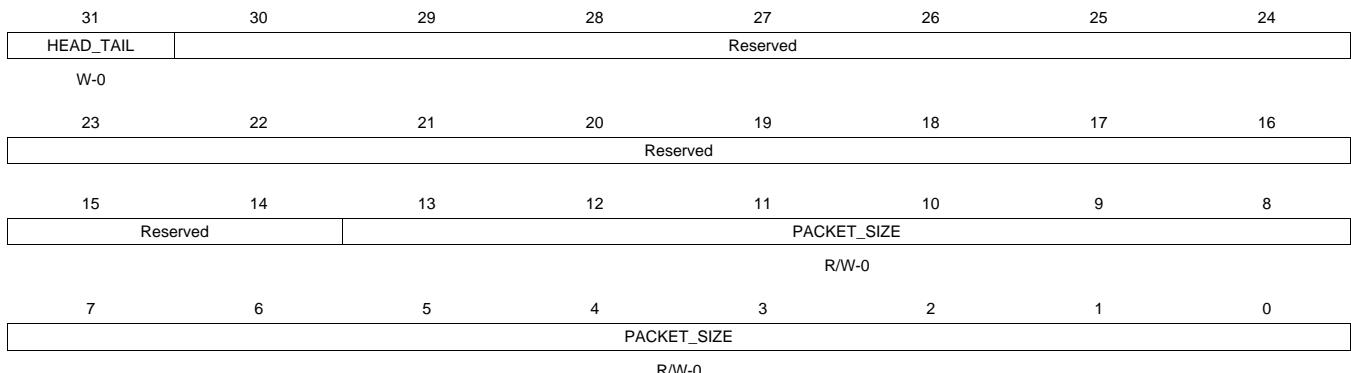
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-762. QUEUE\_109\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.473 QUEUE\_109\_C Register (offset = 26D8h) [reset = 0h]**

QUEUE\_109\_C is shown in [Figure 16-749](#) and described in [Table 16-763](#).

**Figure 16-749. QUEUE\_109\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

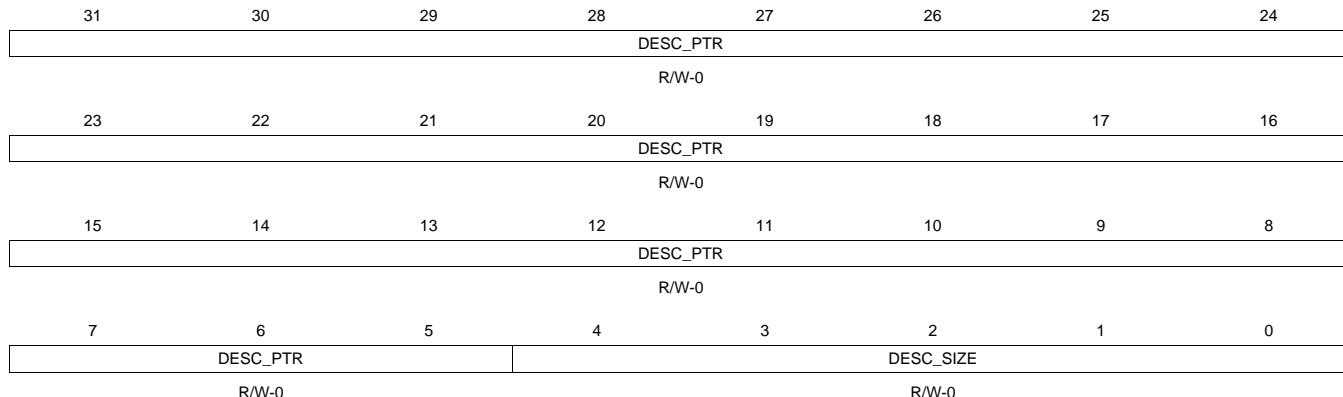
**Table 16-763. QUEUE\_109\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.474 QUEUE\_109\_D Register (offset = 26DCh) [reset = 0h]

QUEUE\_109\_D is shown in [Figure 16-750](#) and described in [Table 16-764](#).

**Figure 16-750. QUEUE\_109\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-764. QUEUE\_109\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.475 QUEUE\_110\_A Register (offset = 26E0h) [reset = 0h]**

QUEUE\_110\_A is shown in [Figure 16-751](#) and described in [Table 16-765](#).

**Figure 16-751. QUEUE\_110\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-765. QUEUE\_110\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.476 QUEUE\_110\_B Register (offset = 26E4h) [reset = 0h]**

QUEUE\_110\_B is shown in [Figure 16-752](#) and described in [Table 16-766](#).

**Figure 16-752. QUEUE\_110\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

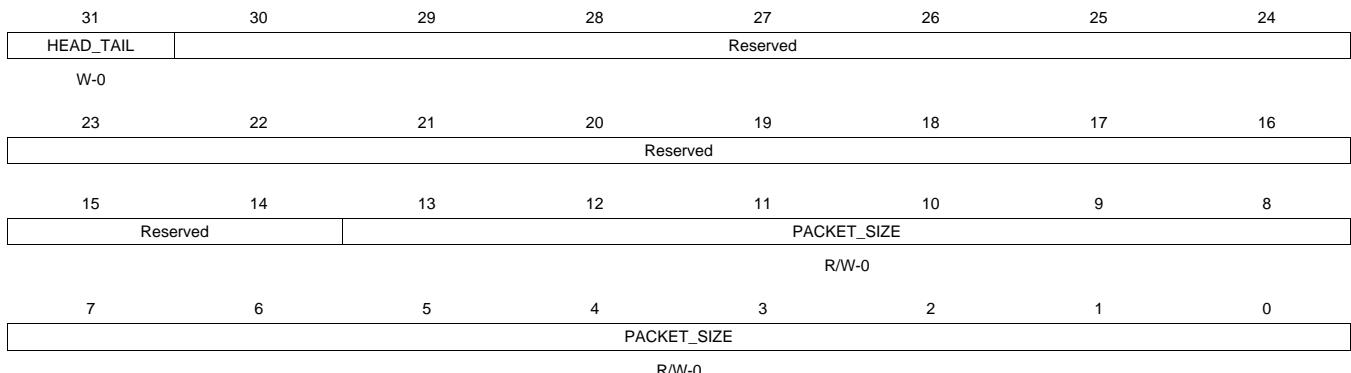
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-766. QUEUE\_110\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.477 QUEUE\_110\_C Register (offset = 26E8h) [reset = 0h]**

QUEUE\_110\_C is shown in [Figure 16-753](#) and described in [Table 16-767](#).

**Figure 16-753. QUEUE\_110\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

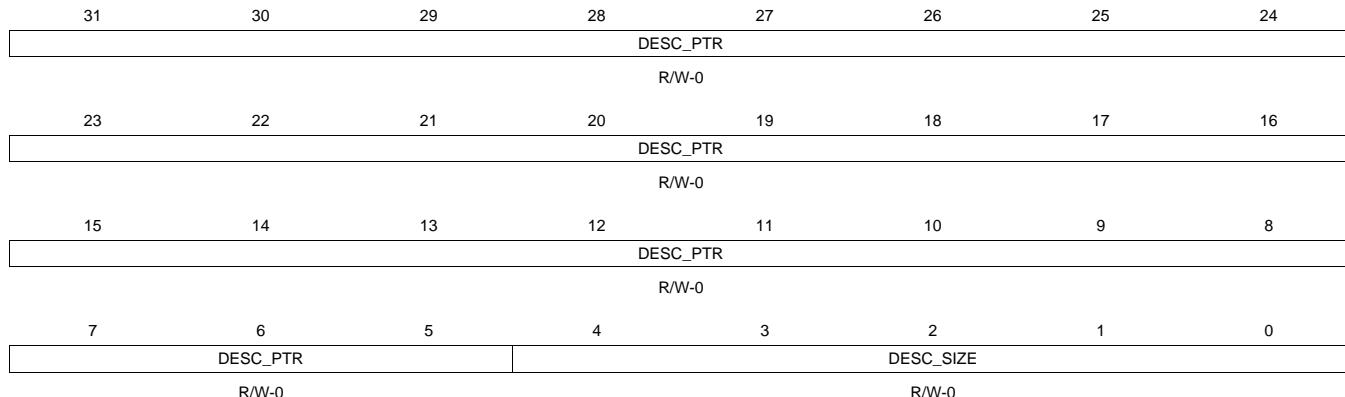
**Table 16-767. QUEUE\_110\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.478 QUEUE\_110\_D Register (offset = 26ECh) [reset = 0h]

QUEUE\_110\_D is shown in [Figure 16-754](#) and described in [Table 16-768](#).

**Figure 16-754. QUEUE\_110\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-768. QUEUE\_110\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.479 QUEUE\_111\_A Register (offset = 26F0h) [reset = 0h]**

QUEUE\_111\_A is shown in [Figure 16-755](#) and described in [Table 16-769](#).

**Figure 16-755. QUEUE\_111\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-769. QUEUE\_111\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.480 QUEUE\_111\_B Register (offset = 26F4h) [reset = 0h]**

QUEUE\_111\_B is shown in [Figure 16-756](#) and described in [Table 16-770](#).

**Figure 16-756. QUEUE\_111\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

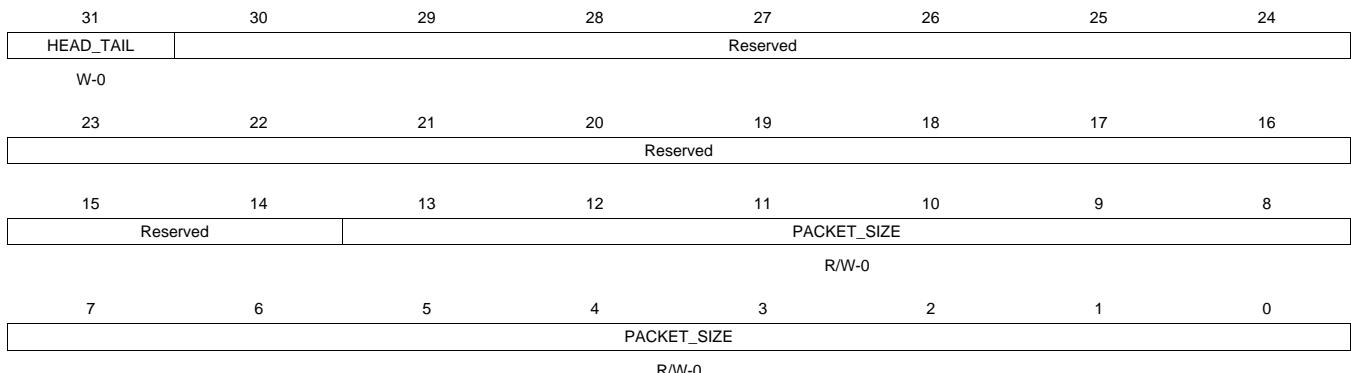
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-770. QUEUE\_111\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.481 QUEUE\_111\_C Register (offset = 26F8h) [reset = 0h]**

QUEUE\_111\_C is shown in [Figure 16-757](#) and described in [Table 16-771](#).

**Figure 16-757. QUEUE\_111\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

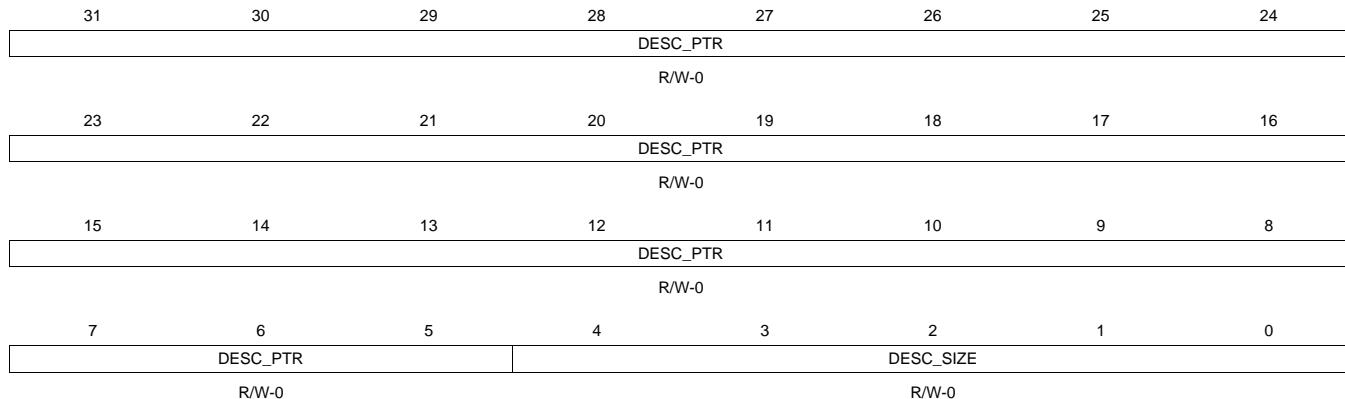
**Table 16-771. QUEUE\_111\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.482 QUEUE\_111\_D Register (offset = 26FCh) [reset = 0h]

QUEUE\_111\_D is shown in [Figure 16-758](#) and described in [Table 16-772](#).

**Figure 16-758. QUEUE\_111\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-772. QUEUE\_111\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.483 QUEUE\_112\_A Register (offset = 2700h) [reset = 0h]**

QUEUE\_112\_A is shown in [Figure 16-759](#) and described in [Table 16-773](#).

**Figure 16-759. QUEUE\_112\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-773. QUEUE\_112\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.484 QUEUE\_112\_B Register (offset = 2704h) [reset = 0h]**

QUEUE\_112\_B is shown in [Figure 16-760](#) and described in [Table 16-774](#).

**Figure 16-760. QUEUE\_112\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

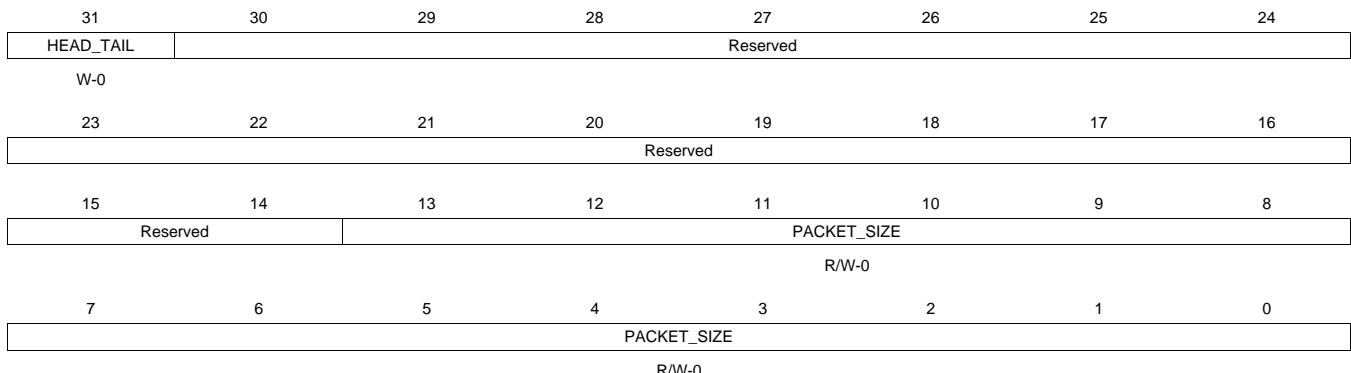
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-774. QUEUE\_112\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.485 QUEUE\_112\_C Register (offset = 2708h) [reset = 0h]**

QUEUE\_112\_C is shown in [Figure 16-761](#) and described in [Table 16-775](#).

**Figure 16-761. QUEUE\_112\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

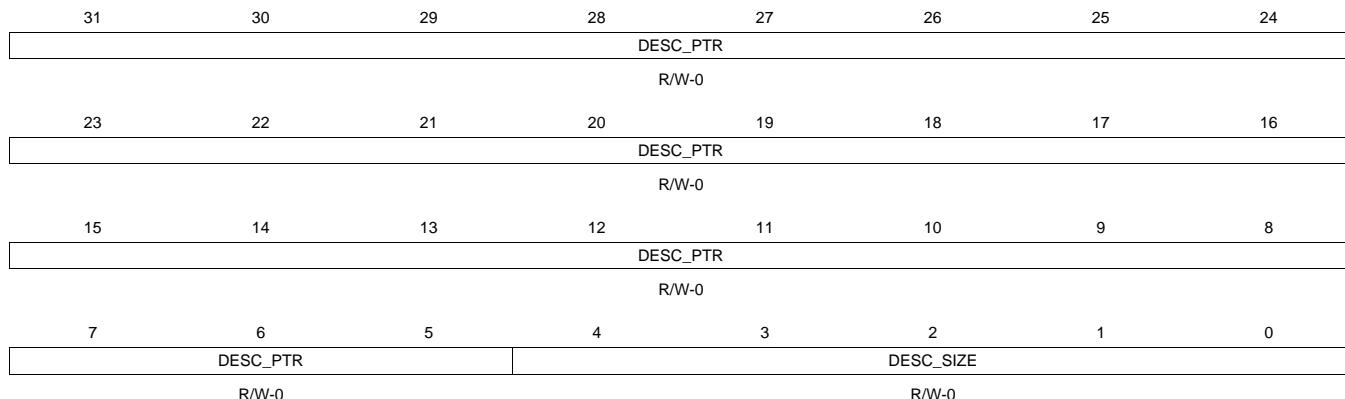
**Table 16-775. QUEUE\_112\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.486 QUEUE\_112\_D Register (offset = 270Ch) [reset = 0h]

QUEUE\_112\_D is shown in [Figure 16-762](#) and described in [Table 16-776](#).

**Figure 16-762. QUEUE\_112\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-776. QUEUE\_112\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.487 QUEUE\_113\_A Register (offset = 2710h) [reset = 0h]**

QUEUE\_113\_A is shown in [Figure 16-763](#) and described in [Table 16-777](#).

**Figure 16-763. QUEUE\_113\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-777. QUEUE\_113\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.488 QUEUE\_113\_B Register (offset = 2714h) [reset = 0h]**

QUEUE\_113\_B is shown in [Figure 16-764](#) and described in [Table 16-778](#).

**Figure 16-764. QUEUE\_113\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

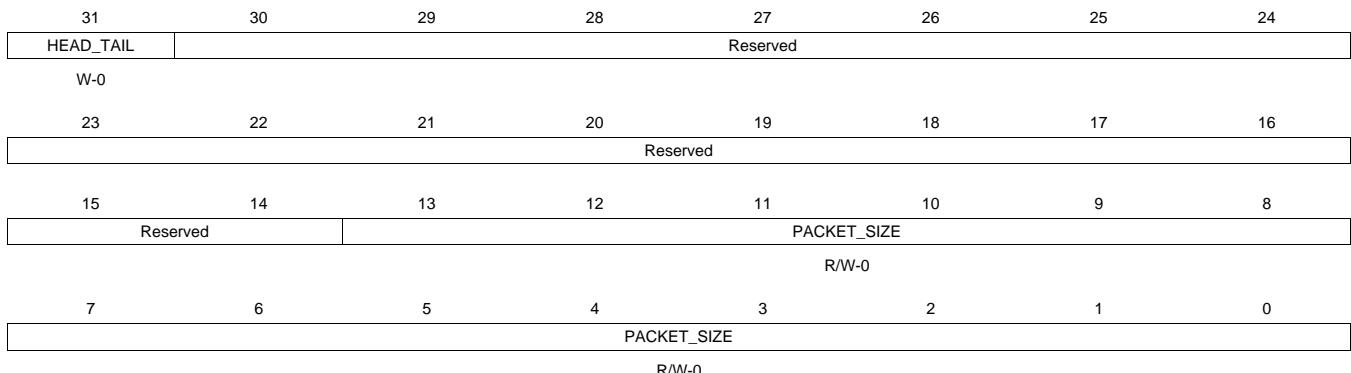
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-778. QUEUE\_113\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.489 QUEUE\_113\_C Register (offset = 2718h) [reset = 0h]**

QUEUE\_113\_C is shown in [Figure 16-765](#) and described in [Table 16-779](#).

**Figure 16-765. QUEUE\_113\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

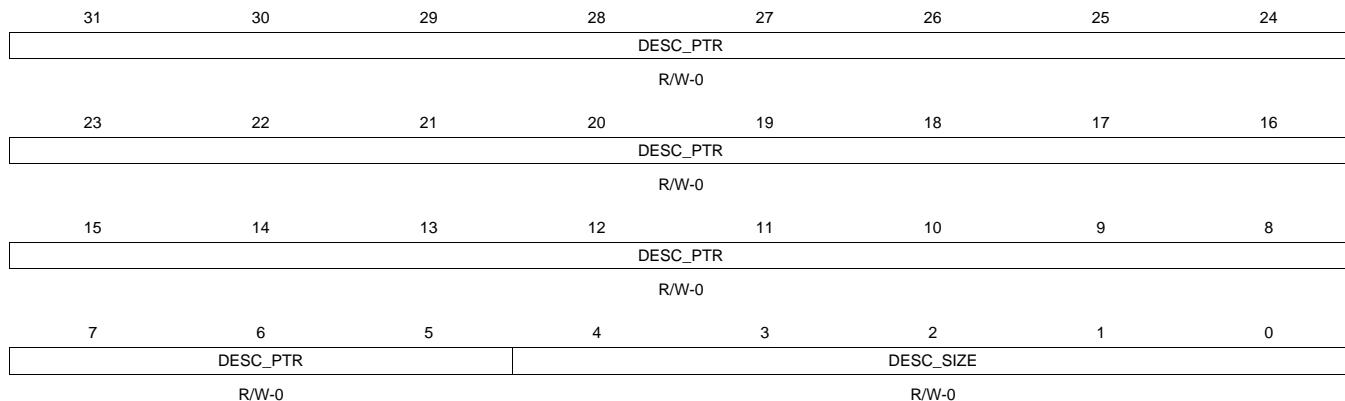
**Table 16-779. QUEUE\_113\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.490 QUEUE\_113\_D Register (offset = 271Ch) [reset = 0h]

QUEUE\_113\_D is shown in [Figure 16-766](#) and described in [Table 16-780](#).

**Figure 16-766. QUEUE\_113\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-780. QUEUE\_113\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.491 QUEUE\_114\_A Register (offset = 2720h) [reset = 0h]**

QUEUE\_114\_A is shown in [Figure 16-767](#) and described in [Table 16-781](#).

**Figure 16-767. QUEUE\_114\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-781. QUEUE\_114\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.492 QUEUE\_114\_B Register (offset = 2724h) [reset = 0h]**

QUEUE\_114\_B is shown in [Figure 16-768](#) and described in [Table 16-782](#).

**Figure 16-768. QUEUE\_114\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

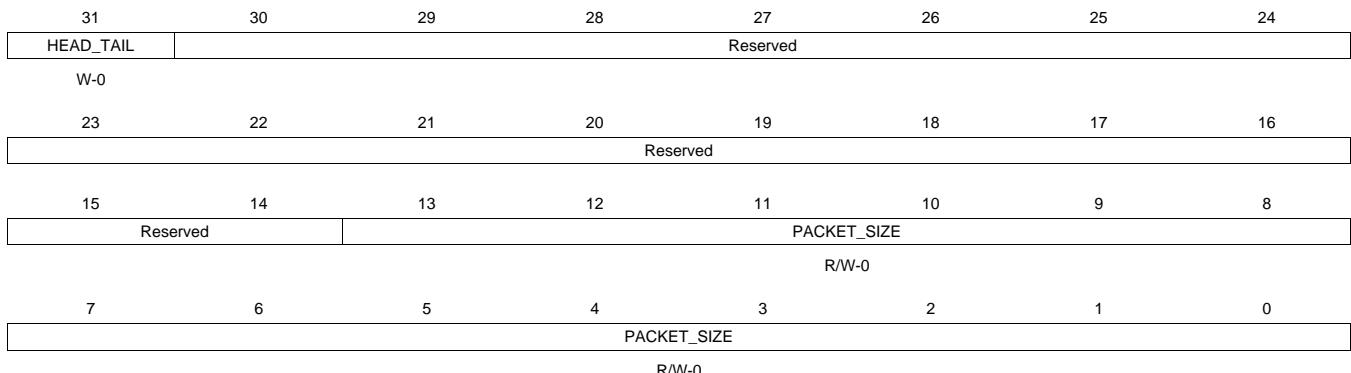
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-782. QUEUE\_114\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.493 QUEUE\_114\_C Register (offset = 2728h) [reset = 0h]**

QUEUE\_114\_C is shown in [Figure 16-769](#) and described in [Table 16-783](#).

**Figure 16-769. QUEUE\_114\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

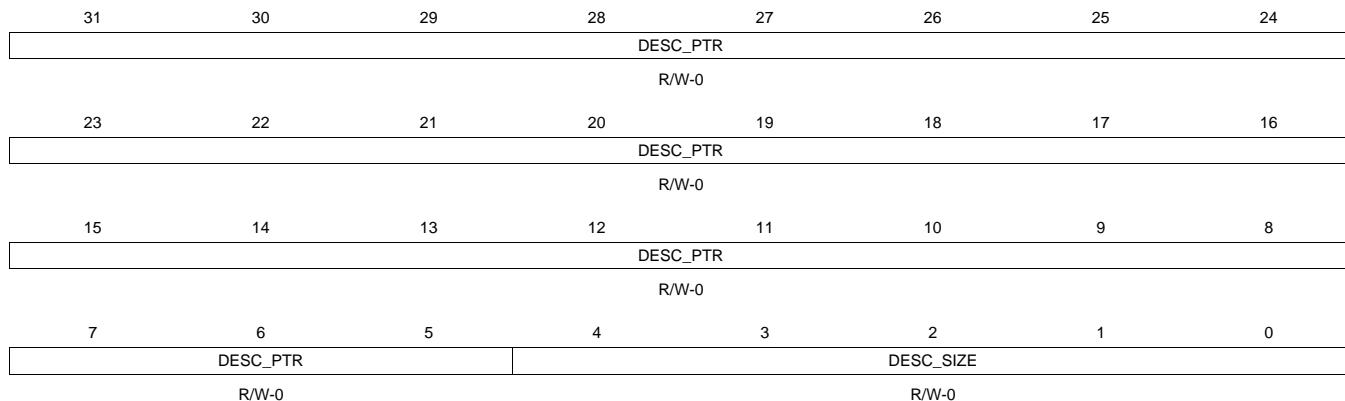
**Table 16-783. QUEUE\_114\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.494 QUEUE\_114\_D Register (offset = 272Ch) [reset = 0h]

QUEUE\_114\_D is shown in [Figure 16-770](#) and described in [Table 16-784](#).

**Figure 16-770. QUEUE\_114\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-784. QUEUE\_114\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.495 QUEUE\_115\_A Register (offset = 2730h) [reset = 0h]**

QUEUE\_115\_A is shown in [Figure 16-771](#) and described in [Table 16-785](#).

**Figure 16-771. QUEUE\_115\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-785. QUEUE\_115\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.496 QUEUE\_115\_B Register (offset = 2734h) [reset = 0h]**

QUEUE\_115\_B is shown in [Figure 16-772](#) and described in [Table 16-786](#).

**Figure 16-772. QUEUE\_115\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

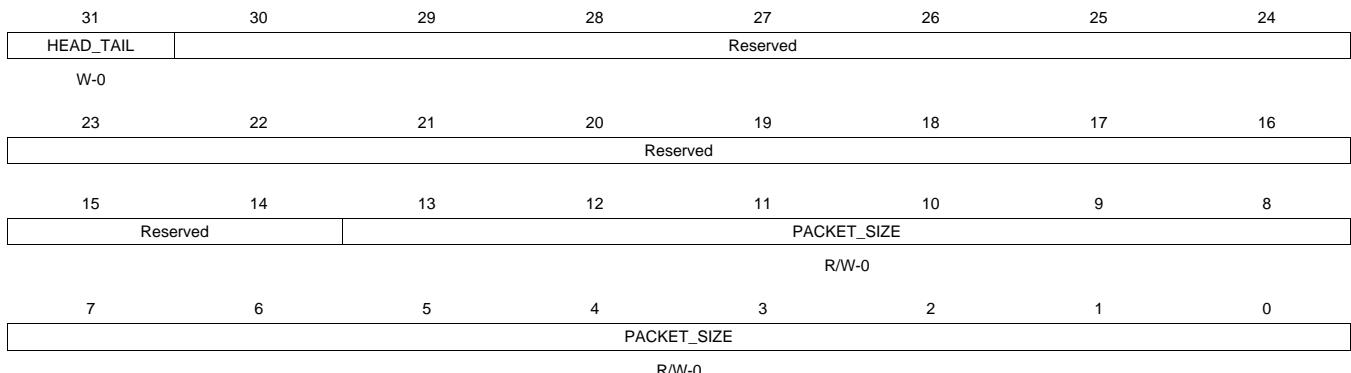
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-786. QUEUE\_115\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.497 QUEUE\_115\_C Register (offset = 2738h) [reset = 0h]**

QUEUE\_115\_C is shown in [Figure 16-773](#) and described in [Table 16-787](#).

**Figure 16-773. QUEUE\_115\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

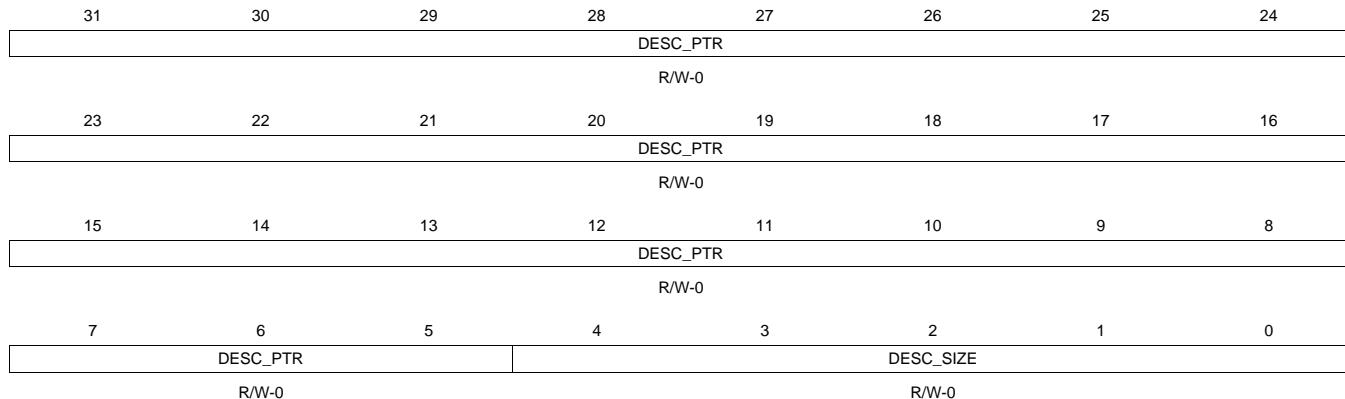
**Table 16-787. QUEUE\_115\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.498 QUEUE\_115\_D Register (offset = 273Ch) [reset = 0h]

QUEUE\_115\_D is shown in [Figure 16-774](#) and described in [Table 16-788](#).

**Figure 16-774. QUEUE\_115\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-788. QUEUE\_115\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.499 QUEUE\_116\_A Register (offset = 2740h) [reset = 0h]**

QUEUE\_116\_A is shown in [Figure 16-775](#) and described in [Table 16-789](#).

**Figure 16-775. QUEUE\_116\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-789. QUEUE\_116\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.500 QUEUE\_116\_B Register (offset = 2744h) [reset = 0h]**

QUEUE\_116\_B is shown in [Figure 16-776](#) and described in [Table 16-790](#).

**Figure 16-776. QUEUE\_116\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

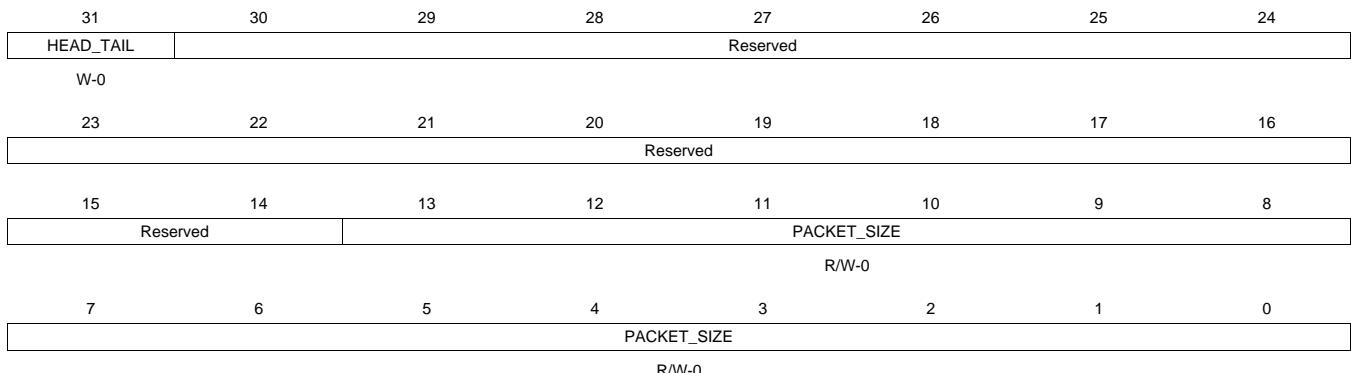
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-790. QUEUE\_116\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.501 QUEUE\_116\_C Register (offset = 2748h) [reset = 0h]**

QUEUE\_116\_C is shown in [Figure 16-777](#) and described in [Table 16-791](#).

**Figure 16-777. QUEUE\_116\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

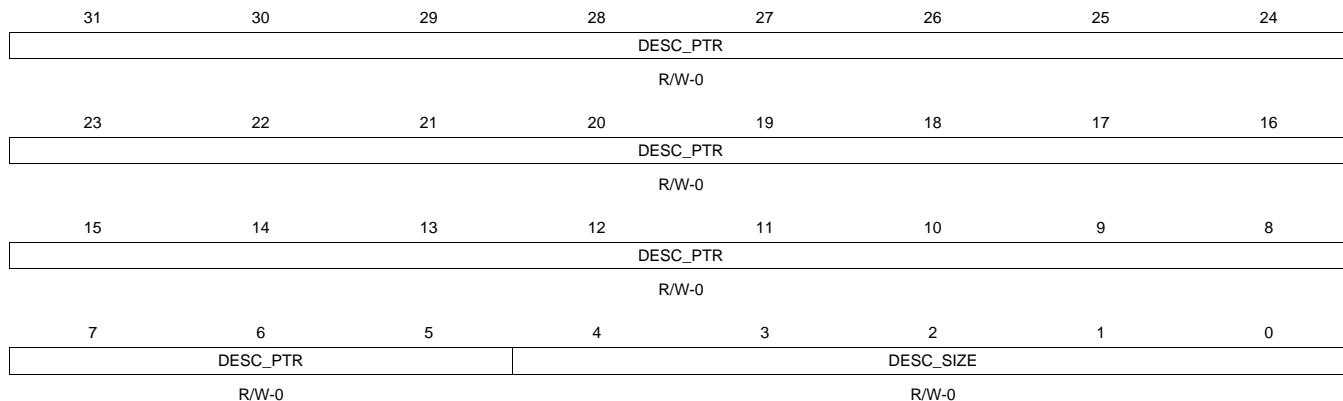
**Table 16-791. QUEUE\_116\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.502 QUEUE\_116\_D Register (offset = 274Ch) [reset = 0h]

QUEUE\_116\_D is shown in [Figure 16-778](#) and described in [Table 16-792](#).

**Figure 16-778. QUEUE\_116\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-792. QUEUE\_116\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.503 QUEUE\_117\_A Register (offset = 2750h) [reset = 0h]**

QUEUE\_117\_A is shown in [Figure 16-779](#) and described in [Table 16-793](#).

**Figure 16-779. QUEUE\_117\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-793. QUEUE\_117\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.504 QUEUE\_117\_B Register (offset = 2754h) [reset = 0h]**

QUEUE\_117\_B is shown in [Figure 16-780](#) and described in [Table 16-794](#).

**Figure 16-780. QUEUE\_117\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

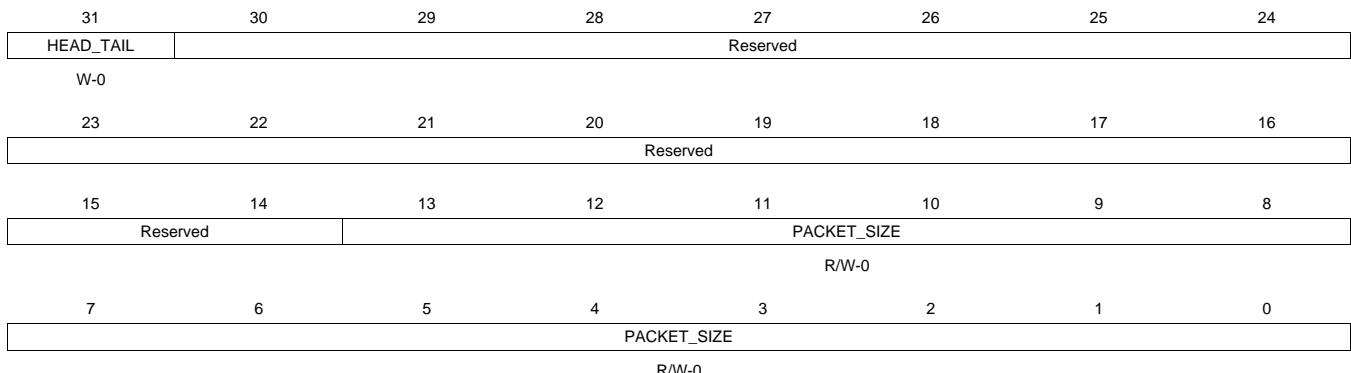
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-794. QUEUE\_117\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.505 QUEUE\_117\_C Register (offset = 2758h) [reset = 0h]**

QUEUE\_117\_C is shown in [Figure 16-781](#) and described in [Table 16-795](#).

**Figure 16-781. QUEUE\_117\_C Register**


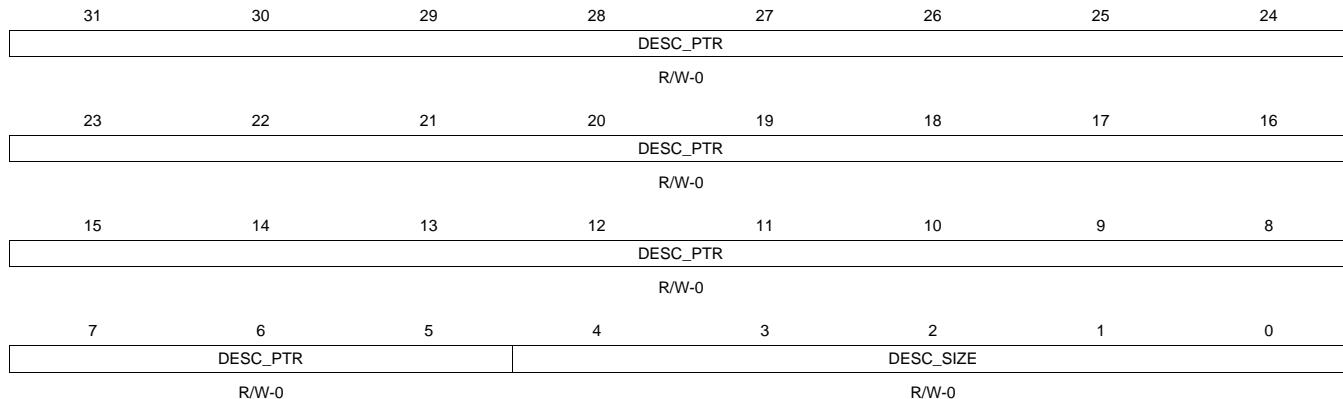
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-795. QUEUE\_117\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.506 QUEUE\_117\_D Register (offset = 275Ch) [reset = 0h]**

 QUEUE\_117\_D is shown in [Figure 16-782](#) and described in [Table 16-796](#).

**Figure 16-782. QUEUE\_117\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-796. QUEUE\_117\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.507 QUEUE\_118\_A Register (offset = 2760h) [reset = 0h]

QUEUE\_118\_A is shown in [Figure 16-783](#) and described in [Table 16-797](#).

**Figure 16-783. QUEUE\_118\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-797. QUEUE\_118\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.508 QUEUE\_118\_B Register (offset = 2764h) [reset = 0h]**

QUEUE\_118\_B is shown in [Figure 16-784](#) and described in [Table 16-798](#).

**Figure 16-784. QUEUE\_118\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

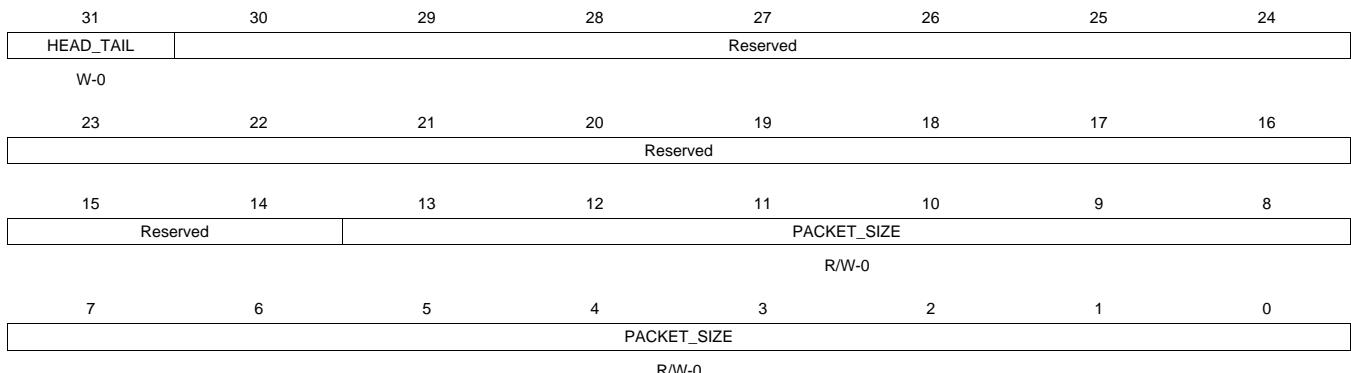
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-798. QUEUE\_118\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.509 QUEUE\_118\_C Register (offset = 2768h) [reset = 0h]**

QUEUE\_118\_C is shown in [Figure 16-785](#) and described in [Table 16-799](#).

**Figure 16-785. QUEUE\_118\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

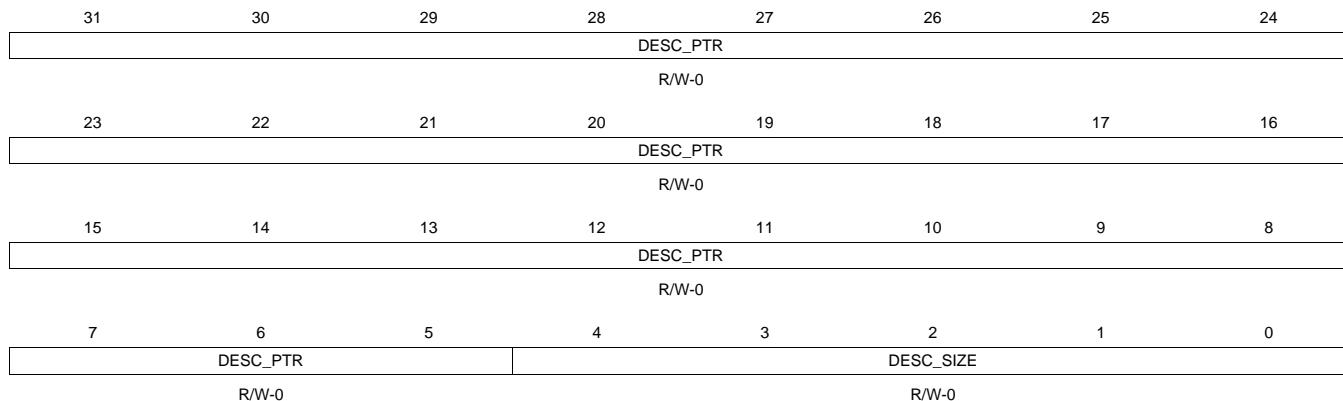
**Table 16-799. QUEUE\_118\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.510 QUEUE\_118\_D Register (offset = 276Ch) [reset = 0h]

QUEUE\_118\_D is shown in [Figure 16-786](#) and described in [Table 16-800](#).

**Figure 16-786. QUEUE\_118\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-800. QUEUE\_118\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

### 16.5.7.511 QUEUE\_119\_A Register (offset = 2770h) [reset = 0h]

QUEUE\_119\_A is shown in [Figure 16-787](#) and described in [Table 16-801](#).

**Figure 16-787. QUEUE\_119\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-801. QUEUE\_119\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.512 QUEUE\_119\_B Register (offset = 2774h) [reset = 0h]**

QUEUE\_119\_B is shown in [Figure 16-788](#) and described in [Table 16-802](#).

**Figure 16-788. QUEUE\_119\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

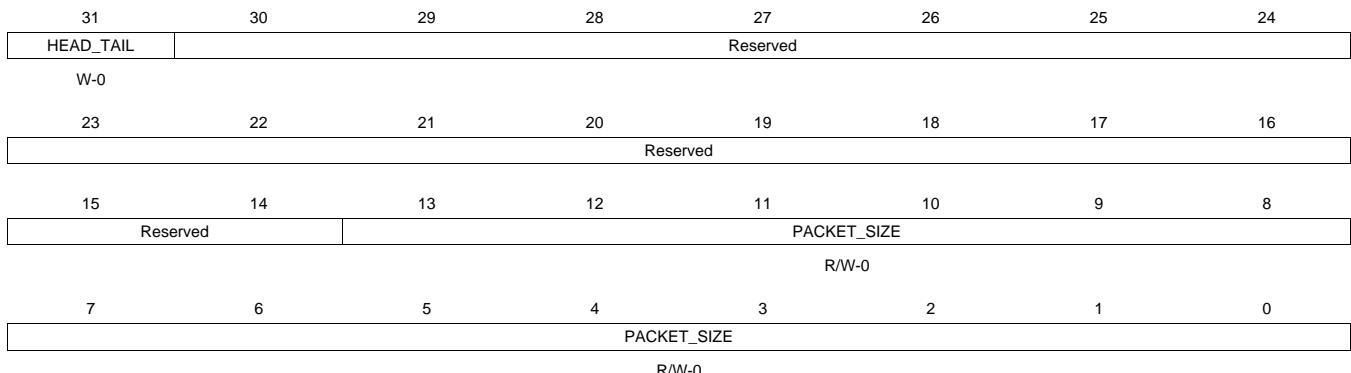
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-802. QUEUE\_119\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.513 QUEUE\_119\_C Register (offset = 2778h) [reset = 0h]**

QUEUE\_119\_C is shown in [Figure 16-789](#) and described in [Table 16-803](#).

**Figure 16-789. QUEUE\_119\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

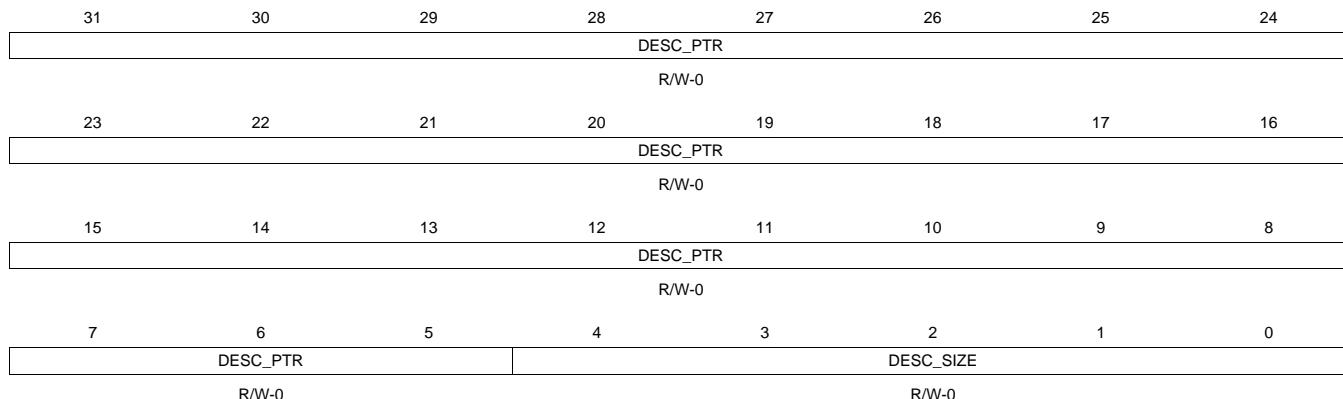
**Table 16-803. QUEUE\_119\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.514 QUEUE\_119\_D Register (offset = 277Ch) [reset = 0h]

QUEUE\_119\_D is shown in [Figure 16-790](#) and described in [Table 16-804](#).

**Figure 16-790. QUEUE\_119\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-804. QUEUE\_119\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.515 QUEUE\_120\_A Register (offset = 2780h) [reset = 0h]**

QUEUE\_120\_A is shown in [Figure 16-791](#) and described in [Table 16-805](#).

**Figure 16-791. QUEUE\_120\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-805. QUEUE\_120\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

### 16.5.7.516 QUEUE\_120\_B Register (offset = 2784h) [reset = 0h]

QUEUE\_120\_B is shown in [Figure 16-792](#) and described in [Table 16-806](#).

**Figure 16-792. QUEUE\_120\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

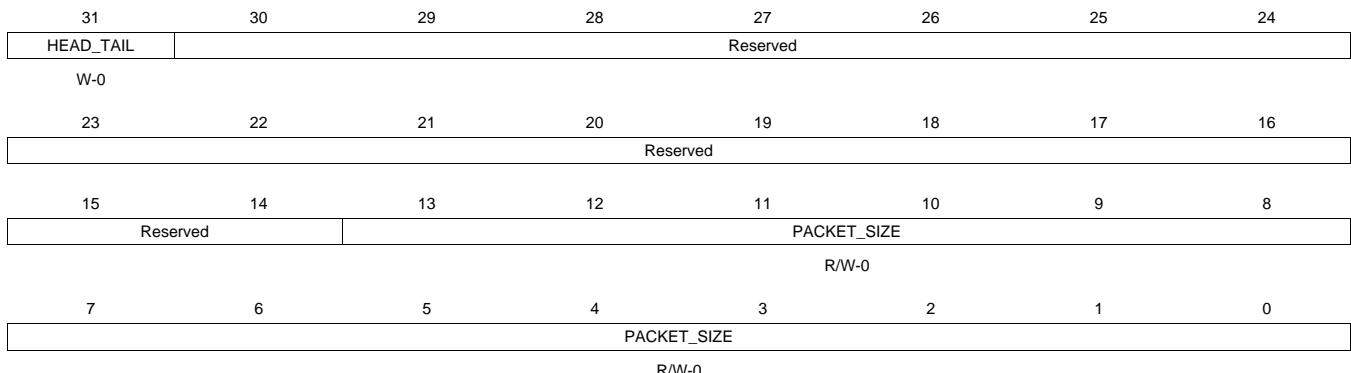
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-806. QUEUE\_120\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.517 QUEUE\_120\_C Register (offset = 2788h) [reset = 0h]**

 QUEUE\_120\_C is shown in [Figure 16-793](#) and described in [Table 16-807](#).

**Figure 16-793. QUEUE\_120\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

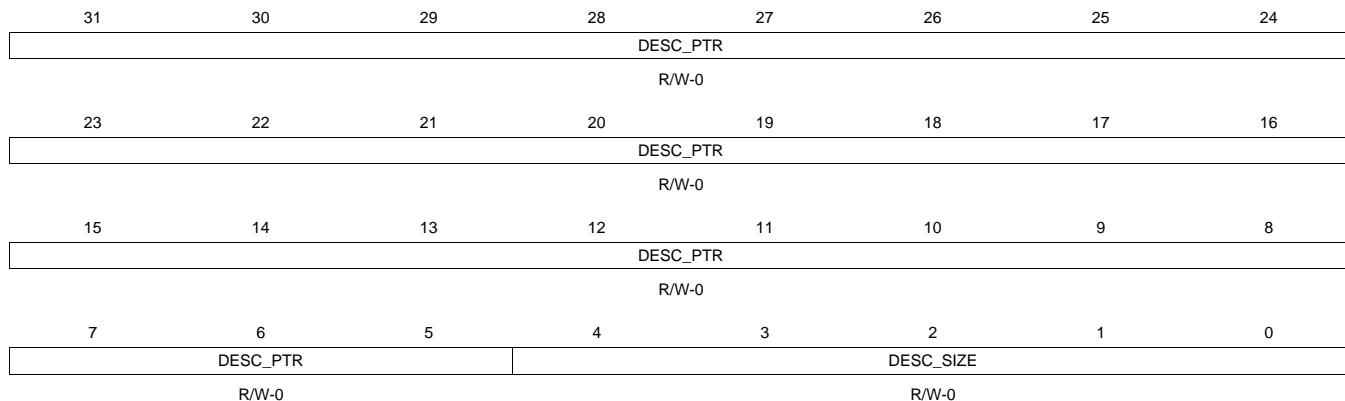
**Table 16-807. QUEUE\_120\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.518 QUEUE\_120\_D Register (offset = 278Ch) [reset = 0h]

QUEUE\_120\_D is shown in [Figure 16-794](#) and described in [Table 16-808](#).

**Figure 16-794. QUEUE\_120\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-808. QUEUE\_120\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.519 QUEUE\_121\_A Register (offset = 2790h) [reset = 0h]**

QUEUE\_121\_A is shown in [Figure 16-795](#) and described in [Table 16-809](#).

**Figure 16-795. QUEUE\_121\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-809. QUEUE\_121\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.520 QUEUE\_121\_B Register (offset = 2794h) [reset = 0h]**

QUEUE\_121\_B is shown in [Figure 16-796](#) and described in [Table 16-810](#).

**Figure 16-796. QUEUE\_121\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

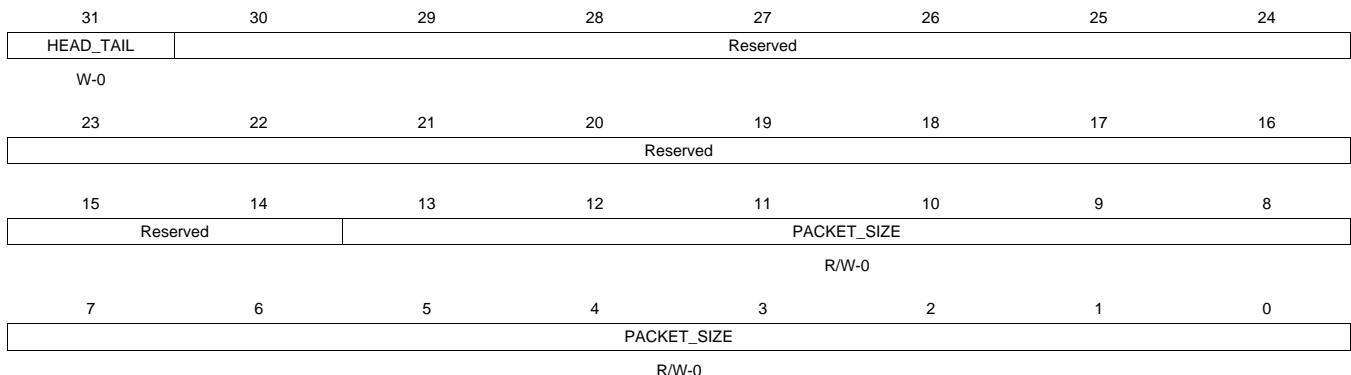
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-810. QUEUE\_121\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.521 QUEUE\_121\_C Register (offset = 2798h) [reset = 0h]**

QUEUE\_121\_C is shown in [Figure 16-797](#) and described in [Table 16-811](#).

**Figure 16-797. QUEUE\_121\_C Register**


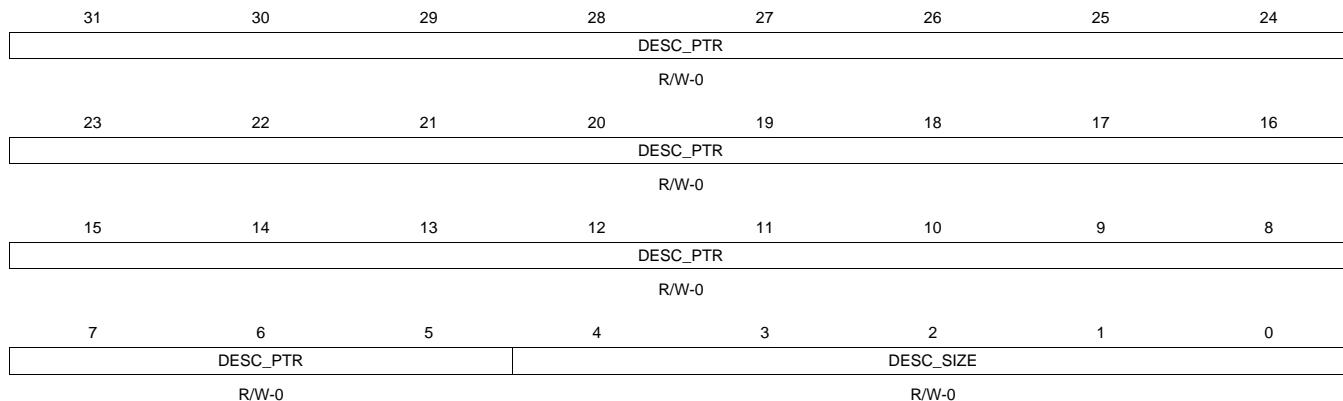
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-811. QUEUE\_121\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.522 QUEUE\_121\_D Register (offset = 279Ch) [reset = 0h]**

 QUEUE\_121\_D is shown in [Figure 16-798](#) and described in [Table 16-812](#).

**Figure 16-798. QUEUE\_121\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-812. QUEUE\_121\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.523 QUEUE\_122\_A Register (offset = 27A0h) [reset = 0h]**

QUEUE\_122\_A is shown in [Figure 16-799](#) and described in [Table 16-813](#).

**Figure 16-799. QUEUE\_122\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-813. QUEUE\_122\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.524 QUEUE\_122\_B Register (offset = 27A4h) [reset = 0h]**

QUEUE\_122\_B is shown in [Figure 16-800](#) and described in [Table 16-814](#).

**Figure 16-800. QUEUE\_122\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

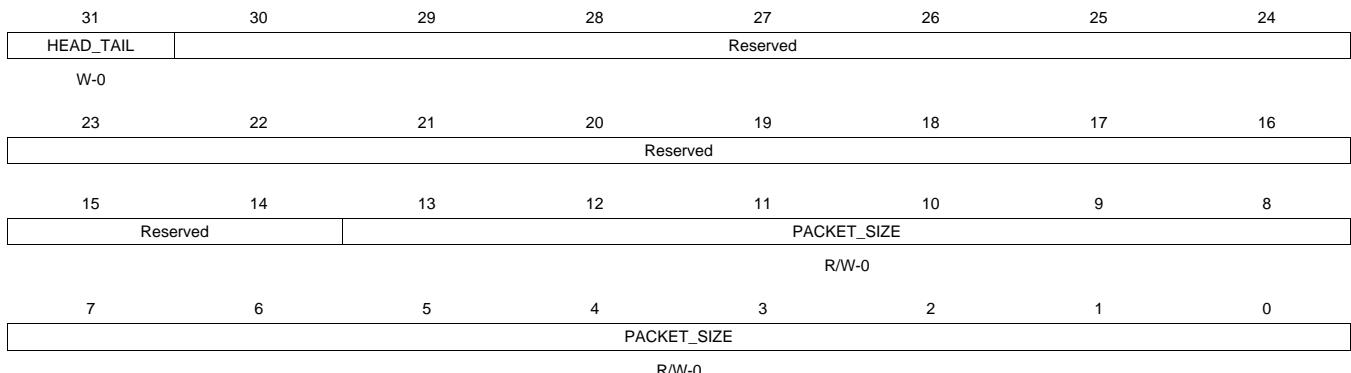
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-814. QUEUE\_122\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.525 QUEUE\_122\_C Register (offset = 27A8h) [reset = 0h]**

QUEUE\_122\_C is shown in [Figure 16-801](#) and described in [Table 16-815](#).

**Figure 16-801. QUEUE\_122\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

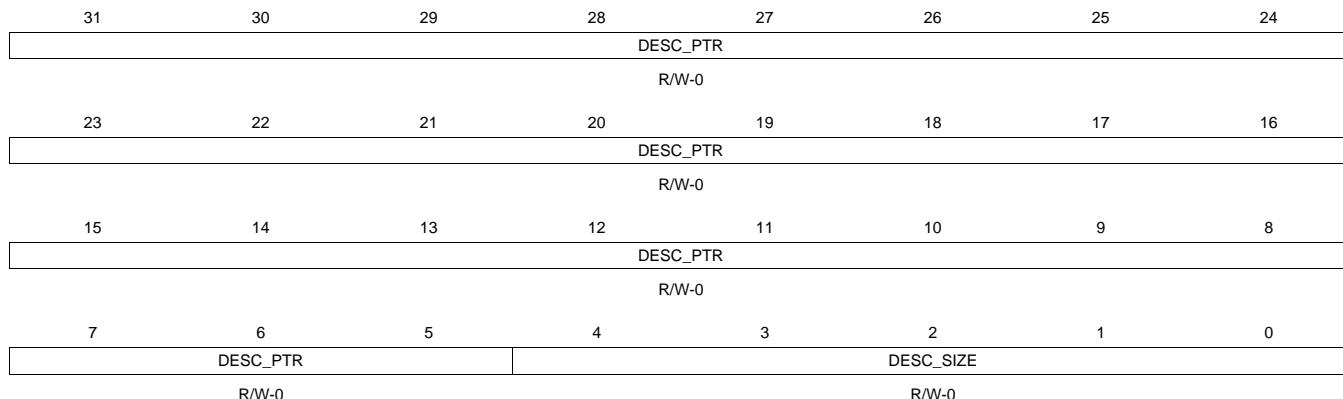
**Table 16-815. QUEUE\_122\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.526 QUEUE\_122\_D Register (offset = 27ACh) [reset = 0h]

QUEUE\_122\_D is shown in [Figure 16-802](#) and described in [Table 16-816](#).

**Figure 16-802. QUEUE\_122\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-816. QUEUE\_122\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.527 QUEUE\_123\_A Register (offset = 27B0h) [reset = 0h]**

QUEUE\_123\_A is shown in [Figure 16-803](#) and described in [Table 16-817](#).

**Figure 16-803. QUEUE\_123\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-817. QUEUE\_123\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.528 QUEUE\_123\_B Register (offset = 27B4h) [reset = 0h]**

QUEUE\_123\_B is shown in [Figure 16-804](#) and described in [Table 16-818](#).

**Figure 16-804. QUEUE\_123\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

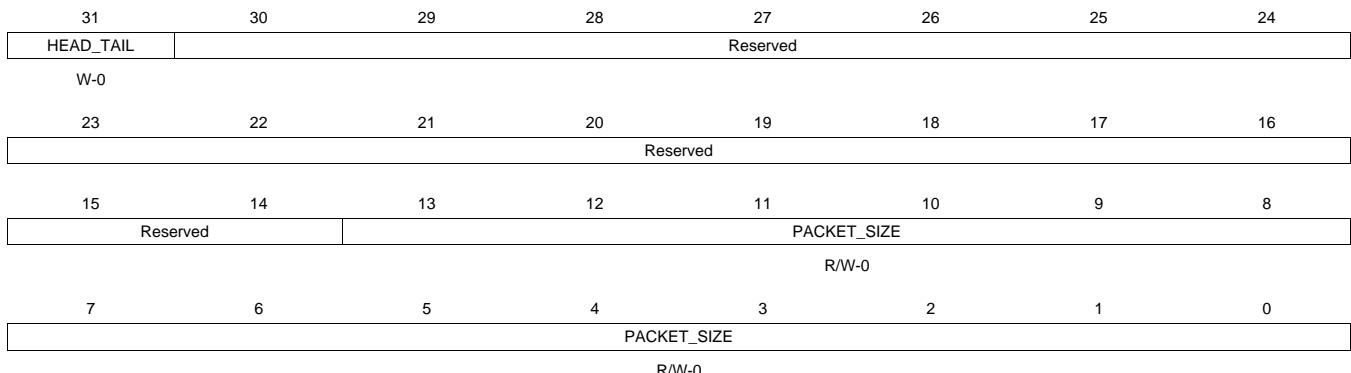
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-818. QUEUE\_123\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.529 QUEUE\_123\_C Register (offset = 27B8h) [reset = 0h]**

QUEUE\_123\_C is shown in [Figure 16-805](#) and described in [Table 16-819](#).

**Figure 16-805. QUEUE\_123\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

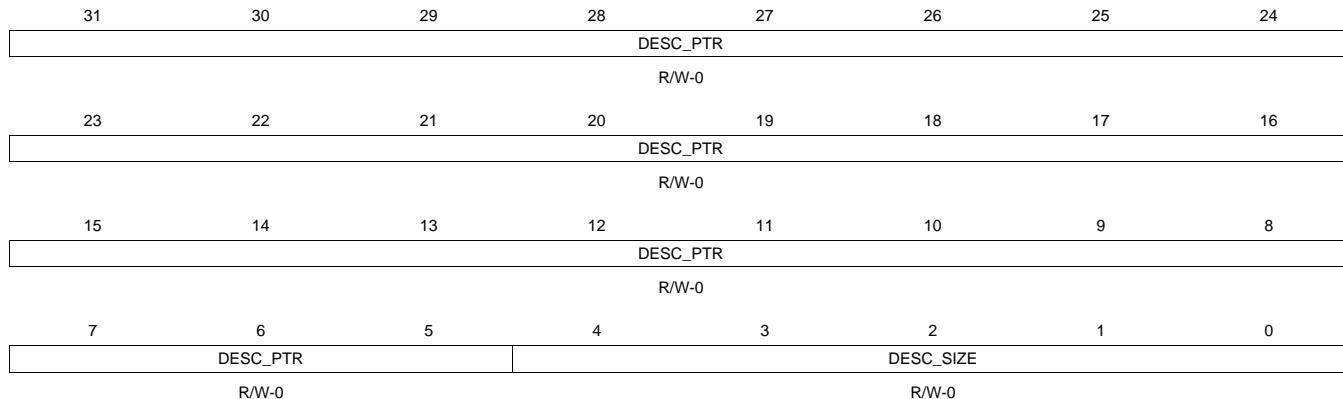
**Table 16-819. QUEUE\_123\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.530 QUEUE\_123\_D Register (offset = 27BCh) [reset = 0h]

QUEUE\_123\_D is shown in [Figure 16-806](#) and described in [Table 16-820](#).

**Figure 16-806. QUEUE\_123\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-820. QUEUE\_123\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.531 QUEUE\_124\_A Register (offset = 27C0h) [reset = 0h]**

QUEUE\_124\_A is shown in [Figure 16-807](#) and described in [Table 16-821](#).

**Figure 16-807. QUEUE\_124\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-821. QUEUE\_124\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.532 QUEUE\_124\_B Register (offset = 27C4h) [reset = 0h]**

QUEUE\_124\_B is shown in [Figure 16-808](#) and described in [Table 16-822](#).

**Figure 16-808. QUEUE\_124\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

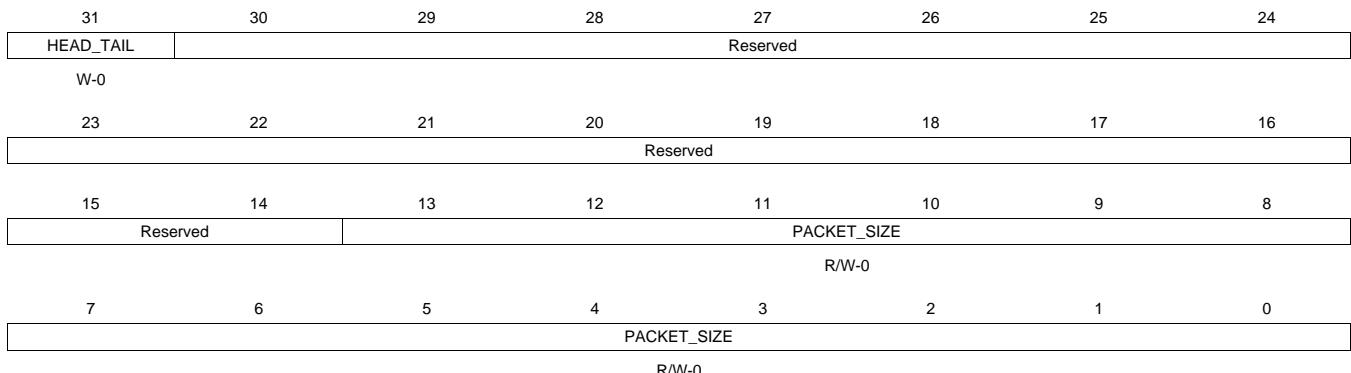
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-822. QUEUE\_124\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.533 QUEUE\_124\_C Register (offset = 27C8h) [reset = 0h]**

QUEUE\_124\_C is shown in [Figure 16-809](#) and described in [Table 16-823](#).

**Figure 16-809. QUEUE\_124\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

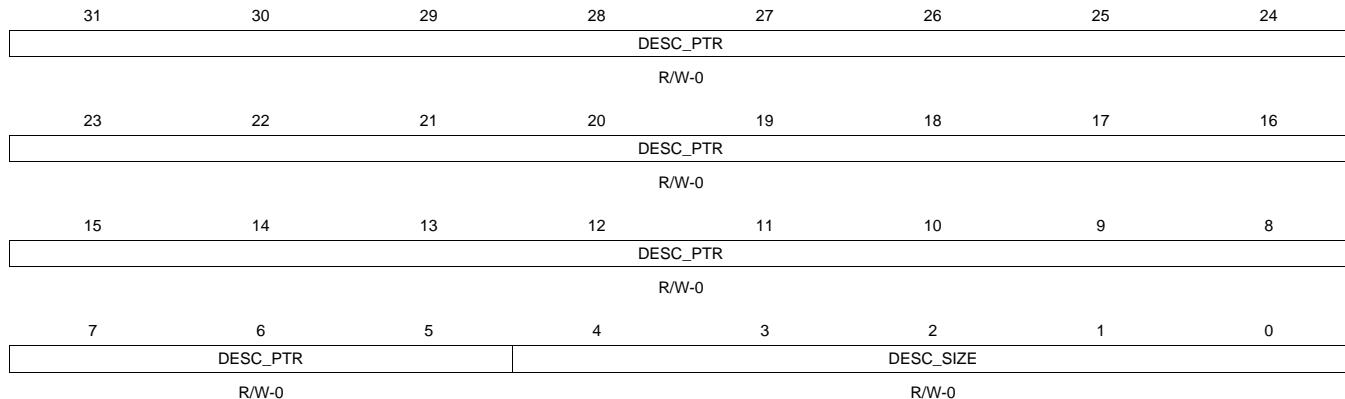
**Table 16-823. QUEUE\_124\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.534 QUEUE\_124\_D Register (offset = 27CCh) [reset = 0h]

QUEUE\_124\_D is shown in [Figure 16-810](#) and described in [Table 16-824](#).

**Figure 16-810. QUEUE\_124\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-824. QUEUE\_124\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.535 QUEUE\_125\_A Register (offset = 27D0h) [reset = 0h]**

QUEUE\_125\_A is shown in [Figure 16-811](#) and described in [Table 16-825](#).

**Figure 16-811. QUEUE\_125\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-825. QUEUE\_125\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.536 QUEUE\_125\_B Register (offset = 27D4h) [reset = 0h]**

QUEUE\_125\_B is shown in [Figure 16-812](#) and described in [Table 16-826](#).

**Figure 16-812. QUEUE\_125\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

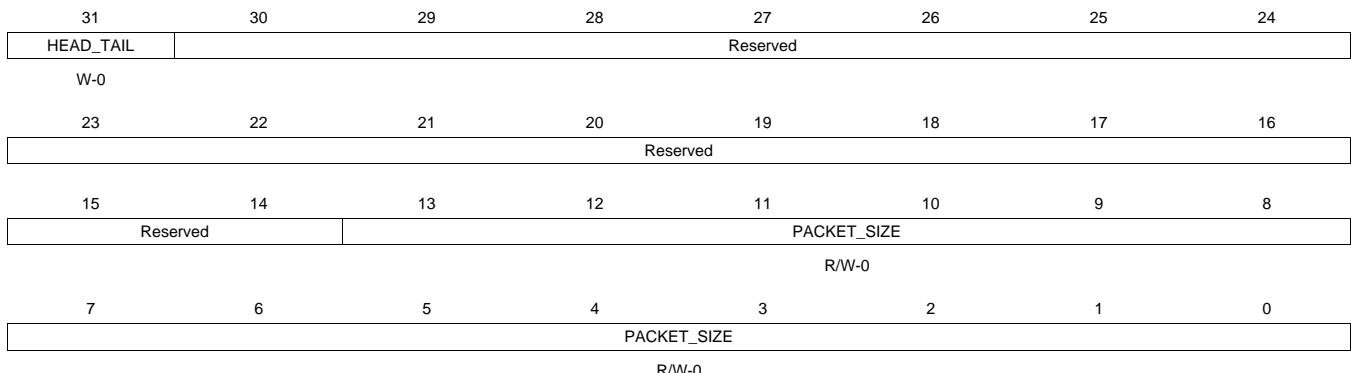
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-826. QUEUE\_125\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.537 QUEUE\_125\_C Register (offset = 27D8h) [reset = 0h]**

 QUEUE\_125\_C is shown in [Figure 16-813](#) and described in [Table 16-827](#).

**Figure 16-813. QUEUE\_125\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

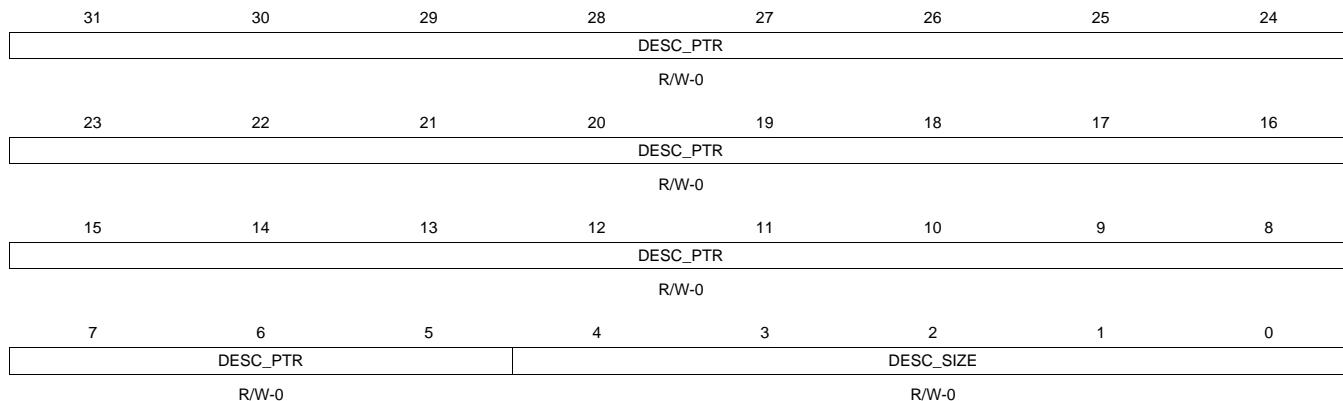
**Table 16-827. QUEUE\_125\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.538 QUEUE\_125\_D Register (offset = 27DCh) [reset = 0h]

QUEUE\_125\_D is shown in [Figure 16-814](#) and described in [Table 16-828](#).

**Figure 16-814. QUEUE\_125\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-828. QUEUE\_125\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.539 QUEUE\_126\_A Register (offset = 27E0h) [reset = 0h]**

QUEUE\_126\_A is shown in [Figure 16-815](#) and described in [Table 16-829](#).

**Figure 16-815. QUEUE\_126\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-829. QUEUE\_126\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.540 QUEUE\_126\_B Register (offset = 27E4h) [reset = 0h]**

QUEUE\_126\_B is shown in [Figure 16-816](#) and described in [Table 16-830](#).

**Figure 16-816. QUEUE\_126\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

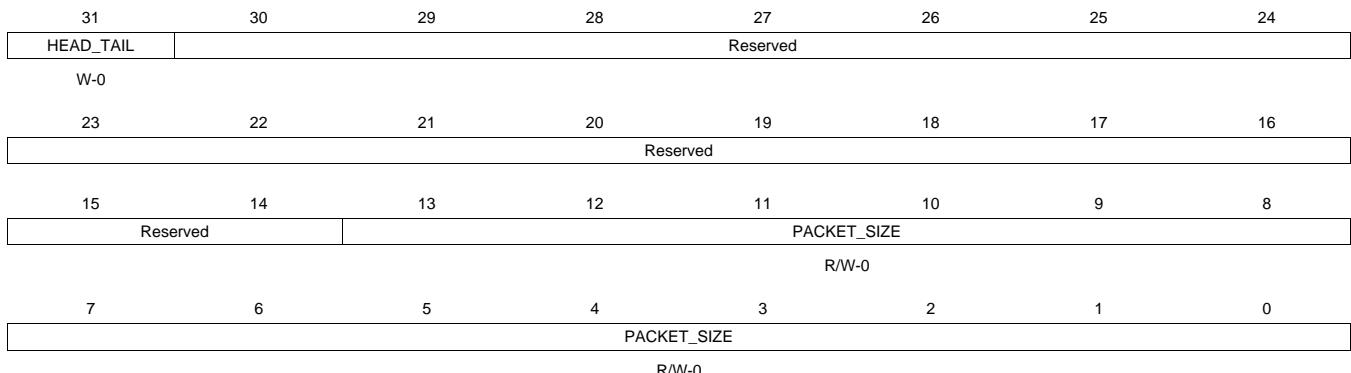
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-830. QUEUE\_126\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.541 QUEUE\_126\_C Register (offset = 27E8h) [reset = 0h]**

QUEUE\_126\_C is shown in [Figure 16-817](#) and described in [Table 16-831](#).

**Figure 16-817. QUEUE\_126\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

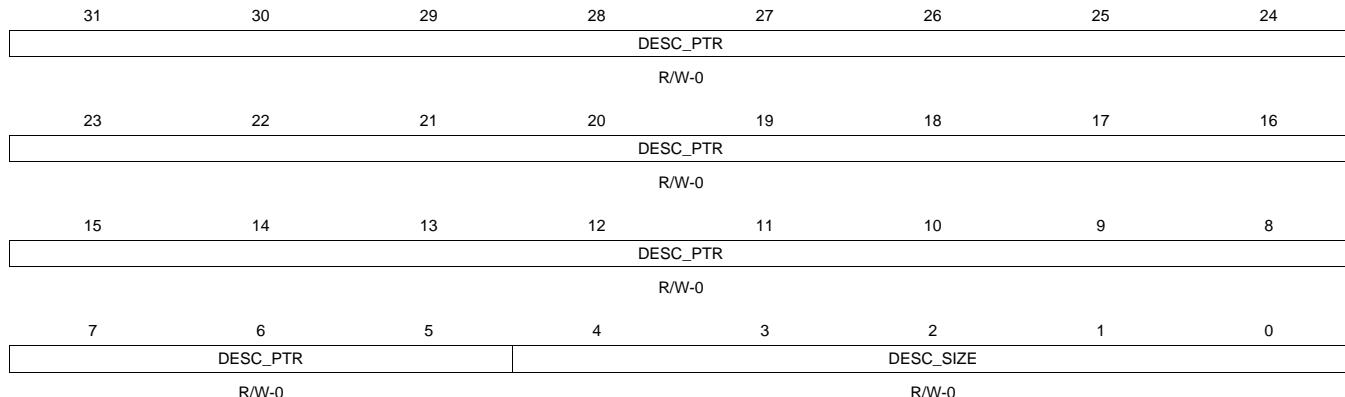
**Table 16-831. QUEUE\_126\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.542 QUEUE\_126\_D Register (offset = 27ECh) [reset = 0h]

QUEUE\_126\_D is shown in [Figure 16-818](#) and described in [Table 16-832](#).

**Figure 16-818. QUEUE\_126\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-832. QUEUE\_126\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.543 QUEUE\_127\_A Register (offset = 27F0h) [reset = 0h]**

QUEUE\_127\_A is shown in [Figure 16-819](#) and described in [Table 16-833](#).

**Figure 16-819. QUEUE\_127\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-833. QUEUE\_127\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.544 QUEUE\_127\_B Register (offset = 27F4h) [reset = 0h]**

QUEUE\_127\_B is shown in [Figure 16-820](#) and described in [Table 16-834](#).

**Figure 16-820. QUEUE\_127\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

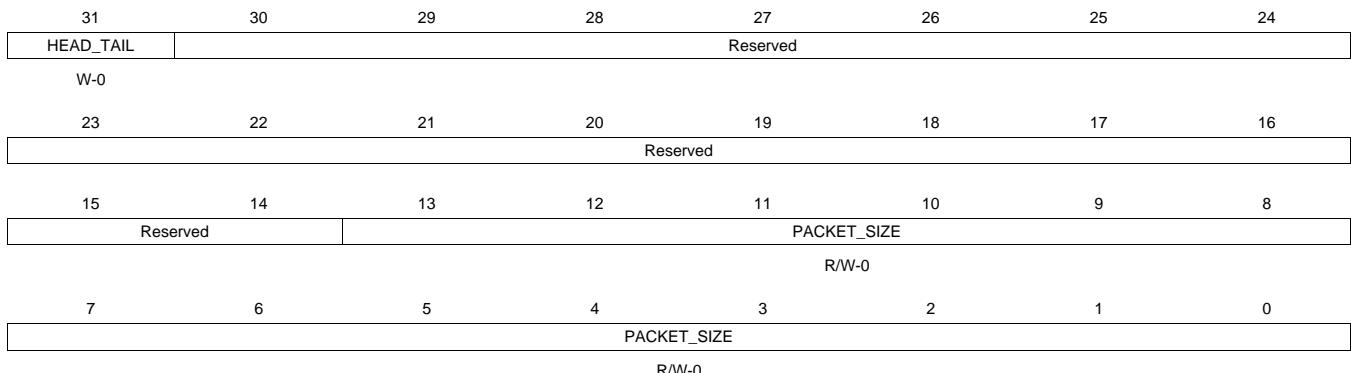
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-834. QUEUE\_127\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.545 QUEUE\_127\_C Register (offset = 27F8h) [reset = 0h]**

QUEUE\_127\_C is shown in [Figure 16-821](#) and described in [Table 16-835](#).

**Figure 16-821. QUEUE\_127\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

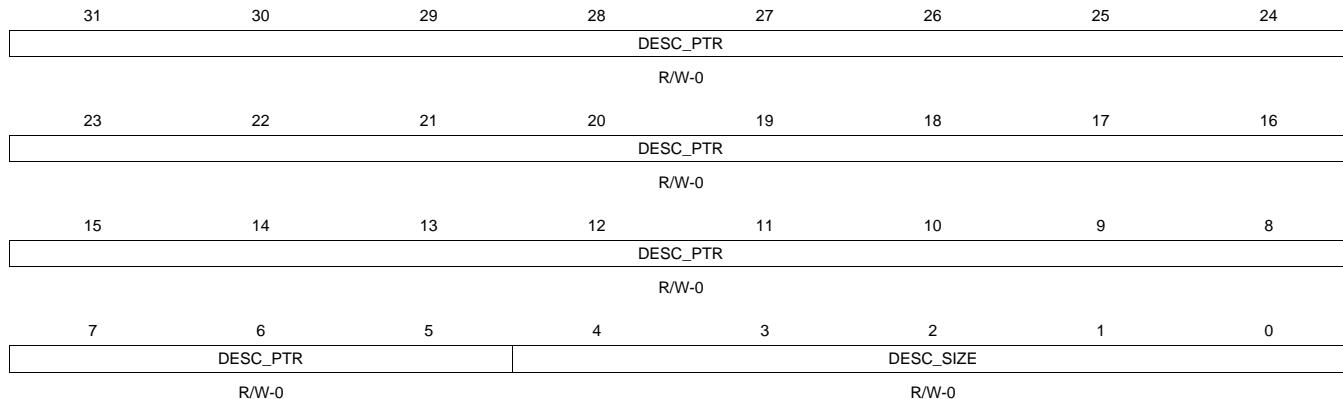
**Table 16-835. QUEUE\_127\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.546 QUEUE\_127\_D Register (offset = 27FCh) [reset = 0h]

QUEUE\_127\_D is shown in [Figure 16-822](#) and described in [Table 16-836](#).

**Figure 16-822. QUEUE\_127\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-836. QUEUE\_127\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.547 QUEUE\_128\_A Register (offset = 2800h) [reset = 0h]**

QUEUE\_128\_A is shown in [Figure 16-823](#) and described in [Table 16-837](#).

**Figure 16-823. QUEUE\_128\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-837. QUEUE\_128\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.548 QUEUE\_128\_B Register (offset = 2804h) [reset = 0h]**

QUEUE\_128\_B is shown in [Figure 16-824](#) and described in [Table 16-838](#).

**Figure 16-824. QUEUE\_128\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

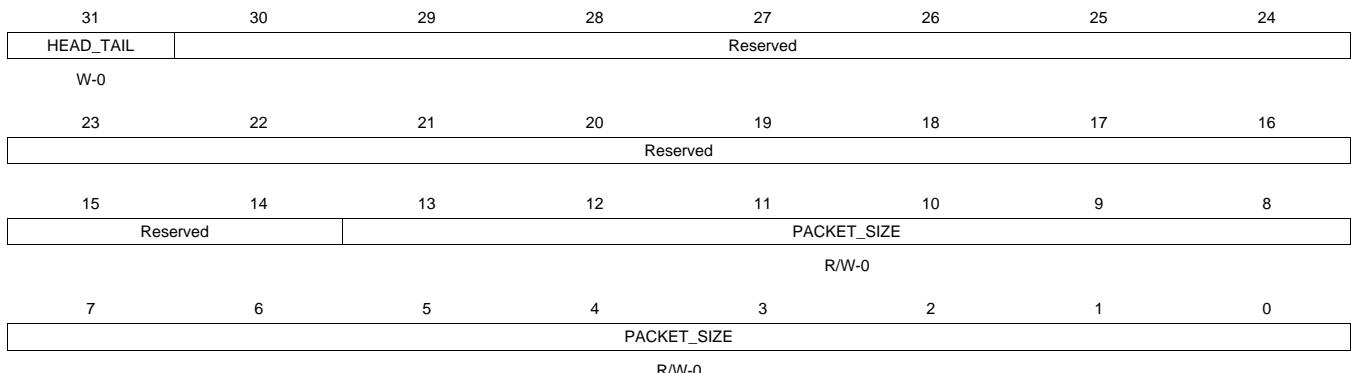
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-838. QUEUE\_128\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.549 QUEUE\_128\_C Register (offset = 2808h) [reset = 0h]**

QUEUE\_128\_C is shown in [Figure 16-825](#) and described in [Table 16-839](#).

**Figure 16-825. QUEUE\_128\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

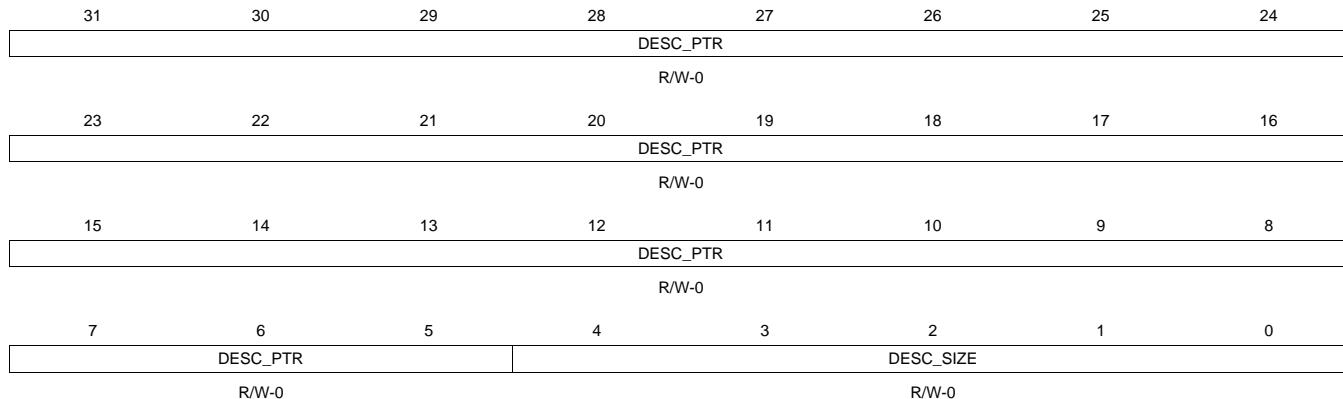
**Table 16-839. QUEUE\_128\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.550 QUEUE\_128\_D Register (offset = 280Ch) [reset = 0h]

QUEUE\_128\_D is shown in [Figure 16-826](#) and described in [Table 16-840](#).

**Figure 16-826. QUEUE\_128\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-840. QUEUE\_128\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.551 QUEUE\_129\_A Register (offset = 2810h) [reset = 0h]**

QUEUE\_129\_A is shown in [Figure 16-827](#) and described in [Table 16-841](#).

**Figure 16-827. QUEUE\_129\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-841. QUEUE\_129\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description   |
|------|-------------------|------|-------|---|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>This count is incremented by 1 whenever a packet is added to the queue.<br>This count is decremented by 1 whenever a packet is popped from the queue. |

**16.5.7.552 QUEUE\_129\_B Register (offset = 2814h) [reset = 0h]**

QUEUE\_129\_B is shown in [Figure 16-828](#) and described in [Table 16-842](#).

**Figure 16-828. QUEUE\_129\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

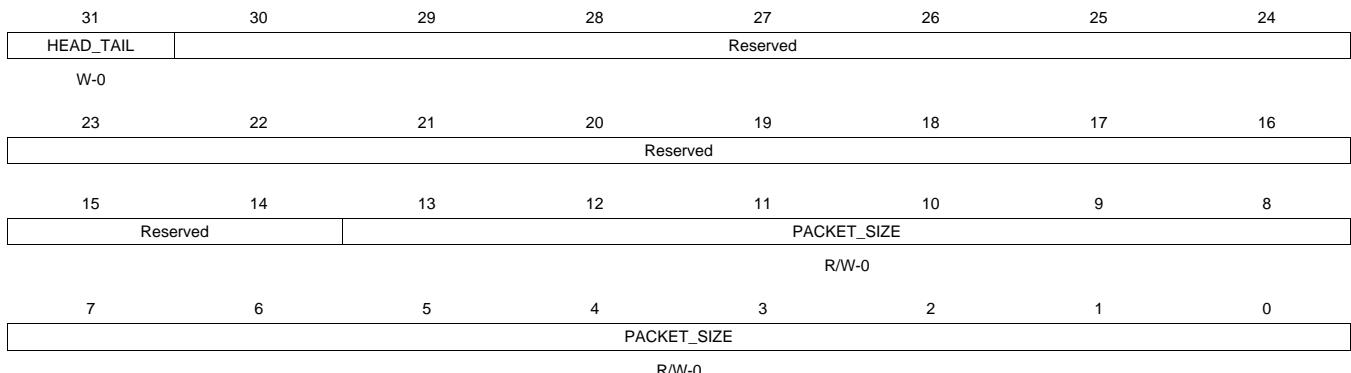
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-842. QUEUE\_129\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.553 QUEUE\_129\_C Register (offset = 2818h) [reset = 0h]**

QUEUE\_129\_C is shown in [Figure 16-829](#) and described in [Table 16-843](#).

**Figure 16-829. QUEUE\_129\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

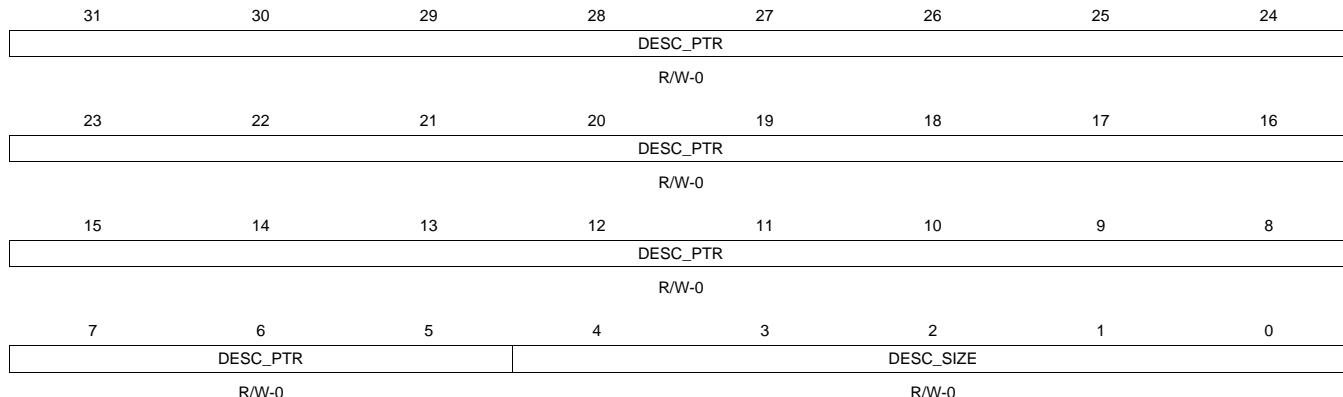
**Table 16-843. QUEUE\_129\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.554 QUEUE\_129\_D Register (offset = 281Ch) [reset = 0h]

QUEUE\_129\_D is shown in [Figure 16-830](#) and described in [Table 16-844](#).

**Figure 16-830. QUEUE\_129\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-844. QUEUE\_129\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.555 QUEUE\_130\_A Register (offset = 2820h) [reset = 0h]**

QUEUE\_130\_A is shown in [Figure 16-831](#) and described in [Table 16-845](#).

**Figure 16-831. QUEUE\_130\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-845. QUEUE\_130\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.556 QUEUE\_130\_B Register (offset = 2824h) [reset = 0h]**

QUEUE\_130\_B is shown in [Figure 16-832](#) and described in [Table 16-846](#).

**Figure 16-832. QUEUE\_130\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

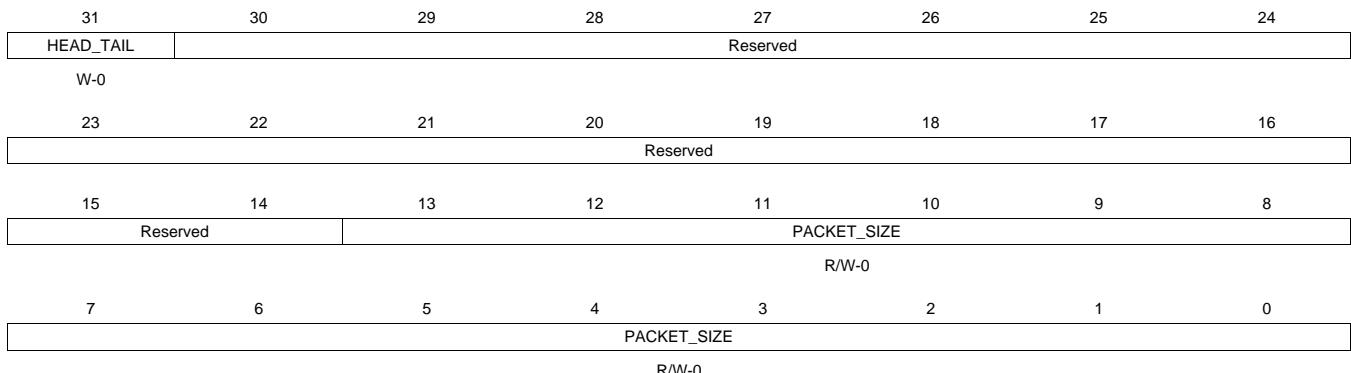
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-846. QUEUE\_130\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.557 QUEUE\_130\_C Register (offset = 2828h) [reset = 0h]**

 QUEUE\_130\_C is shown in [Figure 16-833](#) and described in [Table 16-847](#).

**Figure 16-833. QUEUE\_130\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

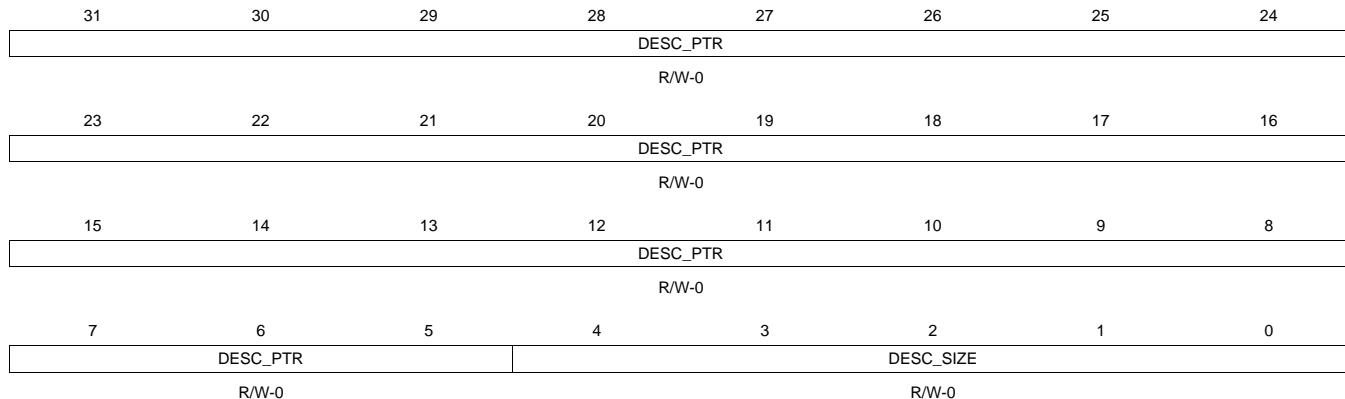
**Table 16-847. QUEUE\_130\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.558 QUEUE\_130\_D Register (offset = 282Ch) [reset = 0h]

QUEUE\_130\_D is shown in [Figure 16-834](#) and described in [Table 16-848](#).

**Figure 16-834. QUEUE\_130\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-848. QUEUE\_130\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.559 QUEUE\_131\_A Register (offset = 2830h) [reset = 0h]**

QUEUE\_131\_A is shown in [Figure 16-835](#) and described in [Table 16-849](#).

**Figure 16-835. QUEUE\_131\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-849. QUEUE\_131\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.560 QUEUE\_131\_B Register (offset = 2834h) [reset = 0h]**

QUEUE\_131\_B is shown in [Figure 16-836](#) and described in [Table 16-850](#).

**Figure 16-836. QUEUE\_131\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

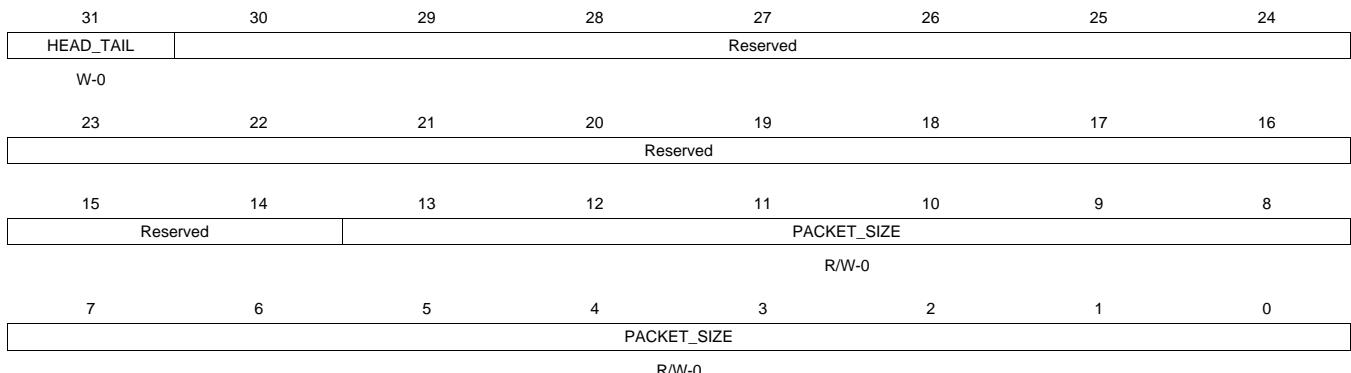
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-850. QUEUE\_131\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.561 QUEUE\_131\_C Register (offset = 2838h) [reset = 0h]**

QUEUE\_131\_C is shown in [Figure 16-837](#) and described in [Table 16-851](#).

**Figure 16-837. QUEUE\_131\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

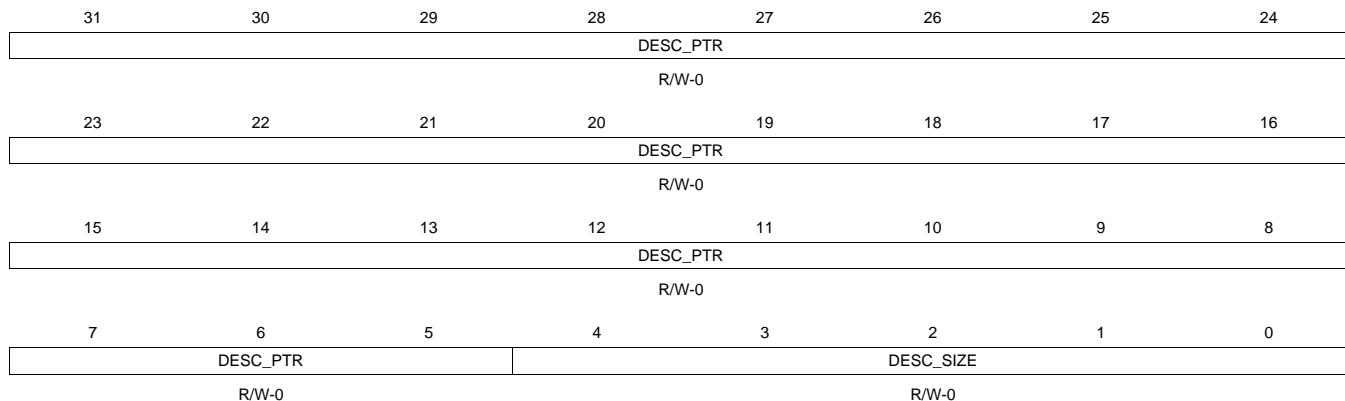
**Table 16-851. QUEUE\_131\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.562 QUEUE\_131\_D Register (offset = 283Ch) [reset = 0h]

QUEUE\_131\_D is shown in [Figure 16-838](#) and described in [Table 16-852](#).

**Figure 16-838. QUEUE\_131\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-852. QUEUE\_131\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.563 QUEUE\_132\_A Register (offset = 2840h) [reset = 0h]**

QUEUE\_132\_A is shown in [Figure 16-839](#) and described in [Table 16-853](#).

**Figure 16-839. QUEUE\_132\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-853. QUEUE\_132\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

### 16.5.7.564 QUEUE\_132\_B Register (offset = 2844h) [reset = 0h]

QUEUE\_132\_B is shown in [Figure 16-840](#) and described in [Table 16-854](#).

**Figure 16-840. QUEUE\_132\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

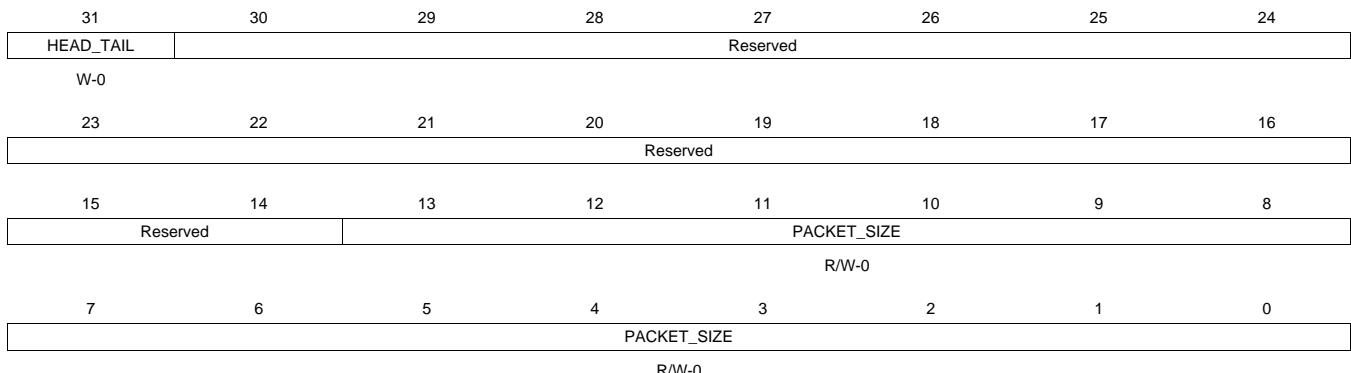
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-854. QUEUE\_132\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.565 QUEUE\_132\_C Register (offset = 2848h) [reset = 0h]**

QUEUE\_132\_C is shown in [Figure 16-841](#) and described in [Table 16-855](#).

**Figure 16-841. QUEUE\_132\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

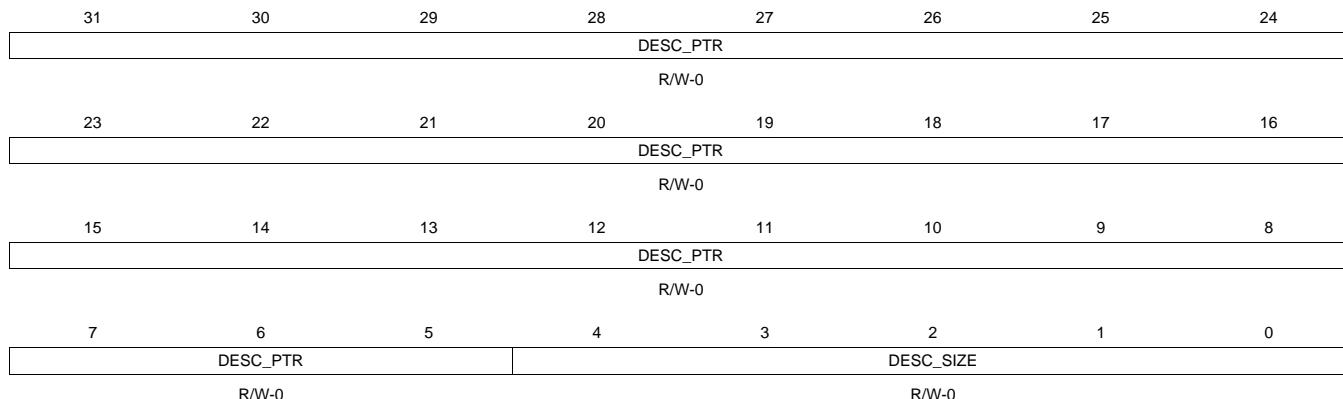
**Table 16-855. QUEUE\_132\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.566 QUEUE\_132\_D Register (offset = 284Ch) [reset = 0h]

QUEUE\_132\_D is shown in [Figure 16-842](#) and described in [Table 16-856](#).

**Figure 16-842. QUEUE\_132\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-856. QUEUE\_132\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.567 QUEUE\_133\_A Register (offset = 2850h) [reset = 0h]**

QUEUE\_133\_A is shown in [Figure 16-843](#) and described in [Table 16-857](#).

**Figure 16-843. QUEUE\_133\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-857. QUEUE\_133\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.568 QUEUE\_133\_B Register (offset = 2854h) [reset = 0h]**

QUEUE\_133\_B is shown in [Figure 16-844](#) and described in [Table 16-858](#).

**Figure 16-844. QUEUE\_133\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

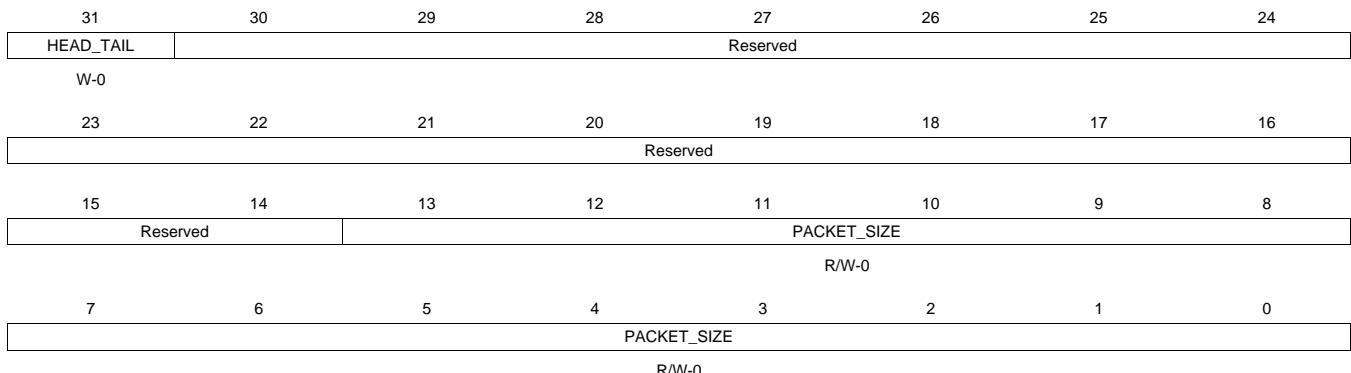
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-858. QUEUE\_133\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.569 QUEUE\_133\_C Register (offset = 2858h) [reset = 0h]**

QUEUE\_133\_C is shown in [Figure 16-845](#) and described in [Table 16-859](#).

**Figure 16-845. QUEUE\_133\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

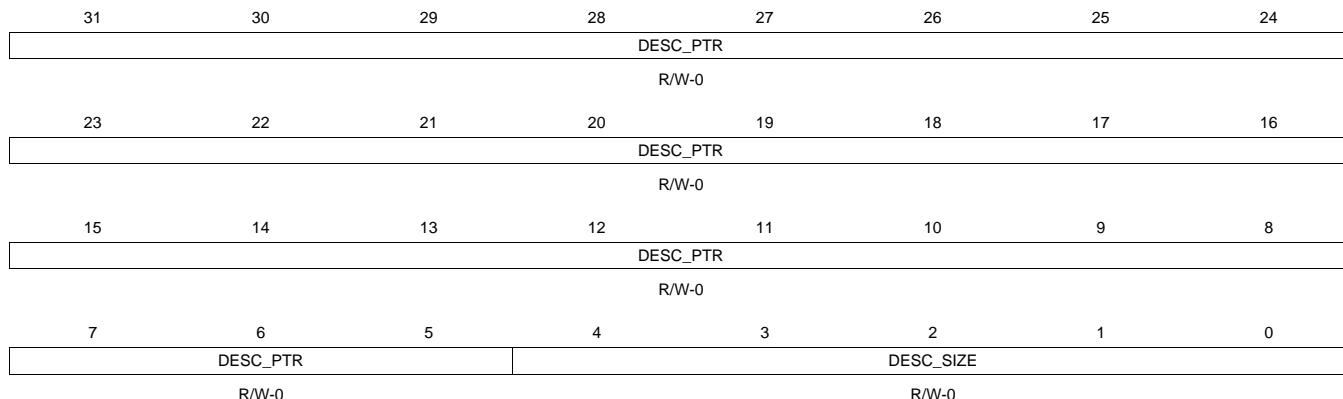
**Table 16-859. QUEUE\_133\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.570 QUEUE\_133\_D Register (offset = 285Ch) [reset = 0h]

QUEUE\_133\_D is shown in [Figure 16-846](#) and described in [Table 16-860](#).

**Figure 16-846. QUEUE\_133\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-860. QUEUE\_133\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.571 QUEUE\_134\_A Register (offset = 2860h) [reset = 0h]**

QUEUE\_134\_A is shown in [Figure 16-847](#) and described in [Table 16-861](#).

**Figure 16-847. QUEUE\_134\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-861. QUEUE\_134\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.572 QUEUE\_134\_B Register (offset = 2864h) [reset = 0h]**

QUEUE\_134\_B is shown in [Figure 16-848](#) and described in [Table 16-862](#).

**Figure 16-848. QUEUE\_134\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

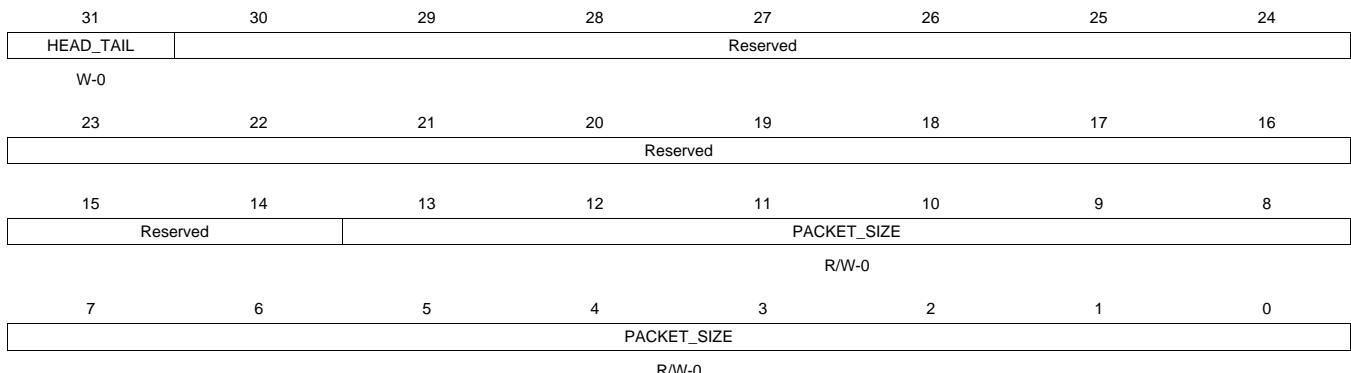
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-862. QUEUE\_134\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.573 QUEUE\_134\_C Register (offset = 2868h) [reset = 0h]**

QUEUE\_134\_C is shown in [Figure 16-849](#) and described in [Table 16-863](#).

**Figure 16-849. QUEUE\_134\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

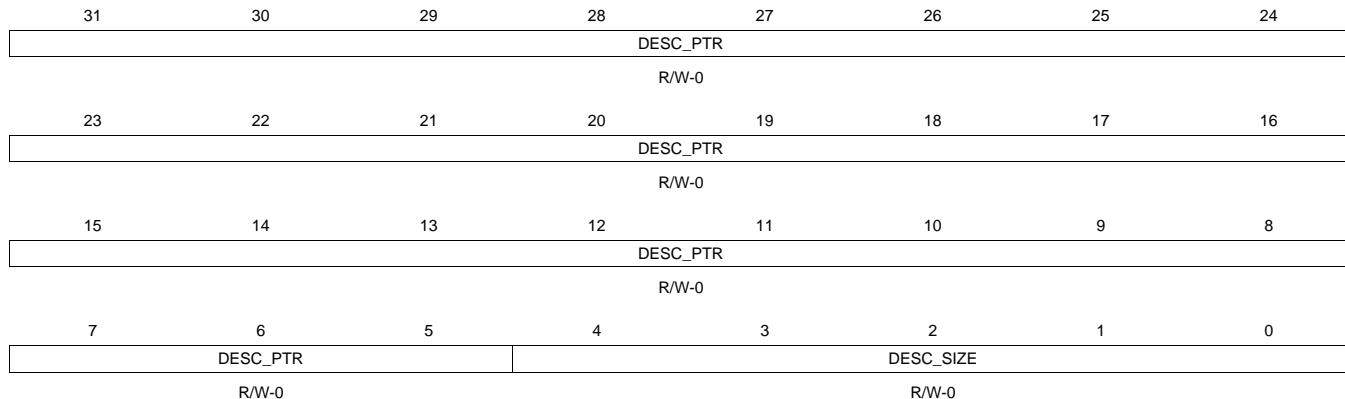
**Table 16-863. QUEUE\_134\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.574 QUEUE\_134\_D Register (offset = 286Ch) [reset = 0h]

QUEUE\_134\_D is shown in [Figure 16-850](#) and described in [Table 16-864](#).

**Figure 16-850. QUEUE\_134\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-864. QUEUE\_134\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.575 QUEUE\_135\_A Register (offset = 2870h) [reset = 0h]**

QUEUE\_135\_A is shown in [Figure 16-851](#) and described in [Table 16-865](#).

**Figure 16-851. QUEUE\_135\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-865. QUEUE\_135\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

### 16.5.7.576 QUEUE\_135\_B Register (offset = 2874h) [reset = 0h]

QUEUE\_135\_B is shown in [Figure 16-852](#) and described in [Table 16-866](#).

**Figure 16-852. QUEUE\_135\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

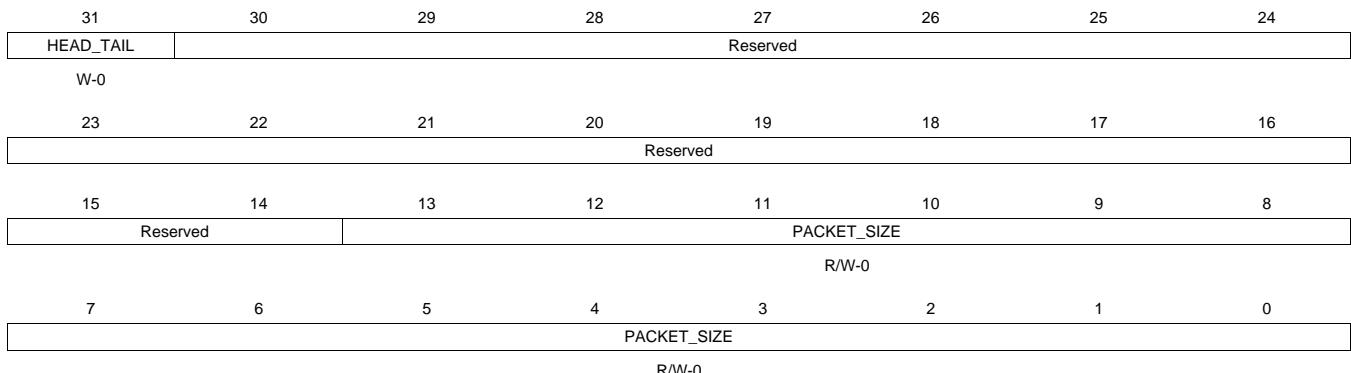
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-866. QUEUE\_135\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.577 QUEUE\_135\_C Register (offset = 2878h) [reset = 0h]**

 QUEUE\_135\_C is shown in [Figure 16-853](#) and described in [Table 16-867](#).

**Figure 16-853. QUEUE\_135\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

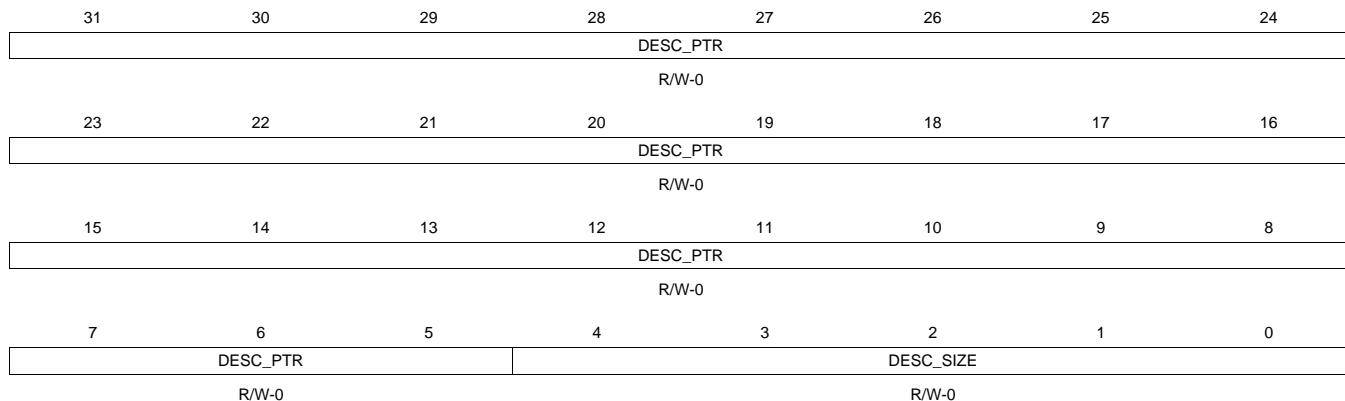
**Table 16-867. QUEUE\_135\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.578 QUEUE\_135\_D Register (offset = 287Ch) [reset = 0h]

QUEUE\_135\_D is shown in [Figure 16-854](#) and described in [Table 16-868](#).

**Figure 16-854. QUEUE\_135\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-868. QUEUE\_135\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.579 QUEUE\_136\_A Register (offset = 2880h) [reset = 0h]**

QUEUE\_136\_A is shown in [Figure 16-855](#) and described in [Table 16-869](#).

**Figure 16-855. QUEUE\_136\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-869. QUEUE\_136\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.580 QUEUE\_136\_B Register (offset = 2884h) [reset = 0h]**

QUEUE\_136\_B is shown in [Figure 16-856](#) and described in [Table 16-870](#).

**Figure 16-856. QUEUE\_136\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

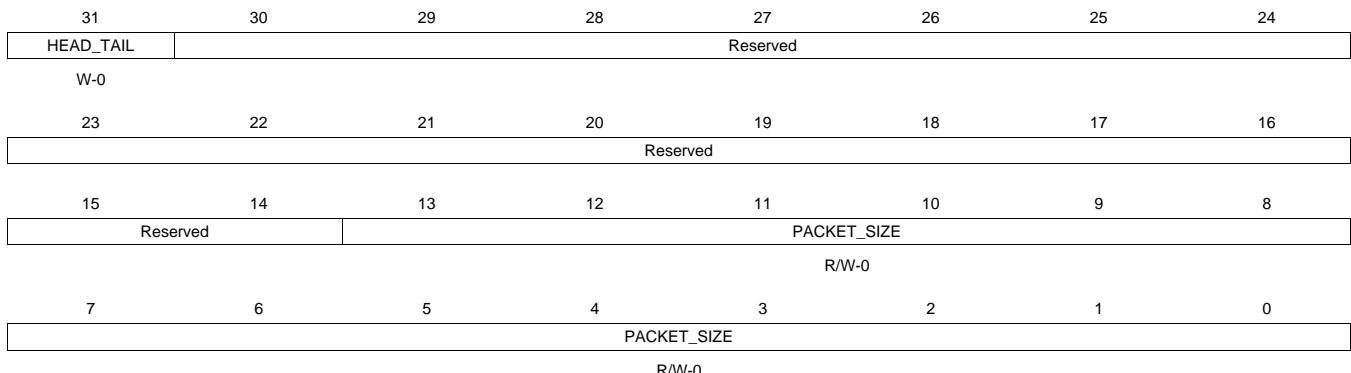
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-870. QUEUE\_136\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.581 QUEUE\_136\_C Register (offset = 2888h) [reset = 0h]**

 QUEUE\_136\_C is shown in [Figure 16-857](#) and described in [Table 16-871](#).

**Figure 16-857. QUEUE\_136\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

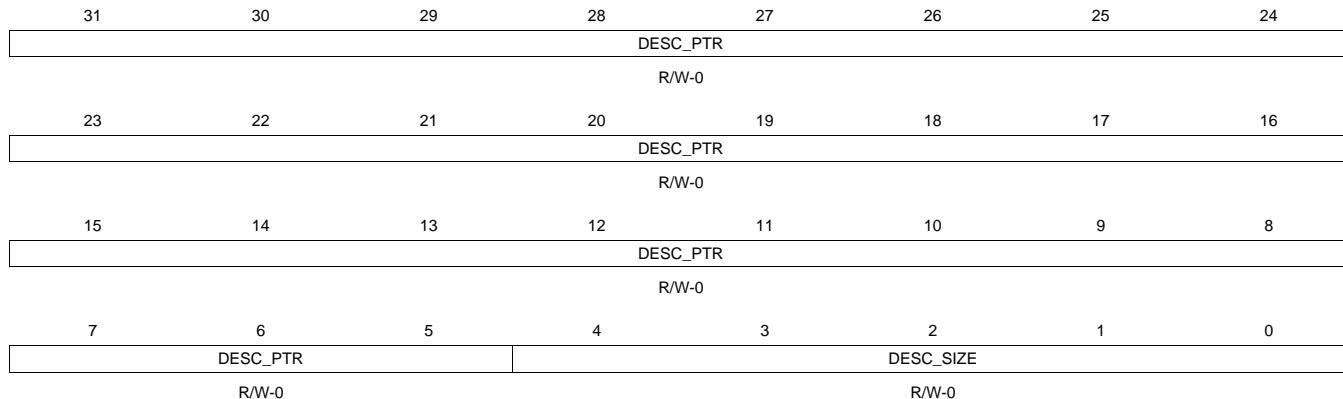
**Table 16-871. QUEUE\_136\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.582 QUEUE\_136\_D Register (offset = 288Ch) [reset = 0h]

QUEUE\_136\_D is shown in [Figure 16-858](#) and described in [Table 16-872](#).

**Figure 16-858. QUEUE\_136\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-872. QUEUE\_136\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.583 QUEUE\_137\_A Register (offset = 2890h) [reset = 0h]**

QUEUE\_137\_A is shown in [Figure 16-859](#) and described in [Table 16-873](#).

**Figure 16-859. QUEUE\_137\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-873. QUEUE\_137\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.584 QUEUE\_137\_B Register (offset = 2894h) [reset = 0h]**

QUEUE\_137\_B is shown in [Figure 16-860](#) and described in [Table 16-874](#).

**Figure 16-860. QUEUE\_137\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

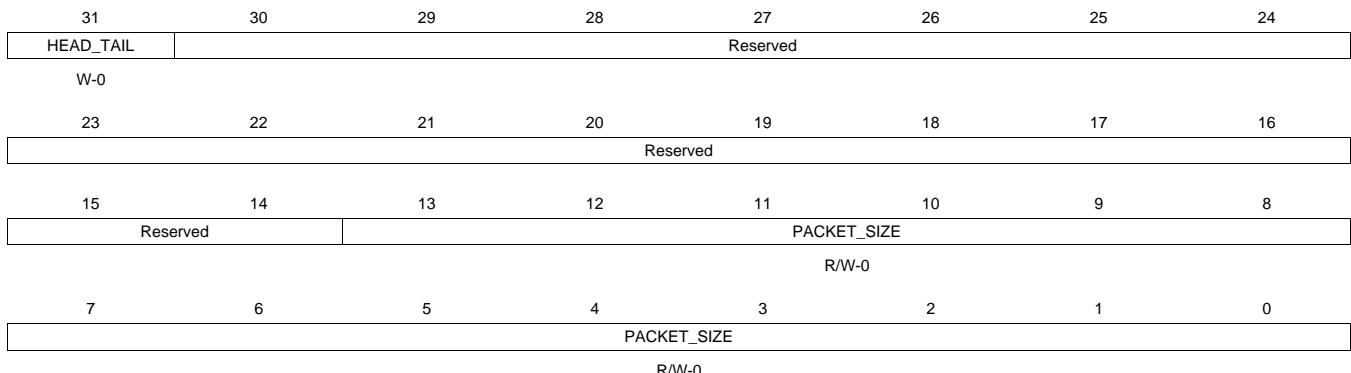
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-874. QUEUE\_137\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.585 QUEUE\_137\_C Register (offset = 2898h) [reset = 0h]**

QUEUE\_137\_C is shown in [Figure 16-861](#) and described in [Table 16-875](#).

**Figure 16-861. QUEUE\_137\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

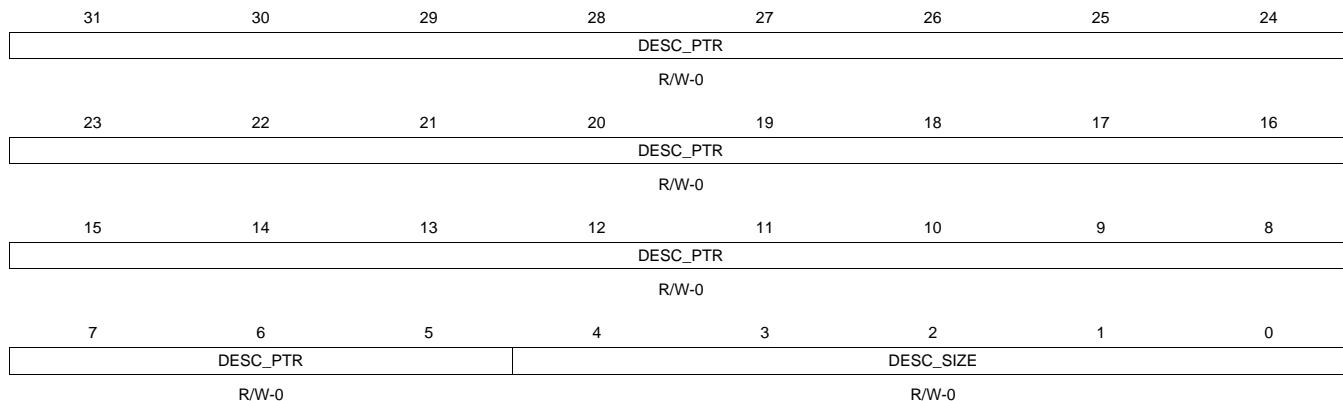
**Table 16-875. QUEUE\_137\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.586 QUEUE\_137\_D Register (offset = 289Ch) [reset = 0h]

QUEUE\_137\_D is shown in [Figure 16-862](#) and described in [Table 16-876](#).

**Figure 16-862. QUEUE\_137\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-876. QUEUE\_137\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.587 QUEUE\_138\_A Register (offset = 28A0h) [reset = 0h]**

QUEUE\_138\_A is shown in [Figure 16-863](#) and described in [Table 16-877](#).

**Figure 16-863. QUEUE\_138\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-877. QUEUE\_138\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.588 QUEUE\_138\_B Register (offset = 28A4h) [reset = 0h]**

QUEUE\_138\_B is shown in [Figure 16-864](#) and described in [Table 16-878](#).

**Figure 16-864. QUEUE\_138\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

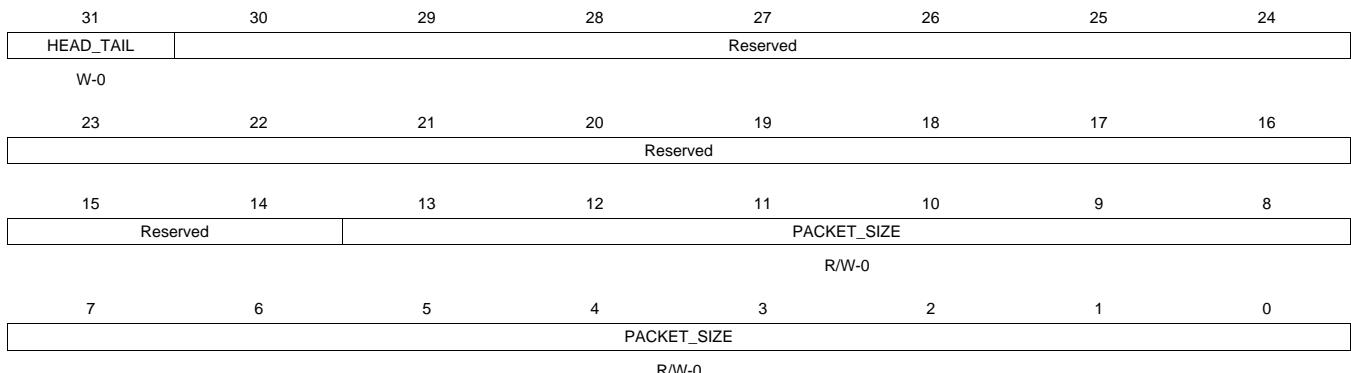
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-878. QUEUE\_138\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.589 QUEUE\_138\_C Register (offset = 28A8h) [reset = 0h]**

QUEUE\_138\_C is shown in [Figure 16-865](#) and described in [Table 16-879](#).

**Figure 16-865. QUEUE\_138\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

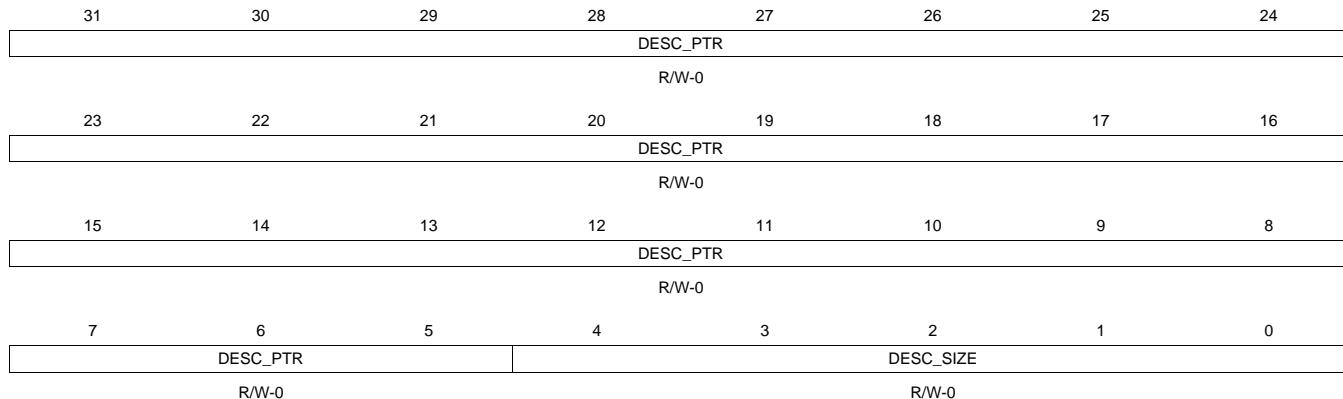
**Table 16-879. QUEUE\_138\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.590 QUEUE\_138\_D Register (offset = 28ACh) [reset = 0h]

QUEUE\_138\_D is shown in [Figure 16-866](#) and described in [Table 16-880](#).

**Figure 16-866. QUEUE\_138\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-880. QUEUE\_138\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.591 QUEUE\_139\_A Register (offset = 28B0h) [reset = 0h]**

QUEUE\_139\_A is shown in [Figure 16-867](#) and described in [Table 16-881](#).

**Figure 16-867. QUEUE\_139\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-881. QUEUE\_139\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.592 QUEUE\_139\_B Register (offset = 28B4h) [reset = 0h]**

QUEUE\_139\_B is shown in [Figure 16-868](#) and described in [Table 16-882](#).

**Figure 16-868. QUEUE\_139\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

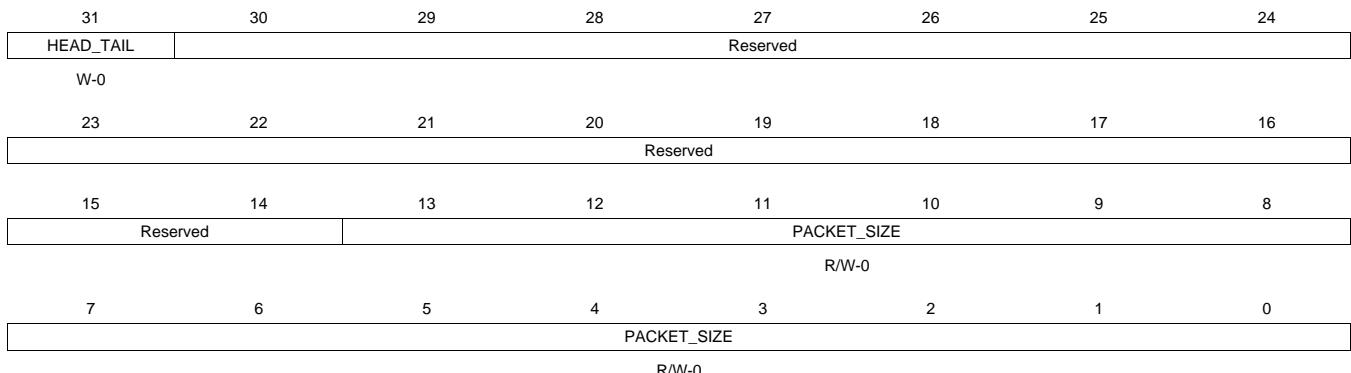
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-882. QUEUE\_139\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.593 QUEUE\_139\_C Register (offset = 28B8h) [reset = 0h]**

QUEUE\_139\_C is shown in [Figure 16-869](#) and described in [Table 16-883](#).

**Figure 16-869. QUEUE\_139\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

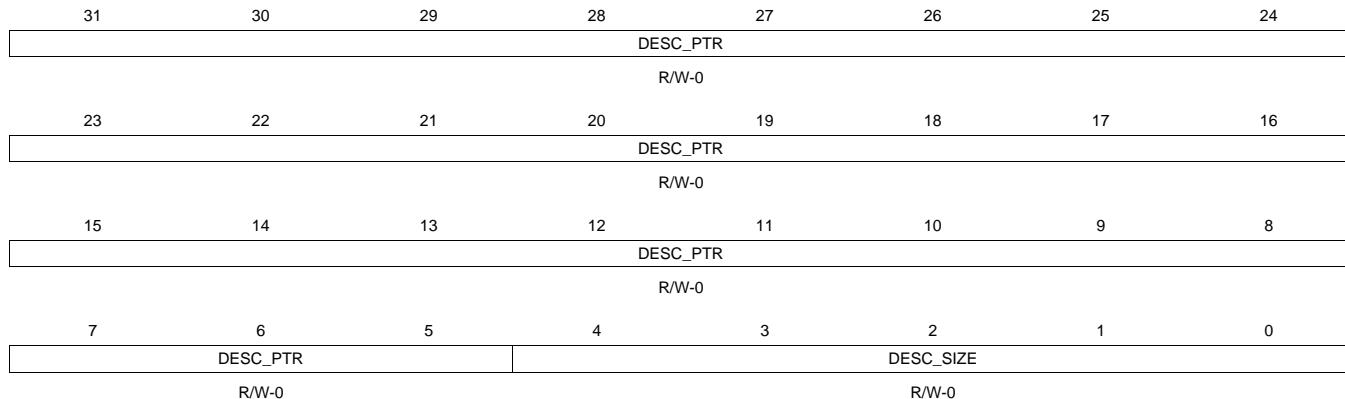
**Table 16-883. QUEUE\_139\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.594 QUEUE\_139\_D Register (offset = 28BCh) [reset = 0h]

QUEUE\_139\_D is shown in [Figure 16-870](#) and described in [Table 16-884](#).

**Figure 16-870. QUEUE\_139\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-884. QUEUE\_139\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.595 QUEUE\_140\_A Register (offset = 28C0h) [reset = 0h]**

QUEUE\_140\_A is shown in [Figure 16-871](#) and described in [Table 16-885](#).

**Figure 16-871. QUEUE\_140\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-885. QUEUE\_140\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.596 QUEUE\_140\_B Register (offset = 28C4h) [reset = 0h]**

QUEUE\_140\_B is shown in [Figure 16-872](#) and described in [Table 16-886](#).

**Figure 16-872. QUEUE\_140\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-886. QUEUE\_140\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.597 QUEUE\_140\_C Register (offset = 28C8h) [reset = 0h]**

 QUEUE\_140\_C is shown in [Figure 16-873](#) and described in [Table 16-887](#).

**Figure 16-873. QUEUE\_140\_C Register**

| 31        | 30 | 29 | 28 | 27          | 26 | 25 | 24 |
|-----------|----|----|----|-------------|----|----|----|
| HEAD_TAIL |    |    |    | Reserved    |    |    |    |
| W-0       |    |    |    |             |    |    |    |
| 23        | 22 | 21 | 20 | 19          | 18 | 17 | 16 |
|           |    |    |    | Reserved    |    |    |    |
| 15        | 14 | 13 | 12 | 11          | 10 | 9  | 8  |
| Reserved  |    |    |    | PACKET_SIZE |    |    |    |
|           |    |    |    | R/W-0       |    |    |    |
| 7         | 6  | 5  | 4  | 3           | 2  | 1  | 0  |
|           |    |    |    | PACKET_SIZE |    |    |    |
|           |    |    |    | R/W-0       |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

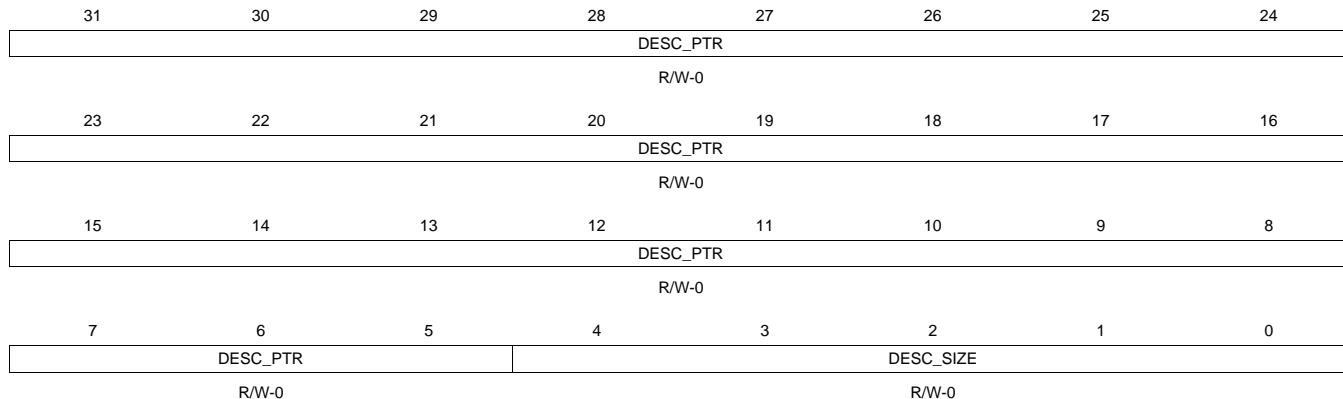
**Table 16-887. QUEUE\_140\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.598 QUEUE\_140\_D Register (offset = 28CCh) [reset = 0h]

QUEUE\_140\_D is shown in [Figure 16-874](#) and described in [Table 16-888](#).

**Figure 16-874. QUEUE\_140\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-888. QUEUE\_140\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.599 QUEUE\_141\_A Register (offset = 28D0h) [reset = 0h]**

QUEUE\_141\_A is shown in [Figure 16-875](#) and described in [Table 16-889](#).

**Figure 16-875. QUEUE\_141\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-889. QUEUE\_141\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.600 QUEUE\_141\_B Register (offset = 28D4h) [reset = 0h]**

QUEUE\_141\_B is shown in [Figure 16-876](#) and described in [Table 16-890](#).

**Figure 16-876. QUEUE\_141\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

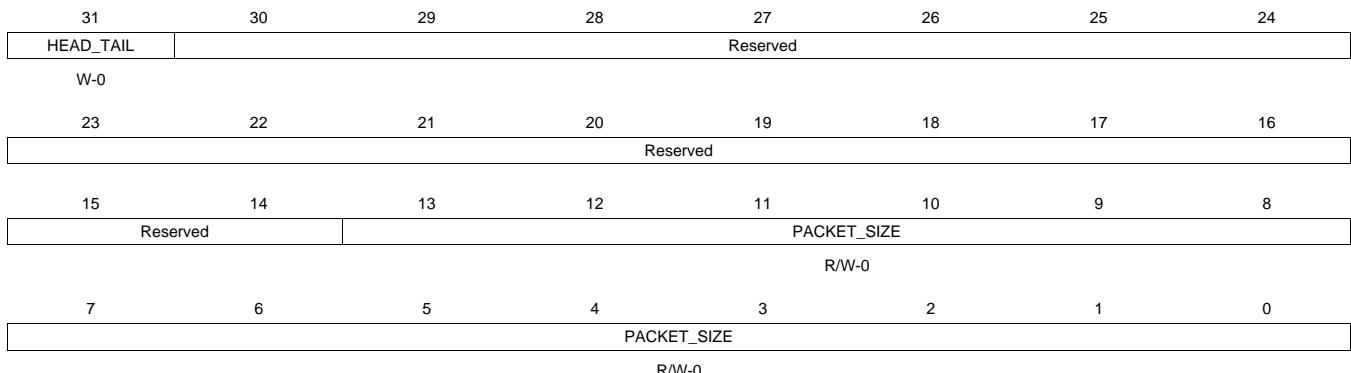
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-890. QUEUE\_141\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.601 QUEUE\_141\_C Register (offset = 28D8h) [reset = 0h]**

QUEUE\_141\_C is shown in [Figure 16-877](#) and described in [Table 16-891](#).

**Figure 16-877. QUEUE\_141\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

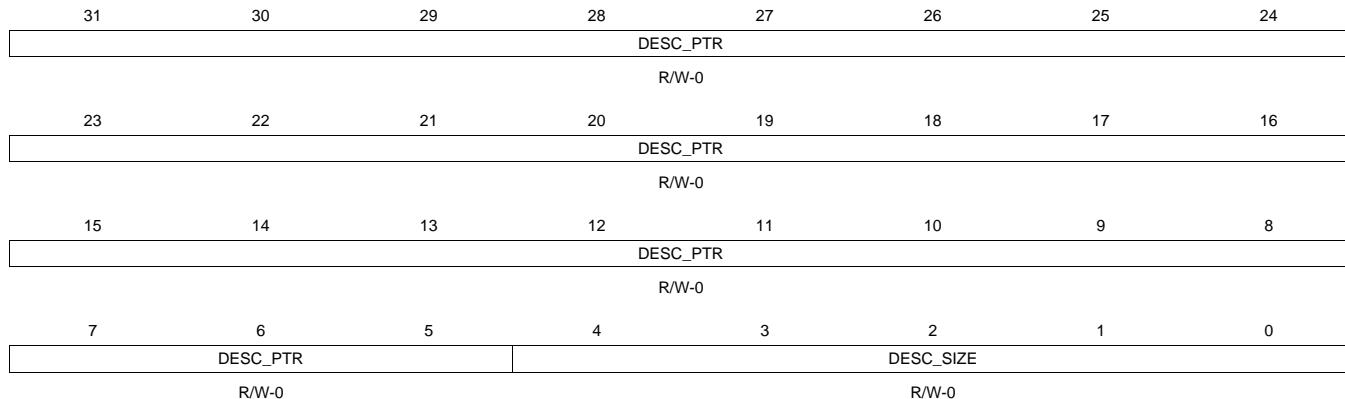
**Table 16-891. QUEUE\_141\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.602 QUEUE\_141\_D Register (offset = 28DCh) [reset = 0h]

QUEUE\_141\_D is shown in [Figure 16-878](#) and described in [Table 16-892](#).

**Figure 16-878. QUEUE\_141\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-892. QUEUE\_141\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.603 QUEUE\_142\_A Register (offset = 28E0h) [reset = 0h]**

QUEUE\_142\_A is shown in [Figure 16-879](#) and described in [Table 16-893](#).

**Figure 16-879. QUEUE\_142\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-893. QUEUE\_142\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.604 QUEUE\_142\_B Register (offset = 28E4h) [reset = 0h]**

QUEUE\_142\_B is shown in [Figure 16-880](#) and described in [Table 16-894](#).

**Figure 16-880. QUEUE\_142\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

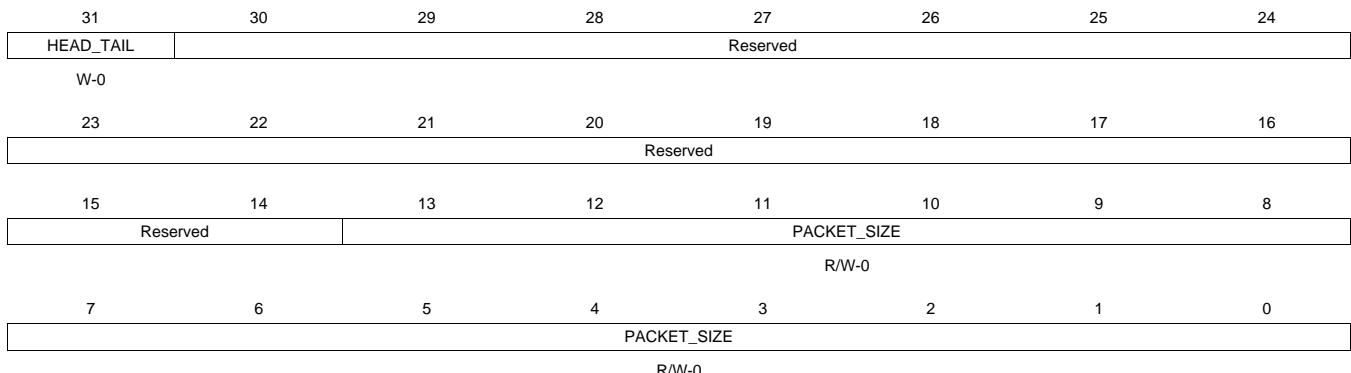
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-894. QUEUE\_142\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.605 QUEUE\_142\_C Register (offset = 28E8h) [reset = 0h]**

QUEUE\_142\_C is shown in [Figure 16-881](#) and described in [Table 16-895](#).

**Figure 16-881. QUEUE\_142\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

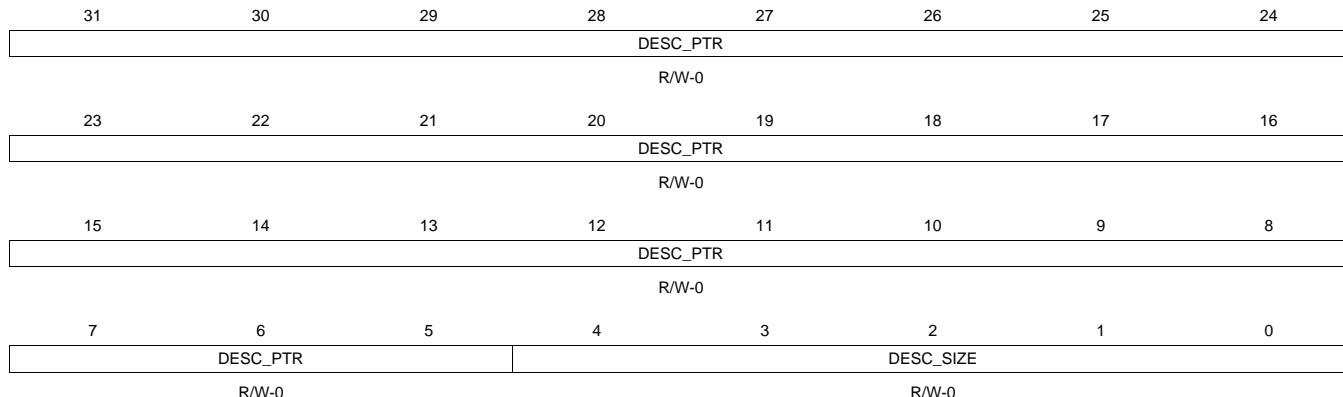
**Table 16-895. QUEUE\_142\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.606 QUEUE\_142\_D Register (offset = 28ECh) [reset = 0h]

QUEUE\_142\_D is shown in [Figure 16-882](#) and described in [Table 16-896](#).

**Figure 16-882. QUEUE\_142\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-896. QUEUE\_142\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.607 QUEUE\_143\_A Register (offset = 28F0h) [reset = 0h]**

QUEUE\_143\_A is shown in [Figure 16-883](#) and described in [Table 16-897](#).

**Figure 16-883. QUEUE\_143\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-897. QUEUE\_143\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.608 QUEUE\_143\_B Register (offset = 28F4h) [reset = 0h]**

QUEUE\_143\_B is shown in [Figure 16-884](#) and described in [Table 16-898](#).

**Figure 16-884. QUEUE\_143\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

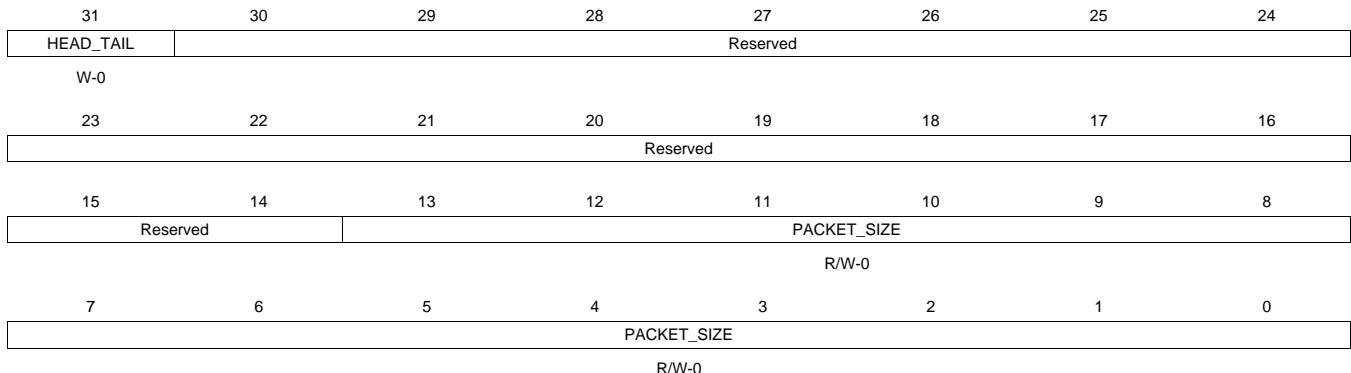
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-898. QUEUE\_143\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.609 QUEUE\_143\_C Register (offset = 28F8h) [reset = 0h]**

QUEUE\_143\_C is shown in [Figure 16-885](#) and described in [Table 16-899](#).

**Figure 16-885. QUEUE\_143\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

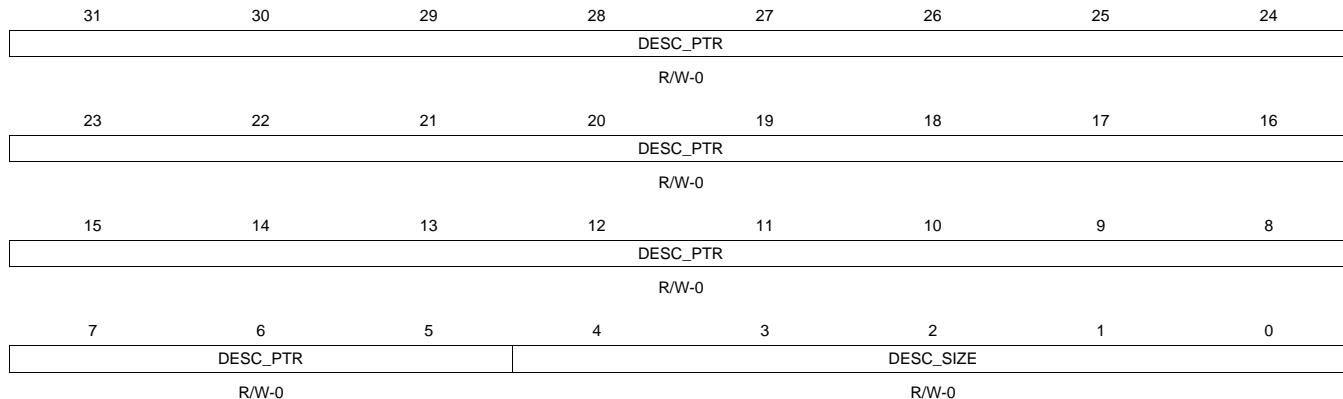
**Table 16-899. QUEUE\_143\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.610 QUEUE\_143\_D Register (offset = 28FCh) [reset = 0h]

QUEUE\_143\_D is shown in [Figure 16-886](#) and described in [Table 16-900](#).

**Figure 16-886. QUEUE\_143\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-900. QUEUE\_143\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.611 QUEUE\_144\_A Register (offset = 2900h) [reset = 0h]**

QUEUE\_144\_A is shown in [Figure 16-887](#) and described in [Table 16-901](#).

**Figure 16-887. QUEUE\_144\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-901. QUEUE\_144\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.612 QUEUE\_144\_B Register (offset = 2904h) [reset = 0h]**

QUEUE\_144\_B is shown in [Figure 16-888](#) and described in [Table 16-902](#).

**Figure 16-888. QUEUE\_144\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

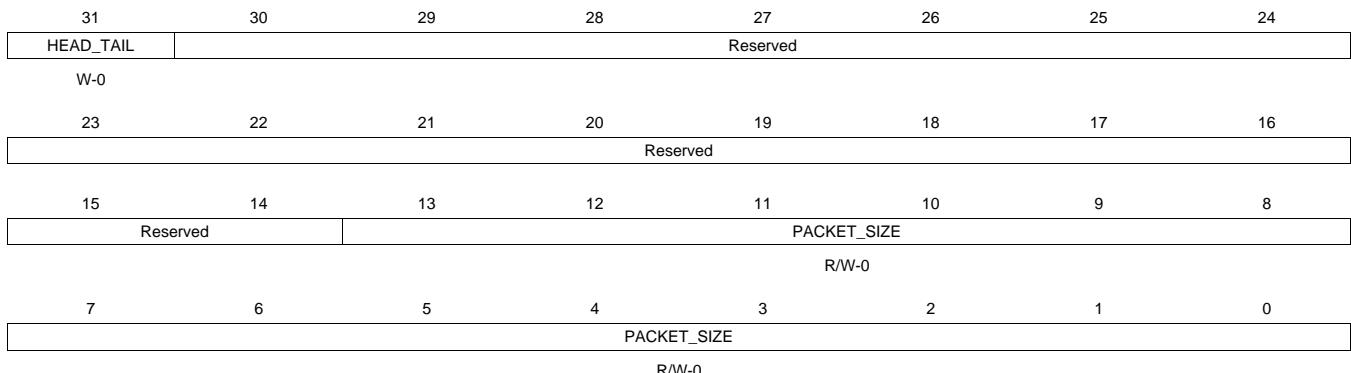
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-902. QUEUE\_144\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.613 QUEUE\_144\_C Register (offset = 2908h) [reset = 0h]**

 QUEUE\_144\_C is shown in [Figure 16-889](#) and described in [Table 16-903](#).

**Figure 16-889. QUEUE\_144\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

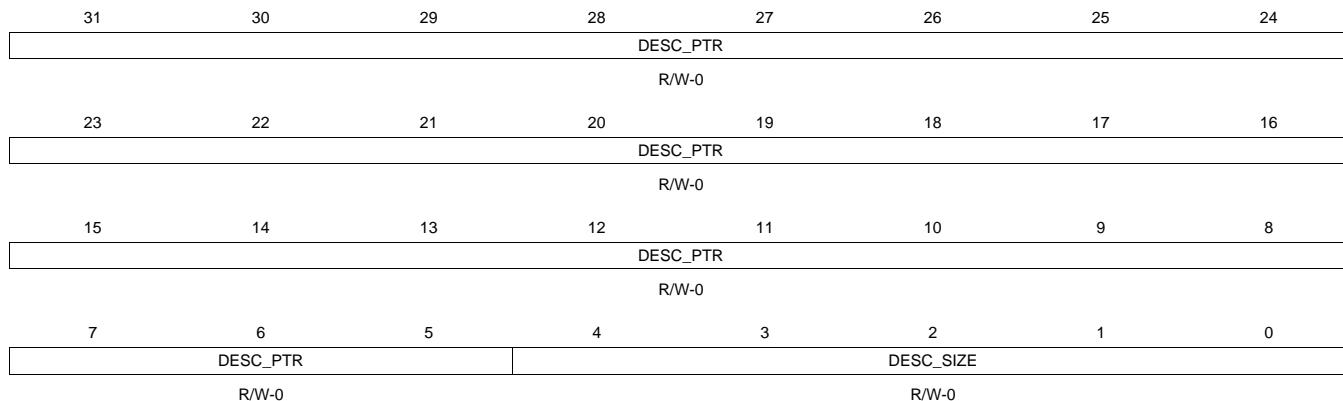
**Table 16-903. QUEUE\_144\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.614 QUEUE\_144\_D Register (offset = 290Ch) [reset = 0h]

QUEUE\_144\_D is shown in [Figure 16-890](#) and described in [Table 16-904](#).

**Figure 16-890. QUEUE\_144\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-904. QUEUE\_144\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.615 QUEUE\_145\_A Register (offset = 2910h) [reset = 0h]**

QUEUE\_145\_A is shown in [Figure 16-891](#) and described in [Table 16-905](#).

**Figure 16-891. QUEUE\_145\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-905. QUEUE\_145\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

### 16.5.7.616 QUEUE\_145\_B Register (offset = 2914h) [reset = 0h]

QUEUE\_145\_B is shown in [Figure 16-892](#) and described in [Table 16-906](#).

**Figure 16-892. QUEUE\_145\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

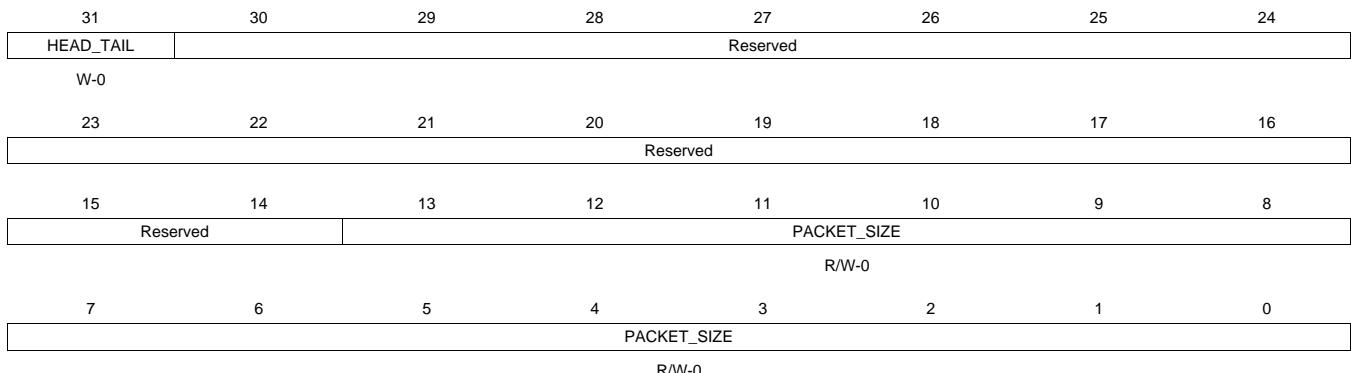
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-906. QUEUE\_145\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.617 QUEUE\_145\_C Register (offset = 2918h) [reset = 0h]**

QUEUE\_145\_C is shown in [Figure 16-893](#) and described in [Table 16-907](#).

**Figure 16-893. QUEUE\_145\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

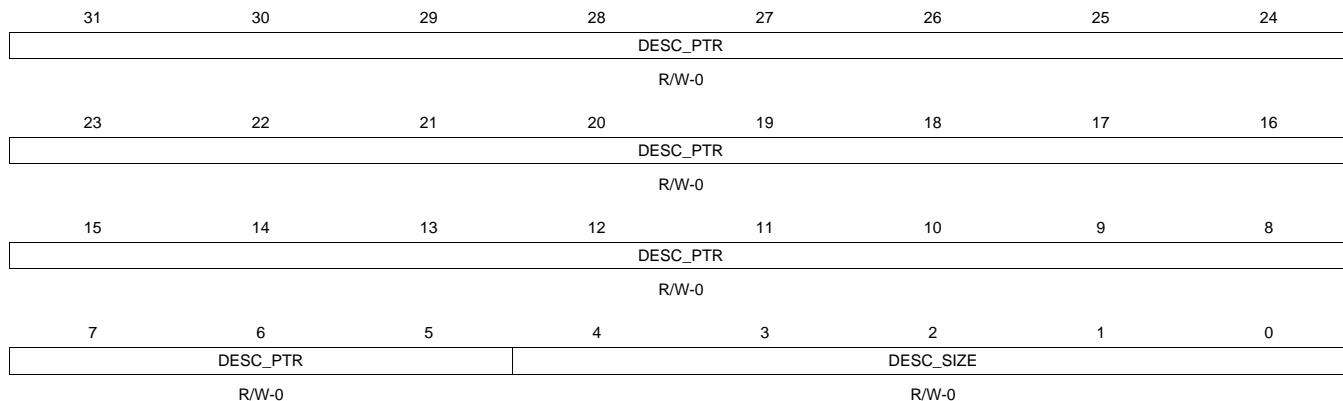
**Table 16-907. QUEUE\_145\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.618 QUEUE\_145\_D Register (offset = 291Ch) [reset = 0h]

QUEUE\_145\_D is shown in [Figure 16-894](#) and described in [Table 16-908](#).

**Figure 16-894. QUEUE\_145\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-908. QUEUE\_145\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.619 QUEUE\_146\_A Register (offset = 2920h) [reset = 0h]**

QUEUE\_146\_A is shown in [Figure 16-895](#) and described in [Table 16-909](#).

**Figure 16-895. QUEUE\_146\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-909. QUEUE\_146\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.620 QUEUE\_146\_B Register (offset = 2924h) [reset = 0h]**

QUEUE\_146\_B is shown in [Figure 16-896](#) and described in [Table 16-910](#).

**Figure 16-896. QUEUE\_146\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

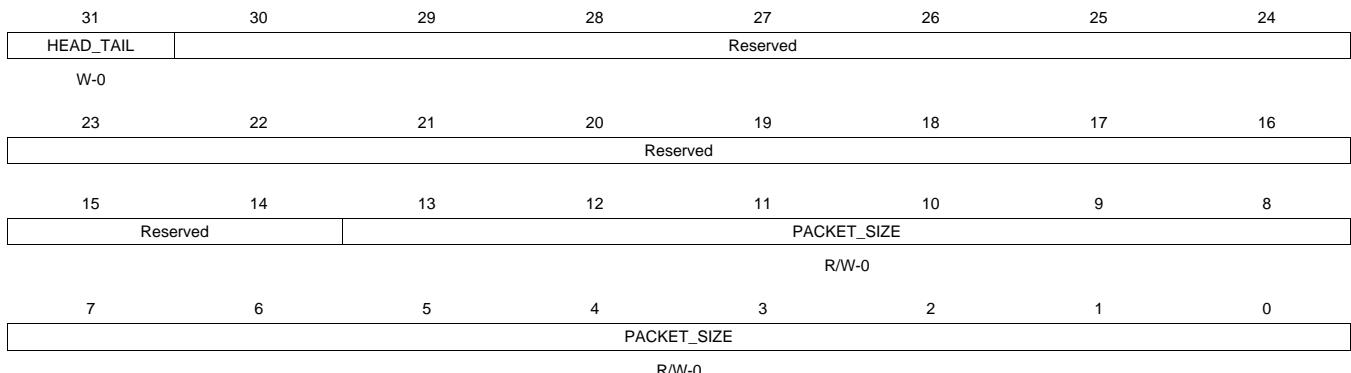
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-910. QUEUE\_146\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.621 QUEUE\_146\_C Register (offset = 2928h) [reset = 0h]**

QUEUE\_146\_C is shown in [Figure 16-897](#) and described in [Table 16-911](#).

**Figure 16-897. QUEUE\_146\_C Register**


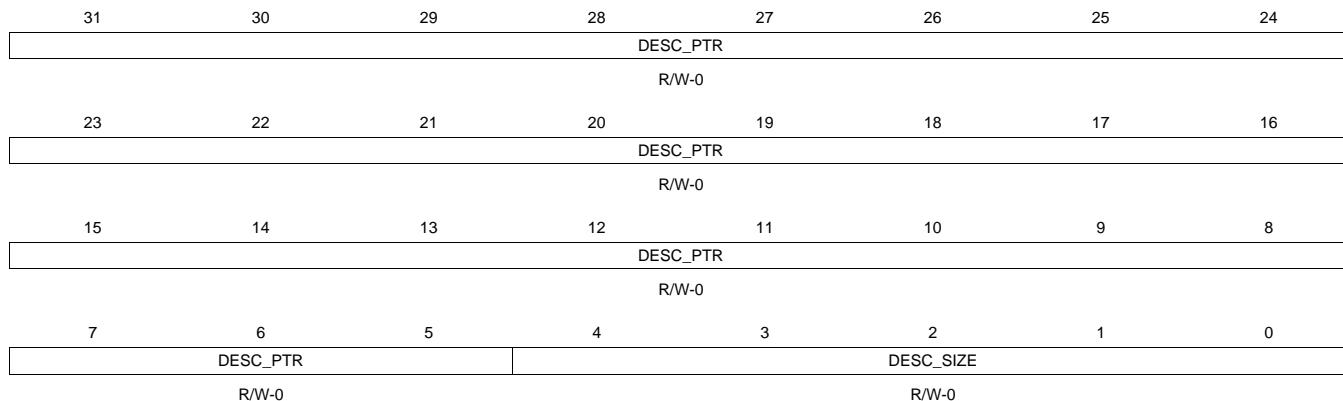
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-911. QUEUE\_146\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.622 QUEUE\_146\_D Register (offset = 292Ch) [reset = 0h]**

 QUEUE\_146\_D is shown in [Figure 16-898](#) and described in [Table 16-912](#).

**Figure 16-898. QUEUE\_146\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-912. QUEUE\_146\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.623 QUEUE\_147\_A Register (offset = 2930h) [reset = 0h]**

QUEUE\_147\_A is shown in [Figure 16-899](#) and described in [Table 16-913](#).

**Figure 16-899. QUEUE\_147\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-913. QUEUE\_147\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.624 QUEUE\_147\_B Register (offset = 2934h) [reset = 0h]**

QUEUE\_147\_B is shown in [Figure 16-900](#) and described in [Table 16-914](#).

**Figure 16-900. QUEUE\_147\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

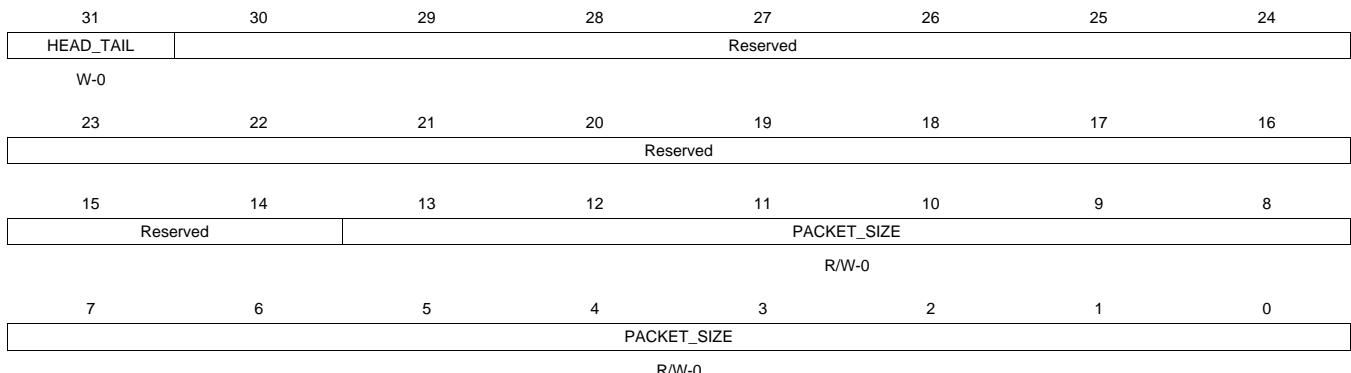
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-914. QUEUE\_147\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.625 QUEUE\_147\_C Register (offset = 2938h) [reset = 0h]**

QUEUE\_147\_C is shown in [Figure 16-901](#) and described in [Table 16-915](#).

**Figure 16-901. QUEUE\_147\_C Register**


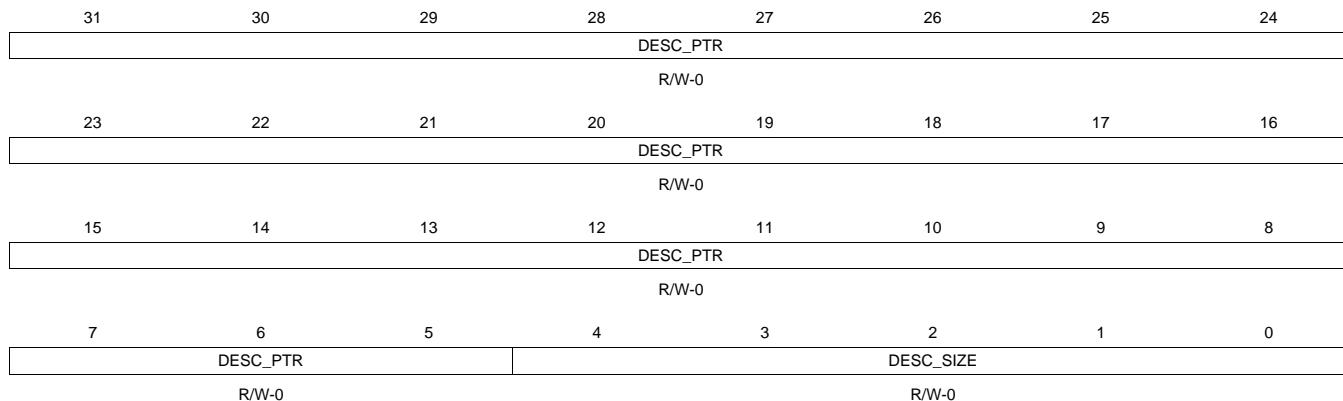
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-915. QUEUE\_147\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

**16.5.7.626 QUEUE\_147\_D Register (offset = 293Ch) [reset = 0h]**

 QUEUE\_147\_D is shown in [Figure 16-902](#) and described in [Table 16-916](#).

**Figure 16-902. QUEUE\_147\_D Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-916. QUEUE\_147\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>                     It will be read as zero if the queue is empty.<br/>                     It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>                     It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>                     This field will return a 0x0 when an empty queue is read.<br/>                     Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>                     However, register D has five LSBs that specify descriptor size.<br/>                     As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>                     The value that is read back is equal to always rounded to an odd number.<br/>                     So, for even values, the value read back is one more than what was written.<br/>                     For odd values, the value read back is same as the value that was written.<br/>                     Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>                     It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.627 QUEUE\_148\_A Register (offset = 2940h) [reset = 0h]**

QUEUE\_148\_A is shown in [Figure 16-903](#) and described in [Table 16-917](#).

**Figure 16-903. QUEUE\_148\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-917. QUEUE\_148\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.628 QUEUE\_148\_B Register (offset = 2944h) [reset = 0h]**

QUEUE\_148\_B is shown in [Figure 16-904](#) and described in [Table 16-918](#).

**Figure 16-904. QUEUE\_148\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

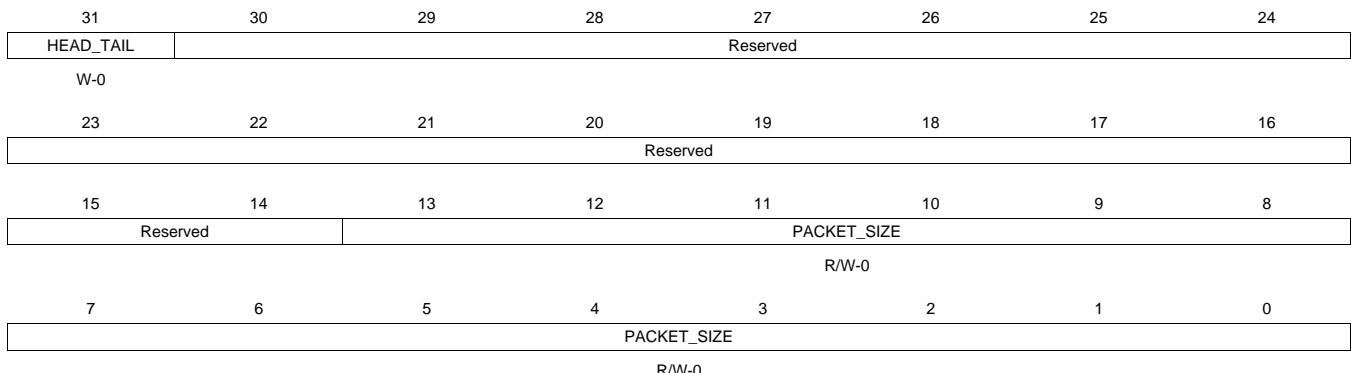
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-918. QUEUE\_148\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.629 QUEUE\_148\_C Register (offset = 2948h) [reset = 0h]**

QUEUE\_148\_C is shown in [Figure 16-905](#) and described in [Table 16-919](#).

**Figure 16-905. QUEUE\_148\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

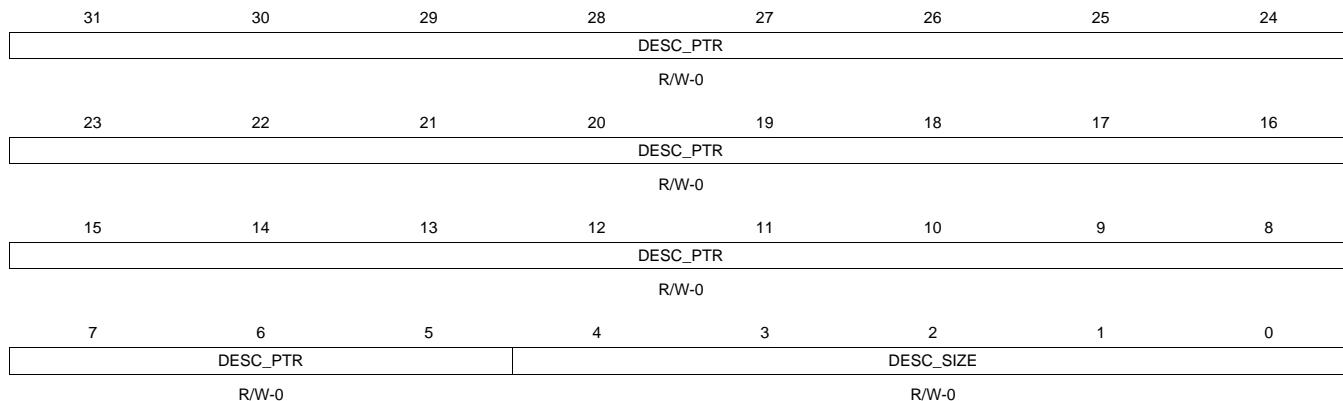
**Table 16-919. QUEUE\_148\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.630 QUEUE\_148\_D Register (offset = 294Ch) [reset = 0h]

QUEUE\_148\_D is shown in [Figure 16-906](#) and described in [Table 16-920](#).

**Figure 16-906. QUEUE\_148\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-920. QUEUE\_148\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.631 QUEUE\_149\_A Register (offset = 2950h) [reset = 0h]**

QUEUE\_149\_A is shown in [Figure 16-907](#) and described in [Table 16-921](#).

**Figure 16-907. QUEUE\_149\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-921. QUEUE\_149\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.632 QUEUE\_149\_B Register (offset = 2954h) [reset = 0h]**

QUEUE\_149\_B is shown in [Figure 16-908](#) and described in [Table 16-922](#).

**Figure 16-908. QUEUE\_149\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

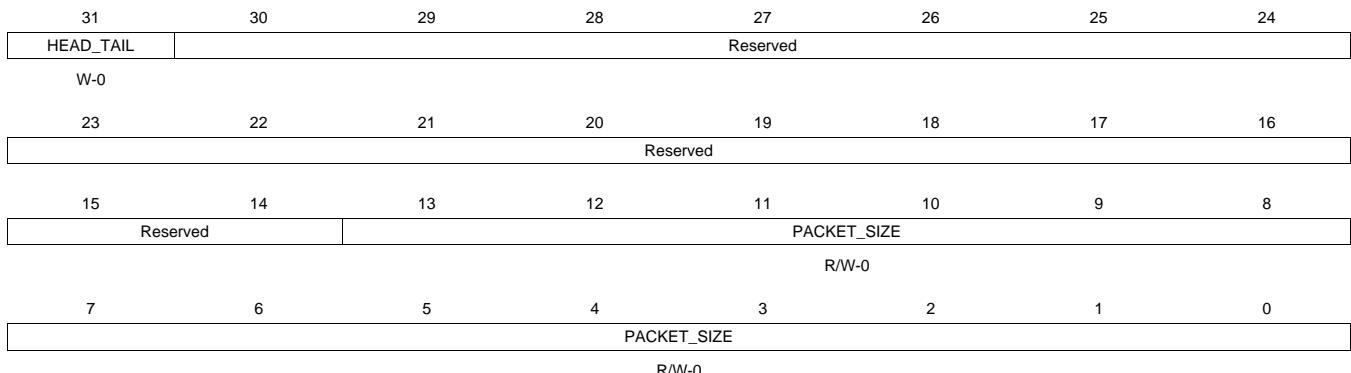
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-922. QUEUE\_149\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.633 QUEUE\_149\_C Register (offset = 2958h) [reset = 0h]**

QUEUE\_149\_C is shown in [Figure 16-909](#) and described in [Table 16-923](#).

**Figure 16-909. QUEUE\_149\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

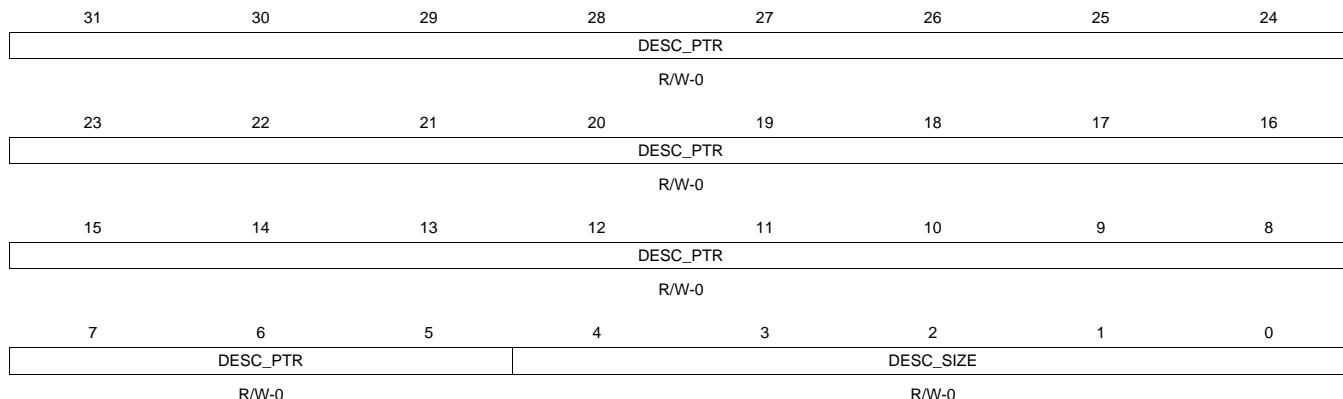
**Table 16-923. QUEUE\_149\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.634 QUEUE\_149\_D Register (offset = 295Ch) [reset = 0h]

QUEUE\_149\_D is shown in [Figure 16-910](#) and described in [Table 16-924](#).

**Figure 16-910. QUEUE\_149\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-924. QUEUE\_149\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.635 QUEUE\_150\_A Register (offset = 2960h) [reset = 0h]**

QUEUE\_150\_A is shown in [Figure 16-911](#) and described in [Table 16-925](#).

**Figure 16-911. QUEUE\_150\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-925. QUEUE\_150\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.636 QUEUE\_150\_B Register (offset = 2964h) [reset = 0h]**

QUEUE\_150\_B is shown in [Figure 16-912](#) and described in [Table 16-926](#).

**Figure 16-912. QUEUE\_150\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

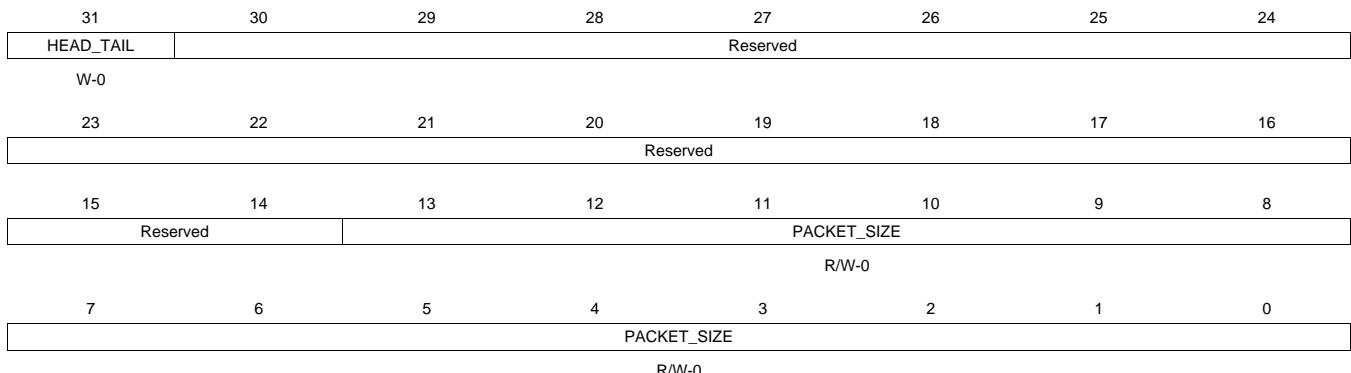
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-926. QUEUE\_150\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.637 QUEUE\_150\_C Register (offset = 2968h) [reset = 0h]**

 QUEUE\_150\_C is shown in [Figure 16-913](#) and described in [Table 16-927](#).

**Figure 16-913. QUEUE\_150\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

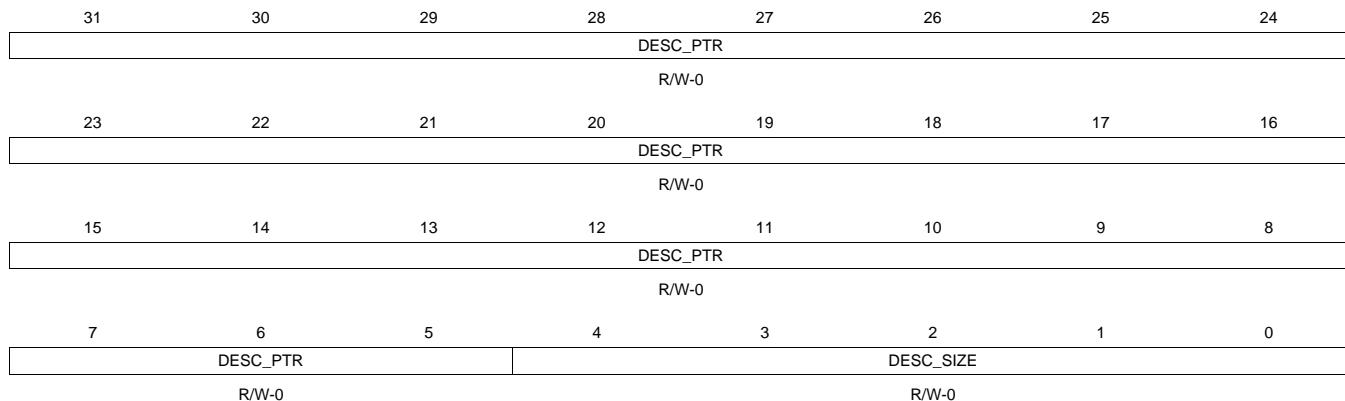
**Table 16-927. QUEUE\_150\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.638 QUEUE\_150\_D Register (offset = 296Ch) [reset = 0h]

QUEUE\_150\_D is shown in [Figure 16-914](#) and described in [Table 16-928](#).

**Figure 16-914. QUEUE\_150\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-928. QUEUE\_150\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.639 QUEUE\_151\_A Register (offset = 2970h) [reset = 0h]**

QUEUE\_151\_A is shown in [Figure 16-915](#) and described in [Table 16-929](#).

**Figure 16-915. QUEUE\_151\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-929. QUEUE\_151\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.640 QUEUE\_151\_B Register (offset = 2974h) [reset = 0h]**

QUEUE\_151\_B is shown in [Figure 16-916](#) and described in [Table 16-930](#).

**Figure 16-916. QUEUE\_151\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

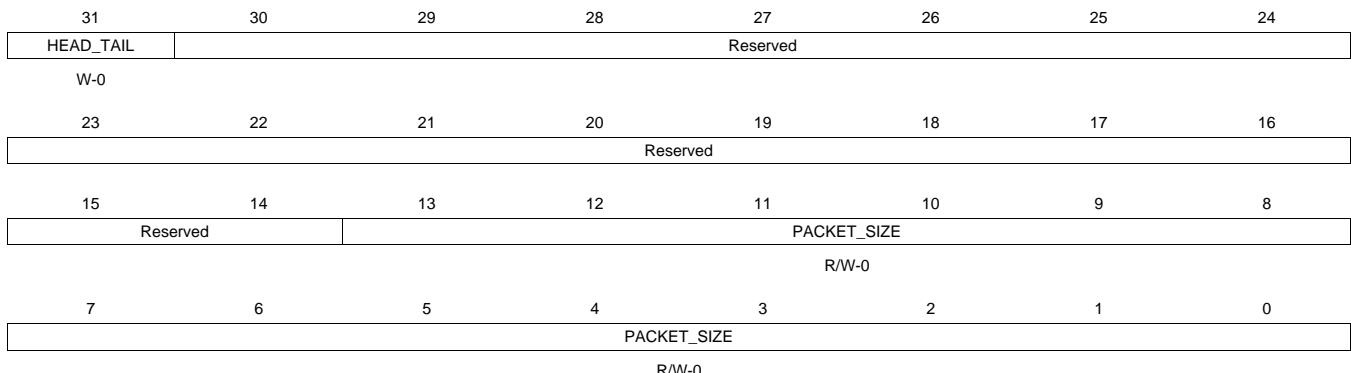
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-930. QUEUE\_151\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.641 QUEUE\_151\_C Register (offset = 2978h) [reset = 0h]**

 QUEUE\_151\_C is shown in [Figure 16-917](#) and described in [Table 16-931](#).

**Figure 16-917. QUEUE\_151\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

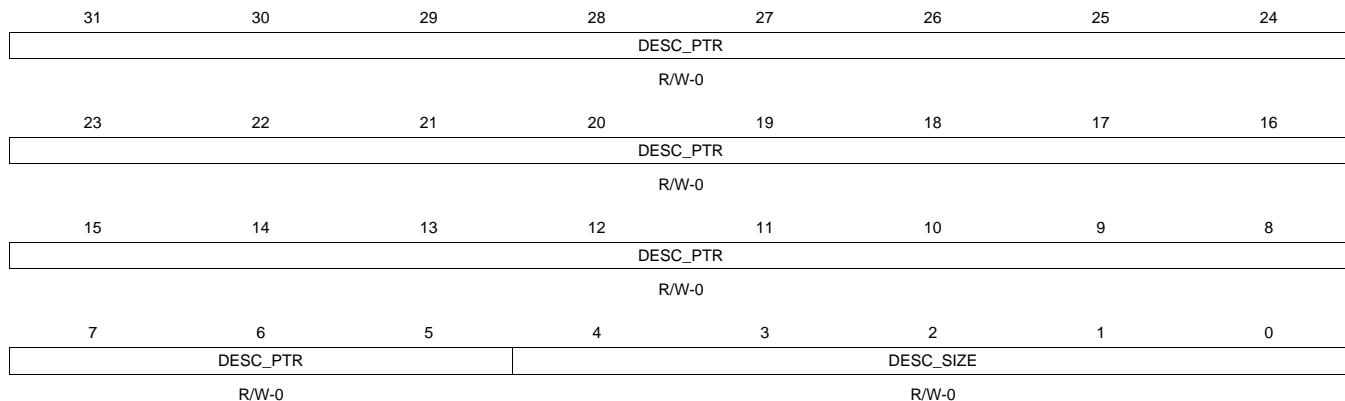
**Table 16-931. QUEUE\_151\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.642 QUEUE\_151\_D Register (offset = 297Ch) [reset = 0h]

QUEUE\_151\_D is shown in [Figure 16-918](#) and described in [Table 16-932](#).

**Figure 16-918. QUEUE\_151\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-932. QUEUE\_151\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.643 QUEUE\_152\_A Register (offset = 2980h) [reset = 0h]**

QUEUE\_152\_A is shown in [Figure 16-919](#) and described in [Table 16-933](#).

**Figure 16-919. QUEUE\_152\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-933. QUEUE\_152\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.644 QUEUE\_152\_B Register (offset = 2984h) [reset = 0h]**

QUEUE\_152\_B is shown in [Figure 16-920](#) and described in [Table 16-934](#).

**Figure 16-920. QUEUE\_152\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

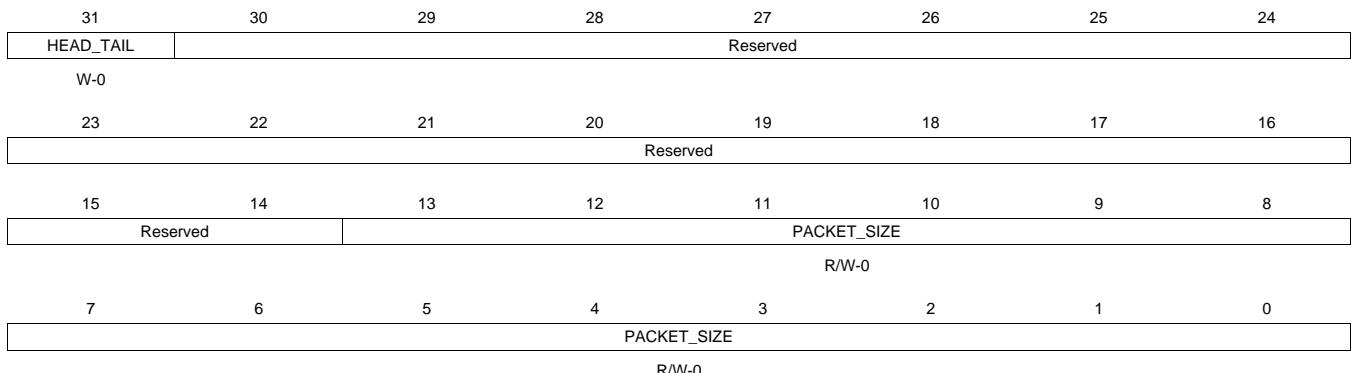
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-934. QUEUE\_152\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.645 QUEUE\_152\_C Register (offset = 2988h) [reset = 0h]**

 QUEUE\_152\_C is shown in [Figure 16-921](#) and described in [Table 16-935](#).

**Figure 16-921. QUEUE\_152\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

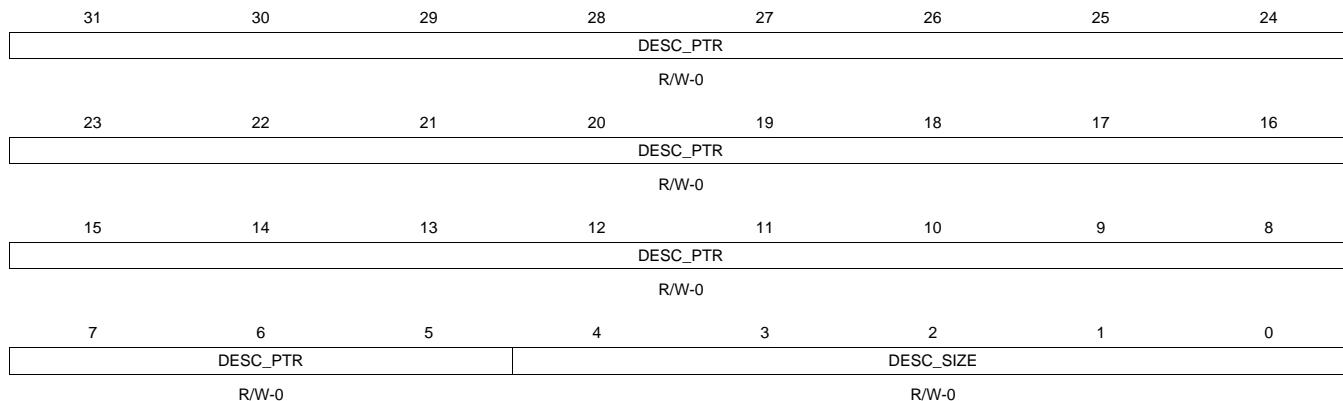
**Table 16-935. QUEUE\_152\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.646 QUEUE\_152\_D Register (offset = 298Ch) [reset = 0h]

QUEUE\_152\_D is shown in [Figure 16-922](#) and described in [Table 16-936](#).

**Figure 16-922. QUEUE\_152\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-936. QUEUE\_152\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.647 QUEUE\_153\_A Register (offset = 2990h) [reset = 0h]**

QUEUE\_153\_A is shown in [Figure 16-923](#) and described in [Table 16-937](#).

**Figure 16-923. QUEUE\_153\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-937. QUEUE\_153\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.648 QUEUE\_153\_B Register (offset = 2994h) [reset = 0h]**

QUEUE\_153\_B is shown in [Figure 16-924](#) and described in [Table 16-938](#).

**Figure 16-924. QUEUE\_153\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

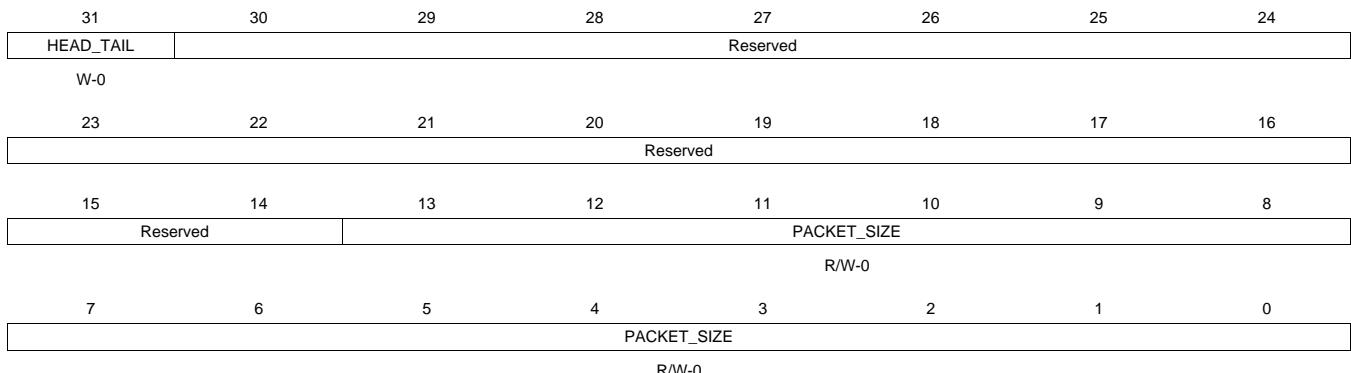
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-938. QUEUE\_153\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.649 QUEUE\_153\_C Register (offset = 2998h) [reset = 0h]**

QUEUE\_153\_C is shown in [Figure 16-925](#) and described in [Table 16-939](#).

**Figure 16-925. QUEUE\_153\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

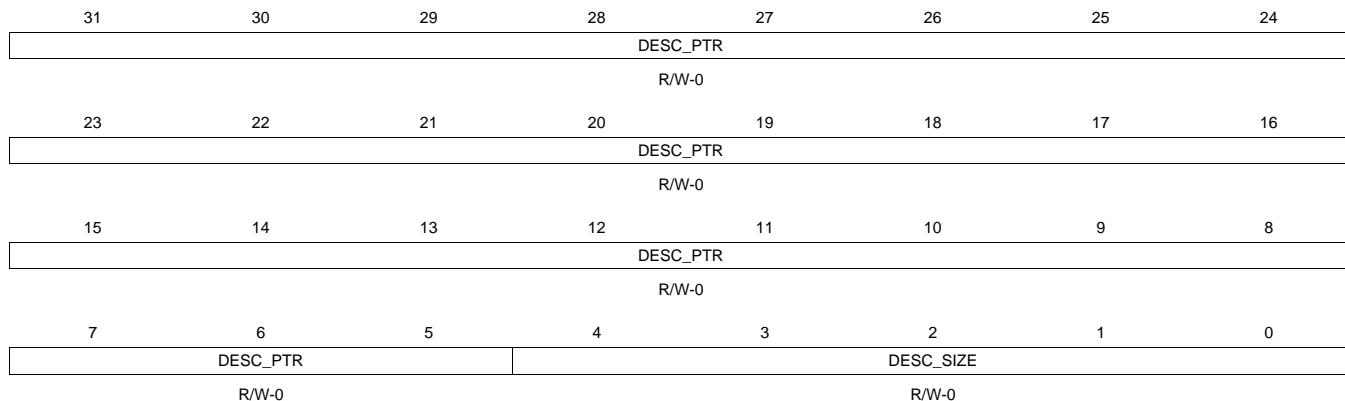
**Table 16-939. QUEUE\_153\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.650 QUEUE\_153\_D Register (offset = 299Ch) [reset = 0h]

QUEUE\_153\_D is shown in [Figure 16-926](#) and described in [Table 16-940](#).

**Figure 16-926. QUEUE\_153\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-940. QUEUE\_153\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.651 QUEUE\_154\_A Register (offset = 29A0h) [reset = 0h]**

QUEUE\_154\_A is shown in [Figure 16-927](#) and described in [Table 16-941](#).

**Figure 16-927. QUEUE\_154\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-941. QUEUE\_154\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.652 QUEUE\_154\_B Register (offset = 29A4h) [reset = 0h]**

QUEUE\_154\_B is shown in [Figure 16-928](#) and described in [Table 16-942](#).

**Figure 16-928. QUEUE\_154\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

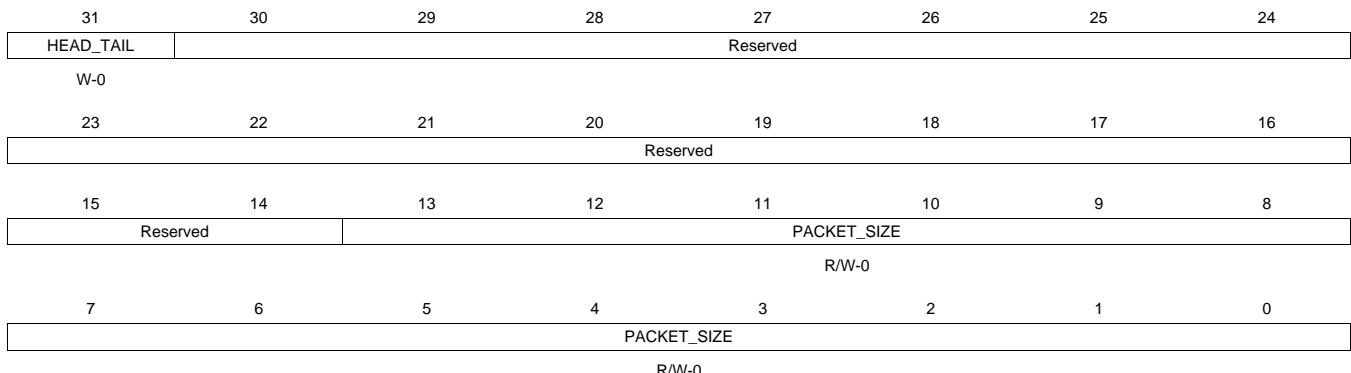
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-942. QUEUE\_154\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.653 QUEUE\_154\_C Register (offset = 29A8h) [reset = 0h]**

QUEUE\_154\_C is shown in [Figure 16-929](#) and described in [Table 16-943](#).

**Figure 16-929. QUEUE\_154\_C Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

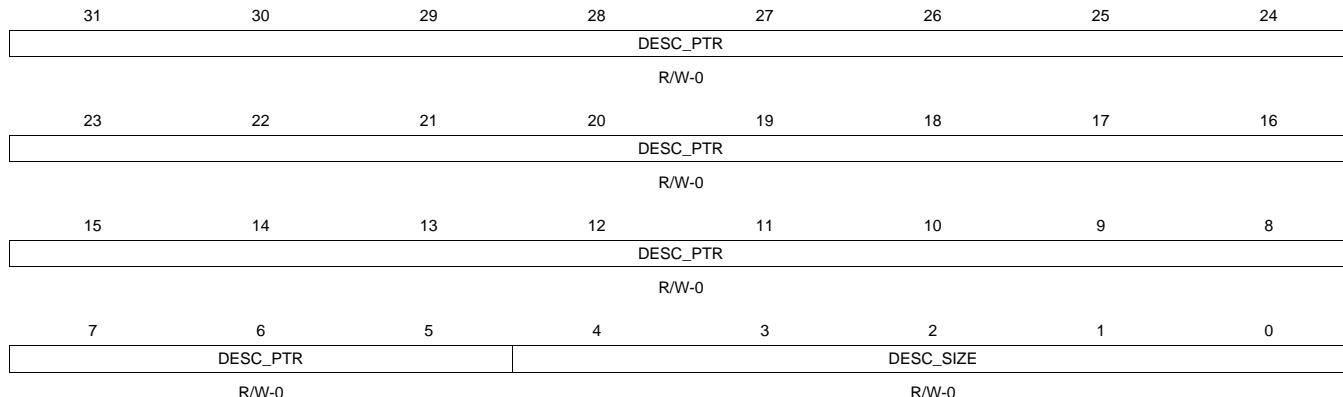
**Table 16-943. QUEUE\_154\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.654 QUEUE\_154\_D Register (offset = 29ACh) [reset = 0h]

QUEUE\_154\_D is shown in [Figure 16-930](#) and described in [Table 16-944](#).

**Figure 16-930. QUEUE\_154\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-944. QUEUE\_154\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.655 QUEUE\_155\_A Register (offset = 29B0h) [reset = 0h]**

QUEUE\_155\_A is shown in [Figure 16-931](#) and described in [Table 16-945](#).

**Figure 16-931. QUEUE\_155\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-945. QUEUE\_155\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | <p>This field indicates how many packets are currently queued on the queue.</p> <p>This count is incremented by 1 whenever a packet is added to the queue.</p> <p>This count is decremented by 1 whenever a packet is popped from the queue.</p> |

**16.5.7.656 QUEUE\_155\_B Register (offset = 29B4h) [reset = 0h]**

QUEUE\_155\_B is shown in [Figure 16-932](#) and described in [Table 16-946](#).

**Figure 16-932. QUEUE\_155\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

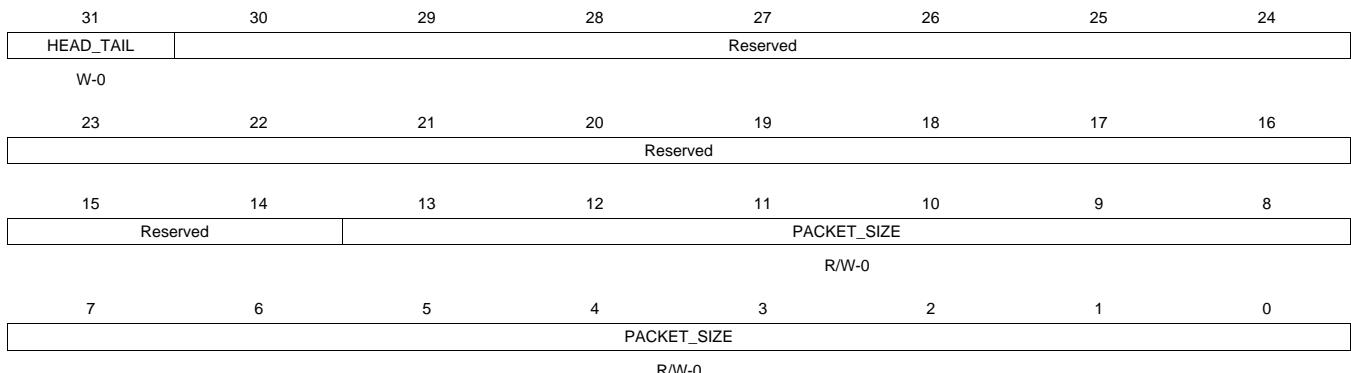
**Table 16-946. QUEUE\_155\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue. |

**16.5.7.657 QUEUE\_155\_C Register (offset = 29B8h) [reset = 0h]**

QUEUE\_155\_C is shown in [Figure 16-933](#) and described in [Table 16-947](#).

**Figure 16-933. QUEUE\_155\_C Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

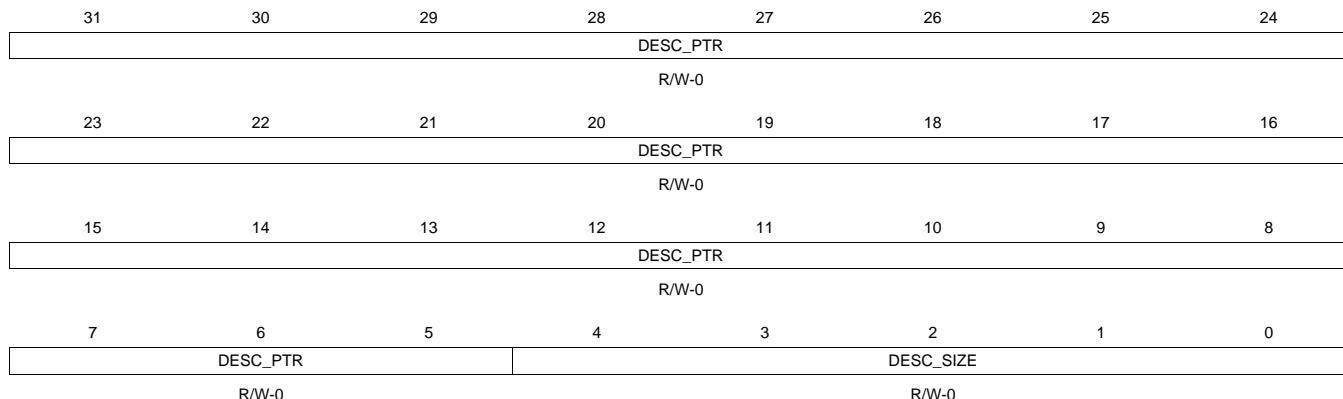
**Table 16-947. QUEUE\_155\_C Register Field Descriptions**

| Bit  | Field       | Type  | Reset | Description  |
|------|-------------|-------|-------|--|
| 31   | HEAD_TAIL   | W-0   | 0     | Head/Tail Push Control.<br>Set to zero in order to push packet onto tail of queue and set to one in order to push packet onto head of queue.   |
| 13-0 | PACKET_SIZE | R/W-0 | 0     | packet_size This field indicates packet size and is assumed to be zero on each packet add unless the value is explicitly overwritten. This field indicates packet size for packet pop operation. |

### 16.5.7.658 QUEUE\_155\_D Register (offset = 29BCh) [reset = 0h]

QUEUE\_155\_D is shown in [Figure 16-934](#) and described in [Table 16-948](#).

**Figure 16-934. QUEUE\_155\_D Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-948. QUEUE\_155\_D Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-5 | DESC_PTR  | R/W-0 | 0     | <p>Descriptor pointer.<br/>           It will be read as zero if the queue is empty.<br/>           It will indicate a 32-bit aligned address that points to a descriptor when the queue is not empty.</p>  |
| 4-0  | DESC_SIZE | R/W-0 | 0     | <p>Descriptor Size.<br/>           It is encoded in 4-byte increments with values 0 to 31 representing 24 and so on to 148 bytes.<br/>           This field will return a 0x0 when an empty queue is read.<br/>           Queue Manager Queue N Registers D To save hardware resources, the queue manager internally stores descriptor size (desc_size) information in four bits.<br/>           However, register D has five LSBs that specify descriptor size.<br/>           As a consequence, the value of desc_size that is pushed may not be same as that is read during a pop.<br/>           The value that is read back is equal to always rounded to an odd number.<br/>           So, for even values, the value read back is one more than what was written.<br/>           For odd values, the value read back is same as the value that was written.<br/>           Note that this 5-bit field (desc_size) is unrelated to the code for size of descriptors in a descriptor region.<br/>           It is just a place holder for a 5-bit value that is maintained across the push and pop operations for every descriptor managed by the queue manager.</p> |

**16.5.7.659 QUEUE\_0\_STATUS\_A Register (offset = 3000h) [reset = 0h]**

QUEUE\_0\_STATUS\_A is shown in [Figure 16-935](#) and described in [Table 16-949](#).

**Figure 16-935. QUEUE\_0\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-949. QUEUE\_0\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.660 QUEUE\_0\_STATUS\_B Register (offset = 3004h) [reset = 0h]**

QUEUE\_0\_STATUS\_B is shown in [Figure 16-936](#) and described in [Table 16-950](#).

**Figure 16-936. QUEUE\_0\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-950. QUEUE\_0\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.661 QUEUE\_0\_STATUS\_C Register (offset = 3008h) [reset = 0h]**

QUEUE\_0\_STATUS\_C is shown in [Figure 16-937](#) and described in [Table 16-951](#).

**Figure 16-937. QUEUE\_0\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-951. QUEUE\_0\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.662 QUEUE\_1\_STATUS\_A Register (offset = 3010h) [reset = 0h]**

QUEUE\_1\_STATUS\_A is shown in [Figure 16-938](#) and described in [Table 16-952](#).

**Figure 16-938. QUEUE\_1\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-952. QUEUE\_1\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.663 QUEUE\_1\_STATUS\_B Register (offset = 3014h) [reset = 0h]**

QUEUE\_1\_STATUS\_B is shown in [Figure 16-939](#) and described in [Table 16-953](#).

**Figure 16-939. QUEUE\_1\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-953. QUEUE\_1\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.664 QUEUE\_1\_STATUS\_C Register (offset = 3018h) [reset = 0h]**

QUEUE\_1\_STATUS\_C is shown in [Figure 16-940](#) and described in [Table 16-954](#).

**Figure 16-940. QUEUE\_1\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-954. QUEUE\_1\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.665 QUEUE\_2\_STATUS\_A Register (offset = 3020h) [reset = 0h]**

QUEUE\_2\_STATUS\_A is shown in [Figure 16-941](#) and described in [Table 16-955](#).

**Figure 16-941. QUEUE\_2\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-955. QUEUE\_2\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.666 QUEUE\_2\_STATUS\_B Register (offset = 3024h) [reset = 0h]**

QUEUE\_2\_STATUS\_B is shown in [Figure 16-942](#) and described in [Table 16-956](#).

**Figure 16-942. QUEUE\_2\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-956. QUEUE\_2\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.667 QUEUE\_2\_STATUS\_C Register (offset = 3028h) [reset = 0h]**

QUEUE\_2\_STATUS\_C is shown in [Figure 16-943](#) and described in [Table 16-957](#).

**Figure 16-943. QUEUE\_2\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-957. QUEUE\_2\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.668 QUEUE\_3\_STATUS\_A Register (offset = 3030h) [reset = 0h]**

QUEUE\_3\_STATUS\_A is shown in [Figure 16-944](#) and described in [Table 16-958](#).

**Figure 16-944. QUEUE\_3\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-958. QUEUE\_3\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.669 QUEUE\_3\_STATUS\_B Register (offset = 3034h) [reset = 0h]**

QUEUE\_3\_STATUS\_B is shown in [Figure 16-945](#) and described in [Table 16-959](#).

**Figure 16-945. QUEUE\_3\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-959. QUEUE\_3\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.670 QUEUE\_3\_STATUS\_C Register (offset = 3038h) [reset = 0h]**

QUEUE\_3\_STATUS\_C is shown in [Figure 16-946](#) and described in [Table 16-960](#).

**Figure 16-946. QUEUE\_3\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-960. QUEUE\_3\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.671 QUEUE\_4\_STATUS\_A Register (offset = 3040h) [reset = 0h]**

QUEUE\_4\_STATUS\_A is shown in [Figure 16-947](#) and described in [Table 16-961](#).

**Figure 16-947. QUEUE\_4\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-961. QUEUE\_4\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.672 QUEUE\_4\_STATUS\_B Register (offset = 3044h) [reset = 0h]**

 QUEUE\_4\_STATUS\_B is shown in [Figure 16-948](#) and described in [Table 16-962](#).

**Figure 16-948. QUEUE\_4\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-962. QUEUE\_4\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.673 QUEUE\_4\_STATUS\_C Register (offset = 3048h) [reset = 0h]**

QUEUE\_4\_STATUS\_C is shown in [Figure 16-949](#) and described in [Table 16-963](#).

**Figure 16-949. QUEUE\_4\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-963. QUEUE\_4\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.674 QUEUE\_5\_STATUS\_A Register (offset = 3050h) [reset = 0h]**

QUEUE\_5\_STATUS\_A is shown in [Figure 16-950](#) and described in [Table 16-964](#).

**Figure 16-950. QUEUE\_5\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-964. QUEUE\_5\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.675 QUEUE\_5\_STATUS\_B Register (offset = 3054h) [reset = 0h]**

QUEUE\_5\_STATUS\_B is shown in [Figure 16-951](#) and described in [Table 16-965](#).

**Figure 16-951. QUEUE\_5\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-965. QUEUE\_5\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.676 QUEUE\_5\_STATUS\_C Register (offset = 3058h) [reset = 0h]**

QUEUE\_5\_STATUS\_C is shown in [Figure 16-952](#) and described in [Table 16-966](#).

**Figure 16-952. QUEUE\_5\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-966. QUEUE\_5\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.677 QUEUE\_6\_STATUS\_A Register (offset = 3060h) [reset = 0h]**

QUEUE\_6\_STATUS\_A is shown in [Figure 16-953](#) and described in [Table 16-967](#).

**Figure 16-953. QUEUE\_6\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-967. QUEUE\_6\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.678 QUEUE\_6\_STATUS\_B Register (offset = 3064h) [reset = 0h]**

QUEUE\_6\_STATUS\_B is shown in [Figure 16-954](#) and described in [Table 16-968](#).

**Figure 16-954. QUEUE\_6\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-968. QUEUE\_6\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.679 QUEUE\_6\_STATUS\_C Register (offset = 3068h) [reset = 0h]**

QUEUE\_6\_STATUS\_C is shown in [Figure 16-955](#) and described in [Table 16-969](#).

**Figure 16-955. QUEUE\_6\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-969. QUEUE\_6\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.680 QUEUE\_7\_STATUS\_A Register (offset = 3070h) [reset = 0h]**

QUEUE\_7\_STATUS\_A is shown in [Figure 16-956](#) and described in [Table 16-970](#).

**Figure 16-956. QUEUE\_7\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-970. QUEUE\_7\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.681 QUEUE\_7\_STATUS\_B Register (offset = 3074h) [reset = 0h]**

QUEUE\_7\_STATUS\_B is shown in [Figure 16-957](#) and described in [Table 16-971](#).

**Figure 16-957. QUEUE\_7\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-971. QUEUE\_7\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.682 QUEUE\_7\_STATUS\_C Register (offset = 3078h) [reset = 0h]**

QUEUE\_7\_STATUS\_C is shown in [Figure 16-958](#) and described in [Table 16-972](#).

**Figure 16-958. QUEUE\_7\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-972. QUEUE\_7\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.683 QUEUE\_8\_STATUS\_A Register (offset = 3080h) [reset = 0h]**

QUEUE\_8\_STATUS\_A is shown in [Figure 16-959](#) and described in [Table 16-973](#).

**Figure 16-959. QUEUE\_8\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-973. QUEUE\_8\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.684 QUEUE\_8\_STATUS\_B Register (offset = 3084h) [reset = 0h]**

QUEUE\_8\_STATUS\_B is shown in [Figure 16-960](#) and described in [Table 16-974](#).

**Figure 16-960. QUEUE\_8\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-974. QUEUE\_8\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.685 QUEUE\_8\_STATUS\_C Register (offset = 3088h) [reset = 0h]**

QUEUE\_8\_STATUS\_C is shown in [Figure 16-961](#) and described in [Table 16-975](#).

**Figure 16-961. QUEUE\_8\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-975. QUEUE\_8\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.686 QUEUE\_9\_STATUS\_A Register (offset = 3090h) [reset = 0h]**

QUEUE\_9\_STATUS\_A is shown in [Figure 16-962](#) and described in [Table 16-976](#).

**Figure 16-962. QUEUE\_9\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-976. QUEUE\_9\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.687 QUEUE\_9\_STATUS\_B Register (offset = 3094h) [reset = 0h]**

QUEUE\_9\_STATUS\_B is shown in [Figure 16-963](#) and described in [Table 16-977](#).

**Figure 16-963. QUEUE\_9\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-977. QUEUE\_9\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.688 QUEUE\_9\_STATUS\_C Register (offset = 3098h) [reset = 0h]**

QUEUE\_9\_STATUS\_C is shown in [Figure 16-964](#) and described in [Table 16-978](#).

**Figure 16-964. QUEUE\_9\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-978. QUEUE\_9\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.689 QUEUE\_10\_STATUS\_A Register (offset = 30A0h) [reset = 0h]**

QUEUE\_10\_STATUS\_A is shown in [Figure 16-965](#) and described in [Table 16-979](#).

**Figure 16-965. QUEUE\_10\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-979. QUEUE\_10\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.690 QUEUE\_10\_STATUS\_B Register (offset = 30A4h) [reset = 0h]**

QUEUE\_10\_STATUS\_B is shown in [Figure 16-966](#) and described in [Table 16-980](#).

**Figure 16-966. QUEUE\_10\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-980. QUEUE\_10\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.691 QUEUE\_10\_STATUS\_C Register (offset = 30A8h) [reset = 0h]**

QUEUE\_10\_STATUS\_C is shown in [Figure 16-967](#) and described in [Table 16-981](#).

**Figure 16-967. QUEUE\_10\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-981. QUEUE\_10\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.692 QUEUE\_11\_STATUS\_A Register (offset = 30B0h) [reset = 0h]**

QUEUE\_11\_STATUS\_A is shown in [Figure 16-968](#) and described in [Table 16-982](#).

**Figure 16-968. QUEUE\_11\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-982. QUEUE\_11\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.693 QUEUE\_11\_STATUS\_B Register (offset = 30B4h) [reset = 0h]**

QUEUE\_11\_STATUS\_B is shown in [Figure 16-969](#) and described in [Table 16-983](#).

**Figure 16-969. QUEUE\_11\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-983. QUEUE\_11\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.694 QUEUE\_11\_STATUS\_C Register (offset = 30B8h) [reset = 0h]**

QUEUE\_11\_STATUS\_C is shown in [Figure 16-970](#) and described in [Table 16-984](#).

**Figure 16-970. QUEUE\_11\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-984. QUEUE\_11\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.695 QUEUE\_12\_STATUS\_A Register (offset = 30C0h) [reset = 0h]**

QUEUE\_12\_STATUS\_A is shown in [Figure 16-971](#) and described in [Table 16-985](#).

**Figure 16-971. QUEUE\_12\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-985. QUEUE\_12\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.696 QUEUE\_12\_STATUS\_B Register (offset = 30C4h) [reset = 0h]**

QUEUE\_12\_STATUS\_B is shown in [Figure 16-972](#) and described in [Table 16-986](#).

**Figure 16-972. QUEUE\_12\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-986. QUEUE\_12\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.697 QUEUE\_12\_STATUS\_C Register (offset = 30C8h) [reset = 0h]**

QUEUE\_12\_STATUS\_C is shown in [Figure 16-973](#) and described in [Table 16-987](#).

**Figure 16-973. QUEUE\_12\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-987. QUEUE\_12\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.698 QUEUE\_13\_STATUS\_A Register (offset = 30D0h) [reset = 0h]**

QUEUE\_13\_STATUS\_A is shown in [Figure 16-974](#) and described in [Table 16-988](#).

**Figure 16-974. QUEUE\_13\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-988. QUEUE\_13\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.699 QUEUE\_13\_STATUS\_B Register (offset = 30D4h) [reset = 0h]**

QUEUE\_13\_STATUS\_B is shown in [Figure 16-975](#) and described in [Table 16-989](#).

**Figure 16-975. QUEUE\_13\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-989. QUEUE\_13\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.700 QUEUE\_13\_STATUS\_C Register (offset = 30D8h) [reset = 0h]**

QUEUE\_13\_STATUS\_C is shown in [Figure 16-976](#) and described in [Table 16-990](#).

**Figure 16-976. QUEUE\_13\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-990. QUEUE\_13\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.701 QUEUE\_14\_STATUS\_A Register (offset = 30E0h) [reset = 0h]**

QUEUE\_14\_STATUS\_A is shown in [Figure 16-977](#) and described in [Table 16-991](#).

**Figure 16-977. QUEUE\_14\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-991. QUEUE\_14\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.702 QUEUE\_14\_STATUS\_B Register (offset = 30E4h) [reset = 0h]**

QUEUE\_14\_STATUS\_B is shown in [Figure 16-978](#) and described in [Table 16-992](#).

**Figure 16-978. QUEUE\_14\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-992. QUEUE\_14\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.703 QUEUE\_14\_STATUS\_C Register (offset = 30E8h) [reset = 0h]**

QUEUE\_14\_STATUS\_C is shown in [Figure 16-979](#) and described in [Table 16-993](#).

**Figure 16-979. QUEUE\_14\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-993. QUEUE\_14\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.704 QUEUE\_15\_STATUS\_A Register (offset = 30F0h) [reset = 0h]**

QUEUE\_15\_STATUS\_A is shown in [Figure 16-980](#) and described in [Table 16-994](#).

**Figure 16-980. QUEUE\_15\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-994. QUEUE\_15\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.705 QUEUE\_15\_STATUS\_B Register (offset = 30F4h) [reset = 0h]**

QUEUE\_15\_STATUS\_B is shown in [Figure 16-981](#) and described in [Table 16-995](#).

**Figure 16-981. QUEUE\_15\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-995. QUEUE\_15\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.706 QUEUE\_15\_STATUS\_C Register (offset = 30F8h) [reset = 0h]**

QUEUE\_15\_STATUS\_C is shown in [Figure 16-982](#) and described in [Table 16-996](#).

**Figure 16-982. QUEUE\_15\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17          | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3   | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|----|---|---|---|---|---|---|-----|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |    |   |   |   |   |   |   | R-0 |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |    |   |   |   |   |   |   |     |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-996. QUEUE\_15\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.707 QUEUE\_16\_STATUS\_A Register (offset = 3100h) [reset = 0h]**

QUEUE\_16\_STATUS\_A is shown in [Figure 16-983](#) and described in [Table 16-997](#).

**Figure 16-983. QUEUE\_16\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-997. QUEUE\_16\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.708 QUEUE\_16\_STATUS\_B Register (offset = 3104h) [reset = 0h]**

QUEUE\_16\_STATUS\_B is shown in [Figure 16-984](#) and described in [Table 16-998](#).

**Figure 16-984. QUEUE\_16\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-998. QUEUE\_16\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.709 QUEUE\_16\_STATUS\_C Register (offset = 3108h) [reset = 0h]**

QUEUE\_16\_STATUS\_C is shown in [Figure 16-985](#) and described in [Table 16-999](#).

**Figure 16-985. QUEUE\_16\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-999. QUEUE\_16\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.710 QUEUE\_17\_STATUS\_A Register (offset = 3110h) [reset = 0h]**

QUEUE\_17\_STATUS\_A is shown in [Figure 16-986](#) and described in [Table 16-1000](#).

**Figure 16-986. QUEUE\_17\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1000. QUEUE\_17\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.711 QUEUE\_17\_STATUS\_B Register (offset = 3114h) [reset = 0h]**

QUEUE\_17\_STATUS\_B is shown in [Figure 16-987](#) and described in [Table 16-1001](#).

**Figure 16-987. QUEUE\_17\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1001. QUEUE\_17\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.712 QUEUE\_17\_STATUS\_C Register (offset = 3118h) [reset = 0h]**

QUEUE\_17\_STATUS\_C is shown in [Figure 16-988](#) and described in [Table 16-1002](#).

**Figure 16-988. QUEUE\_17\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1002. QUEUE\_17\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.713 QUEUE\_18\_STATUS\_A Register (offset = 3120h) [reset = 0h]**

QUEUE\_18\_STATUS\_A is shown in [Figure 16-989](#) and described in [Table 16-1003](#).

**Figure 16-989. QUEUE\_18\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1003. QUEUE\_18\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.714 QUEUE\_18\_STATUS\_B Register (offset = 3124h) [reset = 0h]**

QUEUE\_18\_STATUS\_B is shown in [Figure 16-990](#) and described in [Table 16-1004](#).

**Figure 16-990. QUEUE\_18\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1004. QUEUE\_18\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.715 QUEUE\_18\_STATUS\_C Register (offset = 3128h) [reset = 0h]**

QUEUE\_18\_STATUS\_C is shown in [Figure 16-991](#) and described in [Table 16-1005](#).

**Figure 16-991. QUEUE\_18\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1005. QUEUE\_18\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.716 QUEUE\_19\_STATUS\_A Register (offset = 3130h) [reset = 0h]**

QUEUE\_19\_STATUS\_A is shown in [Figure 16-992](#) and described in [Table 16-1006](#).

**Figure 16-992. QUEUE\_19\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1006. QUEUE\_19\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.717 QUEUE\_19\_STATUS\_B Register (offset = 3134h) [reset = 0h]**

QUEUE\_19\_STATUS\_B is shown in [Figure 16-993](#) and described in [Table 16-1007](#).

**Figure 16-993. QUEUE\_19\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1007. QUEUE\_19\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.718 QUEUE\_19\_STATUS\_C Register (offset = 3138h) [reset = 0h]**

QUEUE\_19\_STATUS\_C is shown in [Figure 16-994](#) and described in [Table 16-1008](#).

**Figure 16-994. QUEUE\_19\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1008. QUEUE\_19\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.719 QUEUE\_20\_STATUS\_A Register (offset = 3140h) [reset = 0h]**

QUEUE\_20\_STATUS\_A is shown in [Figure 16-995](#) and described in [Table 16-1009](#).

**Figure 16-995. QUEUE\_20\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1009. QUEUE\_20\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.720 QUEUE\_20\_STATUS\_B Register (offset = 3144h) [reset = 0h]**

QUEUE\_20\_STATUS\_B is shown in [Figure 16-996](#) and described in [Table 16-1010](#).

**Figure 16-996. QUEUE\_20\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1010. QUEUE\_20\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.721 QUEUE\_20\_STATUS\_C Register (offset = 3148h) [reset = 0h]**

QUEUE\_20\_STATUS\_C is shown in [Figure 16-997](#) and described in [Table 16-1011](#).

**Figure 16-997. QUEUE\_20\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1011. QUEUE\_20\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.722 QUEUE\_21\_STATUS\_A Register (offset = 3150h) [reset = 0h]**

QUEUE\_21\_STATUS\_A is shown in [Figure 16-998](#) and described in [Table 16-1012](#).

**Figure 16-998. QUEUE\_21\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1012. QUEUE\_21\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.723 QUEUE\_21\_STATUS\_B Register (offset = 3154h) [reset = 0h]**

QUEUE\_21\_STATUS\_B is shown in [Figure 16-999](#) and described in [Table 16-1013](#).

**Figure 16-999. QUEUE\_21\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1013. QUEUE\_21\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.724 QUEUE\_21\_STATUS\_C Register (offset = 3158h) [reset = 0h]**

QUEUE\_21\_STATUS\_C is shown in [Figure 16-1000](#) and described in [Table 16-1014](#).

**Figure 16-1000. QUEUE\_21\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1014. QUEUE\_21\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.725 QUEUE\_22\_STATUS\_A Register (offset = 3160h) [reset = 0h]**

QUEUE\_22\_STATUS\_A is shown in [Figure 16-1001](#) and described in [Table 16-1015](#).

**Figure 16-1001. QUEUE\_22\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1015. QUEUE\_22\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.726 QUEUE\_22\_STATUS\_B Register (offset = 3164h) [reset = 0h]**

QUEUE\_22\_STATUS\_B is shown in [Figure 16-1002](#) and described in [Table 16-1016](#).

**Figure 16-1002. QUEUE\_22\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1016. QUEUE\_22\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.727 QUEUE\_22\_STATUS\_C Register (offset = 3168h) [reset = 0h]**

QUEUE\_22\_STATUS\_C is shown in [Figure 16-1003](#) and described in [Table 16-1017](#).

**Figure 16-1003. QUEUE\_22\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1017. QUEUE\_22\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.728 QUEUE\_23\_STATUS\_A Register (offset = 3170h) [reset = 0h]**

QUEUE\_23\_STATUS\_A is shown in [Figure 16-1004](#) and described in [Table 16-1018](#).

**Figure 16-1004. QUEUE\_23\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1018. QUEUE\_23\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.729 QUEUE\_23\_STATUS\_B Register (offset = 3174h) [reset = 0h]**

QUEUE\_23\_STATUS\_B is shown in [Figure 16-1005](#) and described in [Table 16-1019](#).

**Figure 16-1005. QUEUE\_23\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1019. QUEUE\_23\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.730 QUEUE\_23\_STATUS\_C Register (offset = 3178h) [reset = 0h]**

QUEUE\_23\_STATUS\_C is shown in [Figure 16-1006](#) and described in [Table 16-1020](#).

**Figure 16-1006. QUEUE\_23\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1020. QUEUE\_23\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.731 QUEUE\_24\_STATUS\_A Register (offset = 3180h) [reset = 0h]**

QUEUE\_24\_STATUS\_A is shown in [Figure 16-1007](#) and described in [Table 16-1021](#).

**Figure 16-1007. QUEUE\_24\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1021. QUEUE\_24\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.732 QUEUE\_24\_STATUS\_B Register (offset = 3184h) [reset = 0h]**

QUEUE\_24\_STATUS\_B is shown in [Figure 16-1008](#) and described in [Table 16-1022](#).

**Figure 16-1008. QUEUE\_24\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1022. QUEUE\_24\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.733 QUEUE\_24\_STATUS\_C Register (offset = 3188h) [reset = 0h]**

QUEUE\_24\_STATUS\_C is shown in [Figure 16-1009](#) and described in [Table 16-1023](#).

**Figure 16-1009. QUEUE\_24\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1023. QUEUE\_24\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.734 QUEUE\_25\_STATUS\_A Register (offset = 3190h) [reset = 0h]**

QUEUE\_25\_STATUS\_A is shown in [Figure 16-1010](#) and described in [Table 16-1024](#).

**Figure 16-1010. QUEUE\_25\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1024. QUEUE\_25\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.735 QUEUE\_25\_STATUS\_B Register (offset = 3194h) [reset = 0h]**

QUEUE\_25\_STATUS\_B is shown in [Figure 16-1011](#) and described in [Table 16-1025](#).

**Figure 16-1011. QUEUE\_25\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1025. QUEUE\_25\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.736 QUEUE\_25\_STATUS\_C Register (offset = 3198h) [reset = 0h]**

QUEUE\_25\_STATUS\_C is shown in [Figure 16-1012](#) and described in [Table 16-1026](#).

**Figure 16-1012. QUEUE\_25\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1026. QUEUE\_25\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.737 QUEUE\_26\_STATUS\_A Register (offset = 31A0h) [reset = 0h]**

QUEUE\_26\_STATUS\_A is shown in [Figure 16-1013](#) and described in [Table 16-1027](#).

**Figure 16-1013. QUEUE\_26\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1027. QUEUE\_26\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.738 QUEUE\_26\_STATUS\_B Register (offset = 31A4h) [reset = 0h]**

QUEUE\_26\_STATUS\_B is shown in [Figure 16-1014](#) and described in [Table 16-1028](#).

**Figure 16-1014. QUEUE\_26\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1028. QUEUE\_26\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.739 QUEUE\_26\_STATUS\_C Register (offset = 31A8h) [reset = 0h]**

QUEUE\_26\_STATUS\_C is shown in [Figure 16-1015](#) and described in [Table 16-1029](#).

**Figure 16-1015. QUEUE\_26\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1029. QUEUE\_26\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.740 QUEUE\_27\_STATUS\_A Register (offset = 31B0h) [reset = 0h]**

QUEUE\_27\_STATUS\_A is shown in [Figure 16-1016](#) and described in [Table 16-1030](#).

**Figure 16-1016. QUEUE\_27\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1030. QUEUE\_27\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.741 QUEUE\_27\_STATUS\_B Register (offset = 31B4h) [reset = 0h]**

QUEUE\_27\_STATUS\_B is shown in [Figure 16-1017](#) and described in [Table 16-1031](#).

**Figure 16-1017. QUEUE\_27\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1031. QUEUE\_27\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.742 QUEUE\_27\_STATUS\_C Register (offset = 31B8h) [reset = 0h]**

QUEUE\_27\_STATUS\_C is shown in [Figure 16-1018](#) and described in [Table 16-1032](#).

**Figure 16-1018. QUEUE\_27\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1032. QUEUE\_27\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.743 QUEUE\_28\_STATUS\_A Register (offset = 31C0h) [reset = 0h]**

QUEUE\_28\_STATUS\_A is shown in [Figure 16-1019](#) and described in [Table 16-1033](#).

**Figure 16-1019. QUEUE\_28\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1033. QUEUE\_28\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.744 QUEUE\_28\_STATUS\_B Register (offset = 31C4h) [reset = 0h]**

QUEUE\_28\_STATUS\_B is shown in [Figure 16-1020](#) and described in [Table 16-1034](#).

**Figure 16-1020. QUEUE\_28\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1034. QUEUE\_28\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.745 QUEUE\_28\_STATUS\_C Register (offset = 31C8h) [reset = 0h]**

QUEUE\_28\_STATUS\_C is shown in [Figure 16-1021](#) and described in [Table 16-1035](#).

**Figure 16-1021. QUEUE\_28\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1035. QUEUE\_28\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.746 QUEUE\_29\_STATUS\_A Register (offset = 31D0h) [reset = 0h]**

QUEUE\_29\_STATUS\_A is shown in [Figure 16-1022](#) and described in [Table 16-1036](#).

**Figure 16-1022. QUEUE\_29\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1036. QUEUE\_29\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.747 QUEUE\_29\_STATUS\_B Register (offset = 31D4h) [reset = 0h]**

QUEUE\_29\_STATUS\_B is shown in [Figure 16-1023](#) and described in [Table 16-1037](#).

**Figure 16-1023. QUEUE\_29\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1037. QUEUE\_29\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.748 QUEUE\_29\_STATUS\_C Register (offset = 31D8h) [reset = 0h]**

QUEUE\_29\_STATUS\_C is shown in [Figure 16-1024](#) and described in [Table 16-1038](#).

**Figure 16-1024. QUEUE\_29\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1038. QUEUE\_29\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.749 QUEUE\_30\_STATUS\_A Register (offset = 31E0h) [reset = 0h]**

QUEUE\_30\_STATUS\_A is shown in [Figure 16-1025](#) and described in [Table 16-1039](#).

**Figure 16-1025. QUEUE\_30\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1039. QUEUE\_30\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.750 QUEUE\_30\_STATUS\_B Register (offset = 31E4h) [reset = 0h]**

QUEUE\_30\_STATUS\_B is shown in [Figure 16-1026](#) and described in [Table 16-1040](#).

**Figure 16-1026. QUEUE\_30\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1040. QUEUE\_30\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.751 QUEUE\_30\_STATUS\_C Register (offset = 31E8h) [reset = 0h]**

QUEUE\_30\_STATUS\_C is shown in [Figure 16-1027](#) and described in [Table 16-1041](#).

**Figure 16-1027. QUEUE\_30\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1041. QUEUE\_30\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.752 QUEUE\_31\_STATUS\_A Register (offset = 31F0h) [reset = 0h]**

QUEUE\_31\_STATUS\_A is shown in [Figure 16-1028](#) and described in [Table 16-1042](#).

**Figure 16-1028. QUEUE\_31\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1042. QUEUE\_31\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.753 QUEUE\_31\_STATUS\_B Register (offset = 31F4h) [reset = 0h]**

QUEUE\_31\_STATUS\_B is shown in [Figure 16-1029](#) and described in [Table 16-1043](#).

**Figure 16-1029. QUEUE\_31\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1043. QUEUE\_31\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.754 QUEUE\_31\_STATUS\_C Register (offset = 31F8h) [reset = 0h]**

QUEUE\_31\_STATUS\_C is shown in [Figure 16-1030](#) and described in [Table 16-1044](#).

**Figure 16-1030. QUEUE\_31\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1044. QUEUE\_31\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.755 QUEUE\_32\_STATUS\_A Register (offset = 3200h) [reset = 0h]**

QUEUE\_32\_STATUS\_A is shown in [Figure 16-1031](#) and described in [Table 16-1045](#).

**Figure 16-1031. QUEUE\_32\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1045. QUEUE\_32\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.756 QUEUE\_32\_STATUS\_B Register (offset = 3204h) [reset = 0h]**

QUEUE\_32\_STATUS\_B is shown in [Figure 16-1032](#) and described in [Table 16-1046](#).

**Figure 16-1032. QUEUE\_32\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1046. QUEUE\_32\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.757 QUEUE\_32\_STATUS\_C Register (offset = 3208h) [reset = 0h]**

QUEUE\_32\_STATUS\_C is shown in [Figure 16-1033](#) and described in [Table 16-1047](#).

**Figure 16-1033. QUEUE\_32\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1047. QUEUE\_32\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.758 QUEUE\_33\_STATUS\_A Register (offset = 3210h) [reset = 0h]**

QUEUE\_33\_STATUS\_A is shown in [Figure 16-1034](#) and described in [Table 16-1048](#).

**Figure 16-1034. QUEUE\_33\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1048. QUEUE\_33\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.759 QUEUE\_33\_STATUS\_B Register (offset = 3214h) [reset = 0h]**

QUEUE\_33\_STATUS\_B is shown in [Figure 16-1035](#) and described in [Table 16-1049](#).

**Figure 16-1035. QUEUE\_33\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1049. QUEUE\_33\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.760 QUEUE\_33\_STATUS\_C Register (offset = 3218h) [reset = 0h]**

QUEUE\_33\_STATUS\_C is shown in [Figure 16-1036](#) and described in [Table 16-1050](#).

**Figure 16-1036. QUEUE\_33\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1050. QUEUE\_33\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.761 QUEUE\_34\_STATUS\_A Register (offset = 3220h) [reset = 0h]**

QUEUE\_34\_STATUS\_A is shown in [Figure 16-1037](#) and described in [Table 16-1051](#).

**Figure 16-1037. QUEUE\_34\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1051. QUEUE\_34\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.762 QUEUE\_34\_STATUS\_B Register (offset = 3224h) [reset = 0h]**

QUEUE\_34\_STATUS\_B is shown in [Figure 16-1038](#) and described in [Table 16-1052](#).

**Figure 16-1038. QUEUE\_34\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1052. QUEUE\_34\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.763 QUEUE\_34\_STATUS\_C Register (offset = 3228h) [reset = 0h]**

QUEUE\_34\_STATUS\_C is shown in [Figure 16-1039](#) and described in [Table 16-1053](#).

**Figure 16-1039. QUEUE\_34\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1053. QUEUE\_34\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.764 QUEUE\_35\_STATUS\_A Register (offset = 3230h) [reset = 0h]**

QUEUE\_35\_STATUS\_A is shown in [Figure 16-1040](#) and described in [Table 16-1054](#).

**Figure 16-1040. QUEUE\_35\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1054. QUEUE\_35\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.765 QUEUE\_35\_STATUS\_B Register (offset = 3234h) [reset = 0h]**

QUEUE\_35\_STATUS\_B is shown in [Figure 16-1041](#) and described in [Table 16-1055](#).

**Figure 16-1041. QUEUE\_35\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1055. QUEUE\_35\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.766 QUEUE\_35\_STATUS\_C Register (offset = 3238h) [reset = 0h]**

QUEUE\_35\_STATUS\_C is shown in [Figure 16-1042](#) and described in [Table 16-1056](#).

**Figure 16-1042. QUEUE\_35\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1056. QUEUE\_35\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.767 QUEUE\_36\_STATUS\_A Register (offset = 3240h) [reset = 0h]**

QUEUE\_36\_STATUS\_A is shown in [Figure 16-1043](#) and described in [Table 16-1057](#).

**Figure 16-1043. QUEUE\_36\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1057. QUEUE\_36\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.768 QUEUE\_36\_STATUS\_B Register (offset = 3244h) [reset = 0h]**

QUEUE\_36\_STATUS\_B is shown in [Figure 16-1044](#) and described in [Table 16-1058](#).

**Figure 16-1044. QUEUE\_36\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1058. QUEUE\_36\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.769 QUEUE\_36\_STATUS\_C Register (offset = 3248h) [reset = 0h]**

QUEUE\_36\_STATUS\_C is shown in [Figure 16-1045](#) and described in [Table 16-1059](#).

**Figure 16-1045. QUEUE\_36\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1059. QUEUE\_36\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.770 QUEUE\_37\_STATUS\_A Register (offset = 3250h) [reset = 0h]**

QUEUE\_37\_STATUS\_A is shown in [Figure 16-1046](#) and described in [Table 16-1060](#).

**Figure 16-1046. QUEUE\_37\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1060. QUEUE\_37\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.771 QUEUE\_37\_STATUS\_B Register (offset = 3254h) [reset = 0h]**

QUEUE\_37\_STATUS\_B is shown in [Figure 16-1047](#) and described in [Table 16-1061](#).

**Figure 16-1047. QUEUE\_37\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1061. QUEUE\_37\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.772 QUEUE\_37\_STATUS\_C Register (offset = 3258h) [reset = 0h]**

QUEUE\_37\_STATUS\_C is shown in [Figure 16-1048](#) and described in [Table 16-1062](#).

**Figure 16-1048. QUEUE\_37\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1062. QUEUE\_37\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.773 QUEUE\_38\_STATUS\_A Register (offset = 3260h) [reset = 0h]**

QUEUE\_38\_STATUS\_A is shown in [Figure 16-1049](#) and described in [Table 16-1063](#).

**Figure 16-1049. QUEUE\_38\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1063. QUEUE\_38\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.774 QUEUE\_38\_STATUS\_B Register (offset = 3264h) [reset = 0h]**

QUEUE\_38\_STATUS\_B is shown in [Figure 16-1050](#) and described in [Table 16-1064](#).

**Figure 16-1050. QUEUE\_38\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1064. QUEUE\_38\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.775 QUEUE\_38\_STATUS\_C Register (offset = 3268h) [reset = 0h]**

QUEUE\_38\_STATUS\_C is shown in [Figure 16-1051](#) and described in [Table 16-1065](#).

**Figure 16-1051. QUEUE\_38\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1065. QUEUE\_38\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.776 QUEUE\_39\_STATUS\_A Register (offset = 3270h) [reset = 0h]**

QUEUE\_39\_STATUS\_A is shown in [Figure 16-1052](#) and described in [Table 16-1066](#).

**Figure 16-1052. QUEUE\_39\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1066. QUEUE\_39\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.777 QUEUE\_39\_STATUS\_B Register (offset = 3274h) [reset = 0h]**

QUEUE\_39\_STATUS\_B is shown in [Figure 16-1053](#) and described in [Table 16-1067](#).

**Figure 16-1053. QUEUE\_39\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1067. QUEUE\_39\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.778 QUEUE\_39\_STATUS\_C Register (offset = 3278h) [reset = 0h]**

QUEUE\_39\_STATUS\_C is shown in [Figure 16-1054](#) and described in [Table 16-1068](#).

**Figure 16-1054. QUEUE\_39\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1068. QUEUE\_39\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.779 QUEUE\_40\_STATUS\_A Register (offset = 3280h) [reset = 0h]**

QUEUE\_40\_STATUS\_A is shown in [Figure 16-1055](#) and described in [Table 16-1069](#).

**Figure 16-1055. QUEUE\_40\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1069. QUEUE\_40\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.780 QUEUE\_40\_STATUS\_B Register (offset = 3284h) [reset = 0h]**

QUEUE\_40\_STATUS\_B is shown in [Figure 16-1056](#) and described in [Table 16-1070](#).

**Figure 16-1056. QUEUE\_40\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1070. QUEUE\_40\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.781 QUEUE\_40\_STATUS\_C Register (offset = 3288h) [reset = 0h]**

QUEUE\_40\_STATUS\_C is shown in [Figure 16-1057](#) and described in [Table 16-1071](#).

**Figure 16-1057. QUEUE\_40\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1071. QUEUE\_40\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.782 QUEUE\_41\_STATUS\_A Register (offset = 3290h) [reset = 0h]**

QUEUE\_41\_STATUS\_A is shown in [Figure 16-1058](#) and described in [Table 16-1072](#).

**Figure 16-1058. QUEUE\_41\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1072. QUEUE\_41\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.783 QUEUE\_41\_STATUS\_B Register (offset = 3294h) [reset = 0h]**

QUEUE\_41\_STATUS\_B is shown in [Figure 16-1059](#) and described in [Table 16-1073](#).

**Figure 16-1059. QUEUE\_41\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1073. QUEUE\_41\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.784 QUEUE\_41\_STATUS\_C Register (offset = 3298h) [reset = 0h]**

QUEUE\_41\_STATUS\_C is shown in [Figure 16-1060](#) and described in [Table 16-1074](#).

**Figure 16-1060. QUEUE\_41\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1074. QUEUE\_41\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.785 QUEUE\_42\_STATUS\_A Register (offset = 32A0h) [reset = 0h]**

QUEUE\_42\_STATUS\_A is shown in [Figure 16-1061](#) and described in [Table 16-1075](#).

**Figure 16-1061. QUEUE\_42\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1075. QUEUE\_42\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.786 QUEUE\_42\_STATUS\_B Register (offset = 32A4h) [reset = 0h]**

QUEUE\_42\_STATUS\_B is shown in [Figure 16-1062](#) and described in [Table 16-1076](#).

**Figure 16-1062. QUEUE\_42\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1076. QUEUE\_42\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.787 QUEUE\_42\_STATUS\_C Register (offset = 32A8h) [reset = 0h]**

QUEUE\_42\_STATUS\_C is shown in [Figure 16-1063](#) and described in [Table 16-1077](#).

**Figure 16-1063. QUEUE\_42\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1077. QUEUE\_42\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.788 QUEUE\_43\_STATUS\_A Register (offset = 32B0h) [reset = 0h]**

QUEUE\_43\_STATUS\_A is shown in [Figure 16-1064](#) and described in [Table 16-1078](#).

**Figure 16-1064. QUEUE\_43\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1078. QUEUE\_43\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.789 QUEUE\_43\_STATUS\_B Register (offset = 32B4h) [reset = 0h]**

QUEUE\_43\_STATUS\_B is shown in [Figure 16-1065](#) and described in [Table 16-1079](#).

**Figure 16-1065. QUEUE\_43\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1079. QUEUE\_43\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.790 QUEUE\_43\_STATUS\_C Register (offset = 32B8h) [reset = 0h]**

QUEUE\_43\_STATUS\_C is shown in [Figure 16-1066](#) and described in [Table 16-1080](#).

**Figure 16-1066. QUEUE\_43\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1080. QUEUE\_43\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.791 QUEUE\_44\_STATUS\_A Register (offset = 32C0h) [reset = 0h]**

QUEUE\_44\_STATUS\_A is shown in [Figure 16-1067](#) and described in [Table 16-1081](#).

**Figure 16-1067. QUEUE\_44\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1081. QUEUE\_44\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.792 QUEUE\_44\_STATUS\_B Register (offset = 32C4h) [reset = 0h]**

QUEUE\_44\_STATUS\_B is shown in [Figure 16-1068](#) and described in [Table 16-1082](#).

**Figure 16-1068. QUEUE\_44\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1082. QUEUE\_44\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.793 QUEUE\_44\_STATUS\_C Register (offset = 32C8h) [reset = 0h]**

QUEUE\_44\_STATUS\_C is shown in [Figure 16-1069](#) and described in [Table 16-1083](#).

**Figure 16-1069. QUEUE\_44\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1083. QUEUE\_44\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.794 QUEUE\_45\_STATUS\_A Register (offset = 32D0h) [reset = 0h]**

QUEUE\_45\_STATUS\_A is shown in [Figure 16-1070](#) and described in [Table 16-1084](#).

**Figure 16-1070. QUEUE\_45\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1084. QUEUE\_45\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.795 QUEUE\_45\_STATUS\_B Register (offset = 32D4h) [reset = 0h]**

QUEUE\_45\_STATUS\_B is shown in [Figure 16-1071](#) and described in [Table 16-1085](#).

**Figure 16-1071. QUEUE\_45\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1085. QUEUE\_45\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.796 QUEUE\_45\_STATUS\_C Register (offset = 32D8h) [reset = 0h]**

QUEUE\_45\_STATUS\_C is shown in [Figure 16-1072](#) and described in [Table 16-1086](#).

**Figure 16-1072. QUEUE\_45\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1086. QUEUE\_45\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.797 QUEUE\_46\_STATUS\_A Register (offset = 32E0h) [reset = 0h]**

QUEUE\_46\_STATUS\_A is shown in [Figure 16-1073](#) and described in [Table 16-1087](#).

**Figure 16-1073. QUEUE\_46\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1087. QUEUE\_46\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.798 QUEUE\_46\_STATUS\_B Register (offset = 32E4h) [reset = 0h]**

QUEUE\_46\_STATUS\_B is shown in [Figure 16-1074](#) and described in [Table 16-1088](#).

**Figure 16-1074. QUEUE\_46\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1088. QUEUE\_46\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.799 QUEUE\_46\_STATUS\_C Register (offset = 32E8h) [reset = 0h]**

QUEUE\_46\_STATUS\_C is shown in [Figure 16-1075](#) and described in [Table 16-1089](#).

**Figure 16-1075. QUEUE\_46\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1089. QUEUE\_46\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.800 QUEUE\_47\_STATUS\_A Register (offset = 32F0h) [reset = 0h]**

QUEUE\_47\_STATUS\_A is shown in [Figure 16-1076](#) and described in [Table 16-1090](#).

**Figure 16-1076. QUEUE\_47\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1090. QUEUE\_47\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.801 QUEUE\_47\_STATUS\_B Register (offset = 32F4h) [reset = 0h]**

QUEUE\_47\_STATUS\_B is shown in [Figure 16-1077](#) and described in [Table 16-1091](#).

**Figure 16-1077. QUEUE\_47\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1091. QUEUE\_47\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.802 QUEUE\_47\_STATUS\_C Register (offset = 32F8h) [reset = 0h]**

QUEUE\_47\_STATUS\_C is shown in [Figure 16-1078](#) and described in [Table 16-1092](#).

**Figure 16-1078. QUEUE\_47\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1092. QUEUE\_47\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.803 QUEUE\_48\_STATUS\_A Register (offset = 3300h) [reset = 0h]**

QUEUE\_48\_STATUS\_A is shown in [Figure 16-1079](#) and described in [Table 16-1093](#).

**Figure 16-1079. QUEUE\_48\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1093. QUEUE\_48\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.804 QUEUE\_48\_STATUS\_B Register (offset = 3304h) [reset = 0h]**

QUEUE\_48\_STATUS\_B is shown in [Figure 16-1080](#) and described in [Table 16-1094](#).

**Figure 16-1080. QUEUE\_48\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1094. QUEUE\_48\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.805 QUEUE\_48\_STATUS\_C Register (offset = 3308h) [reset = 0h]**

QUEUE\_48\_STATUS\_C is shown in [Figure 16-1081](#) and described in [Table 16-1095](#).

**Figure 16-1081. QUEUE\_48\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1095. QUEUE\_48\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.806 QUEUE\_49\_STATUS\_A Register (offset = 3310h) [reset = 0h]**

QUEUE\_49\_STATUS\_A is shown in [Figure 16-1082](#) and described in [Table 16-1096](#).

**Figure 16-1082. QUEUE\_49\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1096. QUEUE\_49\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.807 QUEUE\_49\_STATUS\_B Register (offset = 3314h) [reset = 0h]**

QUEUE\_49\_STATUS\_B is shown in [Figure 16-1083](#) and described in [Table 16-1097](#).

**Figure 16-1083. QUEUE\_49\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1097. QUEUE\_49\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.808 QUEUE\_49\_STATUS\_C Register (offset = 3318h) [reset = 0h]**

QUEUE\_49\_STATUS\_C is shown in [Figure 16-1084](#) and described in [Table 16-1098](#).

**Figure 16-1084. QUEUE\_49\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1098. QUEUE\_49\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.809 QUEUE\_50\_STATUS\_A Register (offset = 3320h) [reset = 0h]**

QUEUE\_50\_STATUS\_A is shown in [Figure 16-1085](#) and described in [Table 16-1099](#).

**Figure 16-1085. QUEUE\_50\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1099. QUEUE\_50\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.810 QUEUE\_50\_STATUS\_B Register (offset = 3324h) [reset = 0h]**

QUEUE\_50\_STATUS\_B is shown in [Figure 16-1086](#) and described in [Table 16-1100](#).

**Figure 16-1086. QUEUE\_50\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1100. QUEUE\_50\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.811 QUEUE\_50\_STATUS\_C Register (offset = 3328h) [reset = 0h]**

QUEUE\_50\_STATUS\_C is shown in [Figure 16-1087](#) and described in [Table 16-1101](#).

**Figure 16-1087. QUEUE\_50\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1101. QUEUE\_50\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.812 QUEUE\_51\_STATUS\_A Register (offset = 3330h) [reset = 0h]**

QUEUE\_51\_STATUS\_A is shown in [Figure 16-1088](#) and described in [Table 16-1102](#).

**Figure 16-1088. QUEUE\_51\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1102. QUEUE\_51\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.813 QUEUE\_51\_STATUS\_B Register (offset = 3334h) [reset = 0h]**

QUEUE\_51\_STATUS\_B is shown in [Figure 16-1089](#) and described in [Table 16-1103](#).

**Figure 16-1089. QUEUE\_51\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1103. QUEUE\_51\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.814 QUEUE\_51\_STATUS\_C Register (offset = 3338h) [reset = 0h]**

QUEUE\_51\_STATUS\_C is shown in [Figure 16-1090](#) and described in [Table 16-1104](#).

**Figure 16-1090. QUEUE\_51\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1104. QUEUE\_51\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.815 QUEUE\_52\_STATUS\_A Register (offset = 3340h) [reset = 0h]**

QUEUE\_52\_STATUS\_A is shown in [Figure 16-1091](#) and described in [Table 16-1105](#).

**Figure 16-1091. QUEUE\_52\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1105. QUEUE\_52\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.816 QUEUE\_52\_STATUS\_B Register (offset = 3344h) [reset = 0h]**

QUEUE\_52\_STATUS\_B is shown in [Figure 16-1092](#) and described in [Table 16-1106](#).

**Figure 16-1092. QUEUE\_52\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1106. QUEUE\_52\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.817 QUEUE\_52\_STATUS\_C Register (offset = 3348h) [reset = 0h]**

QUEUE\_52\_STATUS\_C is shown in [Figure 16-1093](#) and described in [Table 16-1107](#).

**Figure 16-1093. QUEUE\_52\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1107. QUEUE\_52\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.818 QUEUE\_53\_STATUS\_A Register (offset = 3350h) [reset = 0h]**

QUEUE\_53\_STATUS\_A is shown in [Figure 16-1094](#) and described in [Table 16-1108](#).

**Figure 16-1094. QUEUE\_53\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1108. QUEUE\_53\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.819 QUEUE\_53\_STATUS\_B Register (offset = 3354h) [reset = 0h]**

QUEUE\_53\_STATUS\_B is shown in [Figure 16-1095](#) and described in [Table 16-1109](#).

**Figure 16-1095. QUEUE\_53\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1109. QUEUE\_53\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.820 QUEUE\_53\_STATUS\_C Register (offset = 3358h) [reset = 0h]**

QUEUE\_53\_STATUS\_C is shown in [Figure 16-1096](#) and described in [Table 16-1110](#).

**Figure 16-1096. QUEUE\_53\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1110. QUEUE\_53\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.821 QUEUE\_54\_STATUS\_A Register (offset = 3360h) [reset = 0h]**

QUEUE\_54\_STATUS\_A is shown in [Figure 16-1097](#) and described in [Table 16-1111](#).

**Figure 16-1097. QUEUE\_54\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1111. QUEUE\_54\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.822 QUEUE\_54\_STATUS\_B Register (offset = 3364h) [reset = 0h]**

QUEUE\_54\_STATUS\_B is shown in [Figure 16-1098](#) and described in [Table 16-1112](#).

**Figure 16-1098. QUEUE\_54\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1112. QUEUE\_54\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.823 QUEUE\_54\_STATUS\_C Register (offset = 3368h) [reset = 0h]**

QUEUE\_54\_STATUS\_C is shown in [Figure 16-1099](#) and described in [Table 16-1113](#).

**Figure 16-1099. QUEUE\_54\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1113. QUEUE\_54\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.824 QUEUE\_55\_STATUS\_A Register (offset = 3370h) [reset = 0h]**

QUEUE\_55\_STATUS\_A is shown in [Figure 16-1100](#) and described in [Table 16-1114](#).

**Figure 16-1100. QUEUE\_55\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1114. QUEUE\_55\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.825 QUEUE\_55\_STATUS\_B Register (offset = 3374h) [reset = 0h]**

QUEUE\_55\_STATUS\_B is shown in [Figure 16-1101](#) and described in [Table 16-1115](#).

**Figure 16-1101. QUEUE\_55\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1115. QUEUE\_55\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.826 QUEUE\_55\_STATUS\_C Register (offset = 3378h) [reset = 0h]**

QUEUE\_55\_STATUS\_C is shown in [Figure 16-1102](#) and described in [Table 16-1116](#).

**Figure 16-1102. QUEUE\_55\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1116. QUEUE\_55\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.827 QUEUE\_56\_STATUS\_A Register (offset = 3380h) [reset = 0h]**

QUEUE\_56\_STATUS\_A is shown in [Figure 16-1103](#) and described in [Table 16-1117](#).

**Figure 16-1103. QUEUE\_56\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1117. QUEUE\_56\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.828 QUEUE\_56\_STATUS\_B Register (offset = 3384h) [reset = 0h]**

QUEUE\_56\_STATUS\_B is shown in [Figure 16-1104](#) and described in [Table 16-1118](#).

**Figure 16-1104. QUEUE\_56\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1118. QUEUE\_56\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.829 QUEUE\_56\_STATUS\_C Register (offset = 3388h) [reset = 0h]**

QUEUE\_56\_STATUS\_C is shown in [Figure 16-1105](#) and described in [Table 16-1119](#).

**Figure 16-1105. QUEUE\_56\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1119. QUEUE\_56\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.830 QUEUE\_57\_STATUS\_A Register (offset = 3390h) [reset = 0h]**

QUEUE\_57\_STATUS\_A is shown in [Figure 16-1106](#) and described in [Table 16-1120](#).

**Figure 16-1106. QUEUE\_57\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1120. QUEUE\_57\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.831 QUEUE\_57\_STATUS\_B Register (offset = 3394h) [reset = 0h]**

QUEUE\_57\_STATUS\_B is shown in [Figure 16-1107](#) and described in [Table 16-1121](#).

**Figure 16-1107. QUEUE\_57\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1121. QUEUE\_57\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.832 QUEUE\_57\_STATUS\_C Register (offset = 3398h) [reset = 0h]**

QUEUE\_57\_STATUS\_C is shown in [Figure 16-1108](#) and described in [Table 16-1122](#).

**Figure 16-1108. QUEUE\_57\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1122. QUEUE\_57\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.833 QUEUE\_58\_STATUS\_A Register (offset = 33A0h) [reset = 0h]**

QUEUE\_58\_STATUS\_A is shown in [Figure 16-1109](#) and described in [Table 16-1123](#).

**Figure 16-1109. QUEUE\_58\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1123. QUEUE\_58\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.834 QUEUE\_58\_STATUS\_B Register (offset = 33A4h) [reset = 0h]**

QUEUE\_58\_STATUS\_B is shown in [Figure 16-1110](#) and described in [Table 16-1124](#).

**Figure 16-1110. QUEUE\_58\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1124. QUEUE\_58\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.835 QUEUE\_58\_STATUS\_C Register (offset = 33A8h) [reset = 0h]**

QUEUE\_58\_STATUS\_C is shown in [Figure 16-1111](#) and described in [Table 16-1125](#).

**Figure 16-1111. QUEUE\_58\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1125. QUEUE\_58\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.836 QUEUE\_59\_STATUS\_A Register (offset = 33B0h) [reset = 0h]**

QUEUE\_59\_STATUS\_A is shown in [Figure 16-1112](#) and described in [Table 16-1126](#).

**Figure 16-1112. QUEUE\_59\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1126. QUEUE\_59\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.837 QUEUE\_59\_STATUS\_B Register (offset = 33B4h) [reset = 0h]**

QUEUE\_59\_STATUS\_B is shown in [Figure 16-1113](#) and described in [Table 16-1127](#).

**Figure 16-1113. QUEUE\_59\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1127. QUEUE\_59\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.838 QUEUE\_59\_STATUS\_C Register (offset = 33B8h) [reset = 0h]**

QUEUE\_59\_STATUS\_C is shown in [Figure 16-1114](#) and described in [Table 16-1128](#).

**Figure 16-1114. QUEUE\_59\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1128. QUEUE\_59\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.839 QUEUE\_60\_STATUS\_A Register (offset = 33C0h) [reset = 0h]**

QUEUE\_60\_STATUS\_A is shown in [Figure 16-1115](#) and described in [Table 16-1129](#).

**Figure 16-1115. QUEUE\_60\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1129. QUEUE\_60\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.840 QUEUE\_60\_STATUS\_B Register (offset = 33C4h) [reset = 0h]**

QUEUE\_60\_STATUS\_B is shown in [Figure 16-1116](#) and described in [Table 16-1130](#).

**Figure 16-1116. QUEUE\_60\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1130. QUEUE\_60\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.841 QUEUE\_60\_STATUS\_C Register (offset = 33C8h) [reset = 0h]**

QUEUE\_60\_STATUS\_C is shown in [Figure 16-1117](#) and described in [Table 16-1131](#).

**Figure 16-1117. QUEUE\_60\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1131. QUEUE\_60\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.842 QUEUE\_61\_STATUS\_A Register (offset = 33D0h) [reset = 0h]**

QUEUE\_61\_STATUS\_A is shown in [Figure 16-1118](#) and described in [Table 16-1132](#).

**Figure 16-1118. QUEUE\_61\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1132. QUEUE\_61\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.843 QUEUE\_61\_STATUS\_B Register (offset = 33D4h) [reset = 0h]**

QUEUE\_61\_STATUS\_B is shown in [Figure 16-1119](#) and described in [Table 16-1133](#).

**Figure 16-1119. QUEUE\_61\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1133. QUEUE\_61\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.844 QUEUE\_61\_STATUS\_C Register (offset = 33D8h) [reset = 0h]**

QUEUE\_61\_STATUS\_C is shown in [Figure 16-1120](#) and described in [Table 16-1134](#).

**Figure 16-1120. QUEUE\_61\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1134. QUEUE\_61\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.845 QUEUE\_62\_STATUS\_A Register (offset = 33E0h) [reset = 0h]**

QUEUE\_62\_STATUS\_A is shown in [Figure 16-1121](#) and described in [Table 16-1135](#).

**Figure 16-1121. QUEUE\_62\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1135. QUEUE\_62\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.846 QUEUE\_62\_STATUS\_B Register (offset = 33E4h) [reset = 0h]**

QUEUE\_62\_STATUS\_B is shown in [Figure 16-1122](#) and described in [Table 16-1136](#).

**Figure 16-1122. QUEUE\_62\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1136. QUEUE\_62\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.847 QUEUE\_62\_STATUS\_C Register (offset = 33E8h) [reset = 0h]**

QUEUE\_62\_STATUS\_C is shown in [Figure 16-1123](#) and described in [Table 16-1137](#).

**Figure 16-1123. QUEUE\_62\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1137. QUEUE\_62\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.848 QUEUE\_63\_STATUS\_A Register (offset = 33F0h) [reset = 0h]**

QUEUE\_63\_STATUS\_A is shown in [Figure 16-1124](#) and described in [Table 16-1138](#).

**Figure 16-1124. QUEUE\_63\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1138. QUEUE\_63\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.849 QUEUE\_63\_STATUS\_B Register (offset = 33F4h) [reset = 0h]**

QUEUE\_63\_STATUS\_B is shown in [Figure 16-1125](#) and described in [Table 16-1139](#).

**Figure 16-1125. QUEUE\_63\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1139. QUEUE\_63\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.850 QUEUE\_63\_STATUS\_C Register (offset = 33F8h) [reset = 0h]**

QUEUE\_63\_STATUS\_C is shown in [Figure 16-1126](#) and described in [Table 16-1140](#).

**Figure 16-1126. QUEUE\_63\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1140. QUEUE\_63\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.851 QUEUE\_64\_STATUS\_A Register (offset = 3400h) [reset = 0h]**

QUEUE\_64\_STATUS\_A is shown in [Figure 16-1127](#) and described in [Table 16-1141](#).

**Figure 16-1127. QUEUE\_64\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1141. QUEUE\_64\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.852 QUEUE\_64\_STATUS\_B Register (offset = 3404h) [reset = 0h]**

QUEUE\_64\_STATUS\_B is shown in [Figure 16-1128](#) and described in [Table 16-1142](#).

**Figure 16-1128. QUEUE\_64\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1142. QUEUE\_64\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.853 QUEUE\_64\_STATUS\_C Register (offset = 3408h) [reset = 0h]**

QUEUE\_64\_STATUS\_C is shown in [Figure 16-1129](#) and described in [Table 16-1143](#).

**Figure 16-1129. QUEUE\_64\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1143. QUEUE\_64\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.854 QUEUE\_65\_STATUS\_A Register (offset = 3410h) [reset = 0h]**

QUEUE\_65\_STATUS\_A is shown in [Figure 16-1130](#) and described in [Table 16-1144](#).

**Figure 16-1130. QUEUE\_65\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1144. QUEUE\_65\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.855 QUEUE\_65\_STATUS\_B Register (offset = 3414h) [reset = 0h]**

QUEUE\_65\_STATUS\_B is shown in [Figure 16-1131](#) and described in [Table 16-1145](#).

**Figure 16-1131. QUEUE\_65\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1145. QUEUE\_65\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.856 QUEUE\_65\_STATUS\_C Register (offset = 3418h) [reset = 0h]**

QUEUE\_65\_STATUS\_C is shown in [Figure 16-1132](#) and described in [Table 16-1146](#).

**Figure 16-1132. QUEUE\_65\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1146. QUEUE\_65\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.857 QUEUE\_66\_STATUS\_A Register (offset = 3420h) [reset = 0h]**

QUEUE\_66\_STATUS\_A is shown in [Figure 16-1133](#) and described in [Table 16-1147](#).

**Figure 16-1133. QUEUE\_66\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1147. QUEUE\_66\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.858 QUEUE\_66\_STATUS\_B Register (offset = 3424h) [reset = 0h]**

QUEUE\_66\_STATUS\_B is shown in [Figure 16-1134](#) and described in [Table 16-1148](#).

**Figure 16-1134. QUEUE\_66\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1148. QUEUE\_66\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.859 QUEUE\_66\_STATUS\_C Register (offset = 3428h) [reset = 0h]**

QUEUE\_66\_STATUS\_C is shown in [Figure 16-1135](#) and described in [Table 16-1149](#).

**Figure 16-1135. QUEUE\_66\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1149. QUEUE\_66\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.860 QUEUE\_67\_STATUS\_A Register (offset = 3430h) [reset = 0h]**

QUEUE\_67\_STATUS\_A is shown in [Figure 16-1136](#) and described in [Table 16-1150](#).

**Figure 16-1136. QUEUE\_67\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1150. QUEUE\_67\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.861 QUEUE\_67\_STATUS\_B Register (offset = 3434h) [reset = 0h]**

QUEUE\_67\_STATUS\_B is shown in [Figure 16-1137](#) and described in [Table 16-1151](#).

**Figure 16-1137. QUEUE\_67\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1151. QUEUE\_67\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.862 QUEUE\_67\_STATUS\_C Register (offset = 3438h) [reset = 0h]**

QUEUE\_67\_STATUS\_C is shown in [Figure 16-1138](#) and described in [Table 16-1152](#).

**Figure 16-1138. QUEUE\_67\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1152. QUEUE\_67\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.863 QUEUE\_68\_STATUS\_A Register (offset = 3440h) [reset = 0h]**

QUEUE\_68\_STATUS\_A is shown in [Figure 16-1139](#) and described in [Table 16-1153](#).

**Figure 16-1139. QUEUE\_68\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1153. QUEUE\_68\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.864 QUEUE\_68\_STATUS\_B Register (offset = 3444h) [reset = 0h]**

QUEUE\_68\_STATUS\_B is shown in [Figure 16-1140](#) and described in [Table 16-1154](#).

**Figure 16-1140. QUEUE\_68\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1154. QUEUE\_68\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.865 QUEUE\_68\_STATUS\_C Register (offset = 3448h) [reset = 0h]**

QUEUE\_68\_STATUS\_C is shown in [Figure 16-1141](#) and described in [Table 16-1155](#).

**Figure 16-1141. QUEUE\_68\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1155. QUEUE\_68\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.866 QUEUE\_69\_STATUS\_A Register (offset = 3450h) [reset = 0h]**

QUEUE\_69\_STATUS\_A is shown in [Figure 16-1142](#) and described in [Table 16-1156](#).

**Figure 16-1142. QUEUE\_69\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1156. QUEUE\_69\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.867 QUEUE\_69\_STATUS\_B Register (offset = 3454h) [reset = 0h]**

QUEUE\_69\_STATUS\_B is shown in [Figure 16-1143](#) and described in [Table 16-1157](#).

**Figure 16-1143. QUEUE\_69\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1157. QUEUE\_69\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.868 QUEUE\_69\_STATUS\_C Register (offset = 3458h) [reset = 0h]**

QUEUE\_69\_STATUS\_C is shown in [Figure 16-1144](#) and described in [Table 16-1158](#).

**Figure 16-1144. QUEUE\_69\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1158. QUEUE\_69\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.869 QUEUE\_70\_STATUS\_A Register (offset = 3460h) [reset = 0h]**

QUEUE\_70\_STATUS\_A is shown in [Figure 16-1145](#) and described in [Table 16-1159](#).

**Figure 16-1145. QUEUE\_70\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1159. QUEUE\_70\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.870 QUEUE\_70\_STATUS\_B Register (offset = 3464h) [reset = 0h]**

QUEUE\_70\_STATUS\_B is shown in [Figure 16-1146](#) and described in [Table 16-1160](#).

**Figure 16-1146. QUEUE\_70\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1160. QUEUE\_70\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.871 QUEUE\_70\_STATUS\_C Register (offset = 3468h) [reset = 0h]**

QUEUE\_70\_STATUS\_C is shown in [Figure 16-1147](#) and described in [Table 16-1161](#).

**Figure 16-1147. QUEUE\_70\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1161. QUEUE\_70\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.872 QUEUE\_71\_STATUS\_A Register (offset = 3470h) [reset = 0h]**

QUEUE\_71\_STATUS\_A is shown in [Figure 16-1148](#) and described in [Table 16-1162](#).

**Figure 16-1148. QUEUE\_71\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1162. QUEUE\_71\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.873 QUEUE\_71\_STATUS\_B Register (offset = 3474h) [reset = 0h]**

QUEUE\_71\_STATUS\_B is shown in [Figure 16-1149](#) and described in [Table 16-1163](#).

**Figure 16-1149. QUEUE\_71\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1163. QUEUE\_71\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.874 QUEUE\_71\_STATUS\_C Register (offset = 3478h) [reset = 0h]**

QUEUE\_71\_STATUS\_C is shown in [Figure 16-1150](#) and described in [Table 16-1164](#).

**Figure 16-1150. QUEUE\_71\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1164. QUEUE\_71\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.875 QUEUE\_72\_STATUS\_A Register (offset = 3480h) [reset = 0h]**

QUEUE\_72\_STATUS\_A is shown in [Figure 16-1151](#) and described in [Table 16-1165](#).

**Figure 16-1151. QUEUE\_72\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1165. QUEUE\_72\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.876 QUEUE\_72\_STATUS\_B Register (offset = 3484h) [reset = 0h]**

QUEUE\_72\_STATUS\_B is shown in [Figure 16-1152](#) and described in [Table 16-1166](#).

**Figure 16-1152. QUEUE\_72\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1166. QUEUE\_72\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.877 QUEUE\_72\_STATUS\_C Register (offset = 3488h) [reset = 0h]**

QUEUE\_72\_STATUS\_C is shown in [Figure 16-1153](#) and described in [Table 16-1167](#).

**Figure 16-1153. QUEUE\_72\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1167. QUEUE\_72\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.878 QUEUE\_73\_STATUS\_A Register (offset = 3490h) [reset = 0h]**

QUEUE\_73\_STATUS\_A is shown in [Figure 16-1154](#) and described in [Table 16-1168](#).

**Figure 16-1154. QUEUE\_73\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1168. QUEUE\_73\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.879 QUEUE\_73\_STATUS\_B Register (offset = 3494h) [reset = 0h]**

QUEUE\_73\_STATUS\_B is shown in [Figure 16-1155](#) and described in [Table 16-1169](#).

**Figure 16-1155. QUEUE\_73\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1169. QUEUE\_73\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.880 QUEUE\_73\_STATUS\_C Register (offset = 3498h) [reset = 0h]**

QUEUE\_73\_STATUS\_C is shown in [Figure 16-1156](#) and described in [Table 16-1170](#).

**Figure 16-1156. QUEUE\_73\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1170. QUEUE\_73\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.881 QUEUE\_74\_STATUS\_A Register (offset = 34A0h) [reset = 0h]**

QUEUE\_74\_STATUS\_A is shown in [Figure 16-1157](#) and described in [Table 16-1171](#).

**Figure 16-1157. QUEUE\_74\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1171. QUEUE\_74\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.882 QUEUE\_74\_STATUS\_B Register (offset = 34A4h) [reset = 0h]**

QUEUE\_74\_STATUS\_B is shown in [Figure 16-1158](#) and described in [Table 16-1172](#).

**Figure 16-1158. QUEUE\_74\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1172. QUEUE\_74\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.883 QUEUE\_74\_STATUS\_C Register (offset = 34A8h) [reset = 0h]**

QUEUE\_74\_STATUS\_C is shown in [Figure 16-1159](#) and described in [Table 16-1173](#).

**Figure 16-1159. QUEUE\_74\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1173. QUEUE\_74\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.884 QUEUE\_75\_STATUS\_A Register (offset = 34B0h) [reset = 0h]**

QUEUE\_75\_STATUS\_A is shown in [Figure 16-1160](#) and described in [Table 16-1174](#).

**Figure 16-1160. QUEUE\_75\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1174. QUEUE\_75\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.885 QUEUE\_75\_STATUS\_B Register (offset = 34B4h) [reset = 0h]**

QUEUE\_75\_STATUS\_B is shown in [Figure 16-1161](#) and described in [Table 16-1175](#).

**Figure 16-1161. QUEUE\_75\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1175. QUEUE\_75\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.886 QUEUE\_75\_STATUS\_C Register (offset = 34B8h) [reset = 0h]**

QUEUE\_75\_STATUS\_C is shown in [Figure 16-1162](#) and described in [Table 16-1176](#).

**Figure 16-1162. QUEUE\_75\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1176. QUEUE\_75\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.887 QUEUE\_76\_STATUS\_A Register (offset = 34C0h) [reset = 0h]**

QUEUE\_76\_STATUS\_A is shown in [Figure 16-1163](#) and described in [Table 16-1177](#).

**Figure 16-1163. QUEUE\_76\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1177. QUEUE\_76\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.888 QUEUE\_76\_STATUS\_B Register (offset = 34C4h) [reset = 0h]**

QUEUE\_76\_STATUS\_B is shown in [Figure 16-1164](#) and described in [Table 16-1178](#).

**Figure 16-1164. QUEUE\_76\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1178. QUEUE\_76\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.889 QUEUE\_76\_STATUS\_C Register (offset = 34C8h) [reset = 0h]**

QUEUE\_76\_STATUS\_C is shown in [Figure 16-1165](#) and described in [Table 16-1179](#).

**Figure 16-1165. QUEUE\_76\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1179. QUEUE\_76\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.890 QUEUE\_77\_STATUS\_A Register (offset = 34D0h) [reset = 0h]**

QUEUE\_77\_STATUS\_A is shown in [Figure 16-1166](#) and described in [Table 16-1180](#).

**Figure 16-1166. QUEUE\_77\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1180. QUEUE\_77\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.891 QUEUE\_77\_STATUS\_B Register (offset = 34D4h) [reset = 0h]**

QUEUE\_77\_STATUS\_B is shown in [Figure 16-1167](#) and described in [Table 16-1181](#).

**Figure 16-1167. QUEUE\_77\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1181. QUEUE\_77\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.892 QUEUE\_77\_STATUS\_C Register (offset = 34D8h) [reset = 0h]**

QUEUE\_77\_STATUS\_C is shown in [Figure 16-1168](#) and described in [Table 16-1182](#).

**Figure 16-1168. QUEUE\_77\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1182. QUEUE\_77\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.893 QUEUE\_78\_STATUS\_A Register (offset = 34E0h) [reset = 0h]**

QUEUE\_78\_STATUS\_A is shown in [Figure 16-1169](#) and described in [Table 16-1183](#).

**Figure 16-1169. QUEUE\_78\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1183. QUEUE\_78\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.894 QUEUE\_78\_STATUS\_B Register (offset = 34E4h) [reset = 0h]**

QUEUE\_78\_STATUS\_B is shown in [Figure 16-1170](#) and described in [Table 16-1184](#).

**Figure 16-1170. QUEUE\_78\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1184. QUEUE\_78\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.895 QUEUE\_78\_STATUS\_C Register (offset = 34E8h) [reset = 0h]**

QUEUE\_78\_STATUS\_C is shown in [Figure 16-1171](#) and described in [Table 16-1185](#).

**Figure 16-1171. QUEUE\_78\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1185. QUEUE\_78\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.896 QUEUE\_79\_STATUS\_A Register (offset = 34F0h) [reset = 0h]**

QUEUE\_79\_STATUS\_A is shown in [Figure 16-1172](#) and described in [Table 16-1186](#).

**Figure 16-1172. QUEUE\_79\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1186. QUEUE\_79\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.897 QUEUE\_79\_STATUS\_B Register (offset = 34F4h) [reset = 0h]**

QUEUE\_79\_STATUS\_B is shown in [Figure 16-1173](#) and described in [Table 16-1187](#).

**Figure 16-1173. QUEUE\_79\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1187. QUEUE\_79\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.898 QUEUE\_79\_STATUS\_C Register (offset = 34F8h) [reset = 0h]**

QUEUE\_79\_STATUS\_C is shown in [Figure 16-1174](#) and described in [Table 16-1188](#).

**Figure 16-1174. QUEUE\_79\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1188. QUEUE\_79\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.899 QUEUE\_80\_STATUS\_A Register (offset = 3500h) [reset = 0h]**

QUEUE\_80\_STATUS\_A is shown in [Figure 16-1175](#) and described in [Table 16-1189](#).

**Figure 16-1175. QUEUE\_80\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1189. QUEUE\_80\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.900 QUEUE\_80\_STATUS\_B Register (offset = 3504h) [reset = 0h]**

QUEUE\_80\_STATUS\_B is shown in [Figure 16-1176](#) and described in [Table 16-1190](#).

**Figure 16-1176. QUEUE\_80\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1190. QUEUE\_80\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.901 QUEUE\_80\_STATUS\_C Register (offset = 3508h) [reset = 0h]**

QUEUE\_80\_STATUS\_C is shown in [Figure 16-1177](#) and described in [Table 16-1191](#).

**Figure 16-1177. QUEUE\_80\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1191. QUEUE\_80\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.902 QUEUE\_81\_STATUS\_A Register (offset = 3510h) [reset = 0h]**

QUEUE\_81\_STATUS\_A is shown in [Figure 16-1178](#) and described in [Table 16-1192](#).

**Figure 16-1178. QUEUE\_81\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1192. QUEUE\_81\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.903 QUEUE\_81\_STATUS\_B Register (offset = 3514h) [reset = 0h]**

QUEUE\_81\_STATUS\_B is shown in [Figure 16-1179](#) and described in [Table 16-1193](#).

**Figure 16-1179. QUEUE\_81\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1193. QUEUE\_81\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.904 QUEUE\_81\_STATUS\_C Register (offset = 3518h) [reset = 0h]**

QUEUE\_81\_STATUS\_C is shown in [Figure 16-1180](#) and described in [Table 16-1194](#).

**Figure 16-1180. QUEUE\_81\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1194. QUEUE\_81\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.905 QUEUE\_82\_STATUS\_A Register (offset = 3520h) [reset = 0h]**

QUEUE\_82\_STATUS\_A is shown in [Figure 16-1181](#) and described in [Table 16-1195](#).

**Figure 16-1181. QUEUE\_82\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1195. QUEUE\_82\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.906 QUEUE\_82\_STATUS\_B Register (offset = 3524h) [reset = 0h]**

QUEUE\_82\_STATUS\_B is shown in [Figure 16-1182](#) and described in [Table 16-1196](#).

**Figure 16-1182. QUEUE\_82\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1196. QUEUE\_82\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.907 QUEUE\_82\_STATUS\_C Register (offset = 3528h) [reset = 0h]**

QUEUE\_82\_STATUS\_C is shown in [Figure 16-1183](#) and described in [Table 16-1197](#).

**Figure 16-1183. QUEUE\_82\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1197. QUEUE\_82\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.908 QUEUE\_83\_STATUS\_A Register (offset = 3530h) [reset = 0h]**

QUEUE\_83\_STATUS\_A is shown in [Figure 16-1184](#) and described in [Table 16-1198](#).

**Figure 16-1184. QUEUE\_83\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1198. QUEUE\_83\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.909 QUEUE\_83\_STATUS\_B Register (offset = 3534h) [reset = 0h]**

QUEUE\_83\_STATUS\_B is shown in [Figure 16-1185](#) and described in [Table 16-1199](#).

**Figure 16-1185. QUEUE\_83\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1199. QUEUE\_83\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.910 QUEUE\_83\_STATUS\_C Register (offset = 3538h) [reset = 0h]**

QUEUE\_83\_STATUS\_C is shown in [Figure 16-1186](#) and described in [Table 16-1200](#).

**Figure 16-1186. QUEUE\_83\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1200. QUEUE\_83\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.911 QUEUE\_84\_STATUS\_A Register (offset = 3540h) [reset = 0h]**

QUEUE\_84\_STATUS\_A is shown in [Figure 16-1187](#) and described in [Table 16-1201](#).

**Figure 16-1187. QUEUE\_84\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1201. QUEUE\_84\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.912 QUEUE\_84\_STATUS\_B Register (offset = 3544h) [reset = 0h]**

QUEUE\_84\_STATUS\_B is shown in [Figure 16-1188](#) and described in [Table 16-1202](#).

**Figure 16-1188. QUEUE\_84\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1202. QUEUE\_84\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.913 QUEUE\_84\_STATUS\_C Register (offset = 3548h) [reset = 0h]**

QUEUE\_84\_STATUS\_C is shown in [Figure 16-1189](#) and described in [Table 16-1203](#).

**Figure 16-1189. QUEUE\_84\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1203. QUEUE\_84\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.914 QUEUE\_85\_STATUS\_A Register (offset = 3550h) [reset = 0h]**

QUEUE\_85\_STATUS\_A is shown in [Figure 16-1190](#) and described in [Table 16-1204](#).

**Figure 16-1190. QUEUE\_85\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1204. QUEUE\_85\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.915 QUEUE\_85\_STATUS\_B Register (offset = 3554h) [reset = 0h]**

QUEUE\_85\_STATUS\_B is shown in [Figure 16-1191](#) and described in [Table 16-1205](#).

**Figure 16-1191. QUEUE\_85\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1205. QUEUE\_85\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.916 QUEUE\_85\_STATUS\_C Register (offset = 3558h) [reset = 0h]**

QUEUE\_85\_STATUS\_C is shown in [Figure 16-1192](#) and described in [Table 16-1206](#).

**Figure 16-1192. QUEUE\_85\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1206. QUEUE\_85\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.917 QUEUE\_86\_STATUS\_A Register (offset = 3560h) [reset = 0h]**

QUEUE\_86\_STATUS\_A is shown in [Figure 16-1193](#) and described in [Table 16-1207](#).

**Figure 16-1193. QUEUE\_86\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1207. QUEUE\_86\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.918 QUEUE\_86\_STATUS\_B Register (offset = 3564h) [reset = 0h]**

QUEUE\_86\_STATUS\_B is shown in [Figure 16-1194](#) and described in [Table 16-1208](#).

**Figure 16-1194. QUEUE\_86\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1208. QUEUE\_86\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.919 QUEUE\_86\_STATUS\_C Register (offset = 3568h) [reset = 0h]**

QUEUE\_86\_STATUS\_C is shown in [Figure 16-1195](#) and described in [Table 16-1209](#).

**Figure 16-1195. QUEUE\_86\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1209. QUEUE\_86\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.920 QUEUE\_87\_STATUS\_A Register (offset = 3570h) [reset = 0h]**

QUEUE\_87\_STATUS\_A is shown in [Figure 16-1196](#) and described in [Table 16-1210](#).

**Figure 16-1196. QUEUE\_87\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1210. QUEUE\_87\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.921 QUEUE\_87\_STATUS\_B Register (offset = 3574h) [reset = 0h]**

QUEUE\_87\_STATUS\_B is shown in [Figure 16-1197](#) and described in [Table 16-1211](#).

**Figure 16-1197. QUEUE\_87\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1211. QUEUE\_87\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.922 QUEUE\_87\_STATUS\_C Register (offset = 3578h) [reset = 0h]**

QUEUE\_87\_STATUS\_C is shown in [Figure 16-1198](#) and described in [Table 16-1212](#).

**Figure 16-1198. QUEUE\_87\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1212. QUEUE\_87\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.923 QUEUE\_88\_STATUS\_A Register (offset = 3580h) [reset = 0h]**

QUEUE\_88\_STATUS\_A is shown in [Figure 16-1199](#) and described in [Table 16-1213](#).

**Figure 16-1199. QUEUE\_88\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1213. QUEUE\_88\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.924 QUEUE\_88\_STATUS\_B Register (offset = 3584h) [reset = 0h]**

QUEUE\_88\_STATUS\_B is shown in [Figure 16-1200](#) and described in [Table 16-1214](#).

**Figure 16-1200. QUEUE\_88\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1214. QUEUE\_88\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.925 QUEUE\_88\_STATUS\_C Register (offset = 3588h) [reset = 0h]**

QUEUE\_88\_STATUS\_C is shown in [Figure 16-1201](#) and described in [Table 16-1215](#).

**Figure 16-1201. QUEUE\_88\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1215. QUEUE\_88\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.926 QUEUE\_89\_STATUS\_A Register (offset = 3590h) [reset = 0h]**

QUEUE\_89\_STATUS\_A is shown in [Figure 16-1202](#) and described in [Table 16-1216](#).

**Figure 16-1202. QUEUE\_89\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1216. QUEUE\_89\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.927 QUEUE\_89\_STATUS\_B Register (offset = 3594h) [reset = 0h]**

QUEUE\_89\_STATUS\_B is shown in [Figure 16-1203](#) and described in [Table 16-1217](#).

**Figure 16-1203. QUEUE\_89\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1217. QUEUE\_89\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.928 QUEUE\_89\_STATUS\_C Register (offset = 3598h) [reset = 0h]**

QUEUE\_89\_STATUS\_C is shown in [Figure 16-1204](#) and described in [Table 16-1218](#).

**Figure 16-1204. QUEUE\_89\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1218. QUEUE\_89\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.929 QUEUE\_90\_STATUS\_A Register (offset = 35A0h) [reset = 0h]**

QUEUE\_90\_STATUS\_A is shown in [Figure 16-1205](#) and described in [Table 16-1219](#).

**Figure 16-1205. QUEUE\_90\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1219. QUEUE\_90\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.930 QUEUE\_90\_STATUS\_B Register (offset = 35A4h) [reset = 0h]**

QUEUE\_90\_STATUS\_B is shown in [Figure 16-1206](#) and described in [Table 16-1220](#).

**Figure 16-1206. QUEUE\_90\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1220. QUEUE\_90\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.931 QUEUE\_90\_STATUS\_C Register (offset = 35A8h) [reset = 0h]**

QUEUE\_90\_STATUS\_C is shown in [Figure 16-1207](#) and described in [Table 16-1221](#).

**Figure 16-1207. QUEUE\_90\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1221. QUEUE\_90\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.932 QUEUE\_91\_STATUS\_A Register (offset = 35B0h) [reset = 0h]**

QUEUE\_91\_STATUS\_A is shown in [Figure 16-1208](#) and described in [Table 16-1222](#).

**Figure 16-1208. QUEUE\_91\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1222. QUEUE\_91\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.933 QUEUE\_91\_STATUS\_B Register (offset = 35B4h) [reset = 0h]**

QUEUE\_91\_STATUS\_B is shown in [Figure 16-1209](#) and described in [Table 16-1223](#).

**Figure 16-1209. QUEUE\_91\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1223. QUEUE\_91\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.934 QUEUE\_91\_STATUS\_C Register (offset = 35B8h) [reset = 0h]**

QUEUE\_91\_STATUS\_C is shown in [Figure 16-1210](#) and described in [Table 16-1224](#).

**Figure 16-1210. QUEUE\_91\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1224. QUEUE\_91\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.935 QUEUE\_92\_STATUS\_A Register (offset = 35C0h) [reset = 0h]**

QUEUE\_92\_STATUS\_A is shown in [Figure 16-1211](#) and described in [Table 16-1225](#).

**Figure 16-1211. QUEUE\_92\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1225. QUEUE\_92\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.936 QUEUE\_92\_STATUS\_B Register (offset = 35C4h) [reset = 0h]**

QUEUE\_92\_STATUS\_B is shown in [Figure 16-1212](#) and described in [Table 16-1226](#).

**Figure 16-1212. QUEUE\_92\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1226. QUEUE\_92\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.937 QUEUE\_92\_STATUS\_C Register (offset = 35C8h) [reset = 0h]**

QUEUE\_92\_STATUS\_C is shown in [Figure 16-1213](#) and described in [Table 16-1227](#).

**Figure 16-1213. QUEUE\_92\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1227. QUEUE\_92\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.938 QUEUE\_93\_STATUS\_A Register (offset = 35D0h) [reset = 0h]**

QUEUE\_93\_STATUS\_A is shown in [Figure 16-1214](#) and described in [Table 16-1228](#).

**Figure 16-1214. QUEUE\_93\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1228. QUEUE\_93\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.939 QUEUE\_93\_STATUS\_B Register (offset = 35D4h) [reset = 0h]**

QUEUE\_93\_STATUS\_B is shown in [Figure 16-1215](#) and described in [Table 16-1229](#).

**Figure 16-1215. QUEUE\_93\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1229. QUEUE\_93\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.940 QUEUE\_93\_STATUS\_C Register (offset = 35D8h) [reset = 0h]**

QUEUE\_93\_STATUS\_C is shown in [Figure 16-1216](#) and described in [Table 16-1230](#).

**Figure 16-1216. QUEUE\_93\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1230. QUEUE\_93\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.941 QUEUE\_94\_STATUS\_A Register (offset = 35E0h) [reset = 0h]**

QUEUE\_94\_STATUS\_A is shown in [Figure 16-1217](#) and described in [Table 16-1231](#).

**Figure 16-1217. QUEUE\_94\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1231. QUEUE\_94\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.942 QUEUE\_94\_STATUS\_B Register (offset = 35E4h) [reset = 0h]**

QUEUE\_94\_STATUS\_B is shown in [Figure 16-1218](#) and described in [Table 16-1232](#).

**Figure 16-1218. QUEUE\_94\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1232. QUEUE\_94\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.943 QUEUE\_94\_STATUS\_C Register (offset = 35E8h) [reset = 0h]**

QUEUE\_94\_STATUS\_C is shown in [Figure 16-1219](#) and described in [Table 16-1233](#).

**Figure 16-1219. QUEUE\_94\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1233. QUEUE\_94\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.944 QUEUE\_95\_STATUS\_A Register (offset = 35F0h) [reset = 0h]**

QUEUE\_95\_STATUS\_A is shown in [Figure 16-1220](#) and described in [Table 16-1234](#).

**Figure 16-1220. QUEUE\_95\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1234. QUEUE\_95\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.945 QUEUE\_95\_STATUS\_B Register (offset = 35F4h) [reset = 0h]**

QUEUE\_95\_STATUS\_B is shown in [Figure 16-1221](#) and described in [Table 16-1235](#).

**Figure 16-1221. QUEUE\_95\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1235. QUEUE\_95\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.946 QUEUE\_95\_STATUS\_C Register (offset = 35F8h) [reset = 0h]**

QUEUE\_95\_STATUS\_C is shown in [Figure 16-1222](#) and described in [Table 16-1236](#).

**Figure 16-1222. QUEUE\_95\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1236. QUEUE\_95\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.947 QUEUE\_96\_STATUS\_A Register (offset = 3600h) [reset = 0h]**

QUEUE\_96\_STATUS\_A is shown in [Figure 16-1223](#) and described in [Table 16-1237](#).

**Figure 16-1223. QUEUE\_96\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1237. QUEUE\_96\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.948 QUEUE\_96\_STATUS\_B Register (offset = 3604h) [reset = 0h]**

QUEUE\_96\_STATUS\_B is shown in [Figure 16-1224](#) and described in [Table 16-1238](#).

**Figure 16-1224. QUEUE\_96\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1238. QUEUE\_96\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.949 QUEUE\_96\_STATUS\_C Register (offset = 3608h) [reset = 0h]**

QUEUE\_96\_STATUS\_C is shown in [Figure 16-1225](#) and described in [Table 16-1239](#).

**Figure 16-1225. QUEUE\_96\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1239. QUEUE\_96\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.950 QUEUE\_97\_STATUS\_A Register (offset = 3610h) [reset = 0h]**

QUEUE\_97\_STATUS\_A is shown in [Figure 16-1226](#) and described in [Table 16-1240](#).

**Figure 16-1226. QUEUE\_97\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1240. QUEUE\_97\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.951 QUEUE\_97\_STATUS\_B Register (offset = 3614h) [reset = 0h]**

QUEUE\_97\_STATUS\_B is shown in [Figure 16-1227](#) and described in [Table 16-1241](#).

**Figure 16-1227. QUEUE\_97\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1241. QUEUE\_97\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.952 QUEUE\_97\_STATUS\_C Register (offset = 3618h) [reset = 0h]**

QUEUE\_97\_STATUS\_C is shown in [Figure 16-1228](#) and described in [Table 16-1242](#).

**Figure 16-1228. QUEUE\_97\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1242. QUEUE\_97\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.953 QUEUE\_98\_STATUS\_A Register (offset = 3620h) [reset = 0h]**

QUEUE\_98\_STATUS\_A is shown in [Figure 16-1229](#) and described in [Table 16-1243](#).

**Figure 16-1229. QUEUE\_98\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1243. QUEUE\_98\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.954 QUEUE\_98\_STATUS\_B Register (offset = 3624h) [reset = 0h]**

QUEUE\_98\_STATUS\_B is shown in [Figure 16-1230](#) and described in [Table 16-1244](#).

**Figure 16-1230. QUEUE\_98\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1244. QUEUE\_98\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.955 QUEUE\_98\_STATUS\_C Register (offset = 3628h) [reset = 0h]**

QUEUE\_98\_STATUS\_C is shown in [Figure 16-1231](#) and described in [Table 16-1245](#).

**Figure 16-1231. QUEUE\_98\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1245. QUEUE\_98\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.956 QUEUE\_99\_STATUS\_A Register (offset = 3630h) [reset = 0h]**

QUEUE\_99\_STATUS\_A is shown in [Figure 16-1232](#) and described in [Table 16-1246](#).

**Figure 16-1232. QUEUE\_99\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1246. QUEUE\_99\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.957 QUEUE\_99\_STATUS\_B Register (offset = 3634h) [reset = 0h]**

QUEUE\_99\_STATUS\_B is shown in [Figure 16-1233](#) and described in [Table 16-1247](#).

**Figure 16-1233. QUEUE\_99\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1247. QUEUE\_99\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.958 QUEUE\_99\_STATUS\_C Register (offset = 3638h) [reset = 0h]**

QUEUE\_99\_STATUS\_C is shown in [Figure 16-1234](#) and described in [Table 16-1248](#).

**Figure 16-1234. QUEUE\_99\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1248. QUEUE\_99\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.959 QUEUE\_100\_STATUS\_A Register (offset = 3640h) [reset = 0h]**

QUEUE\_100\_STATUS\_A is shown in [Figure 16-1235](#) and described in [Table 16-1249](#).

**Figure 16-1235. QUEUE\_100\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1249. QUEUE\_100\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.960 QUEUE\_100\_STATUS\_B Register (offset = 3644h) [reset = 0h]**

QUEUE\_100\_STATUS\_B is shown in [Figure 16-1236](#) and described in [Table 16-1250](#).

**Figure 16-1236. QUEUE\_100\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1250. QUEUE\_100\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.961 QUEUE\_100\_STATUS\_C Register (offset = 3648h) [reset = 0h]**

QUEUE\_100\_STATUS\_C is shown in [Figure 16-1237](#) and described in [Table 16-1251](#).

**Figure 16-1237. QUEUE\_100\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1251. QUEUE\_100\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.962 QUEUE\_101\_STATUS\_A Register (offset = 3650h) [reset = 0h]**

QUEUE\_101\_STATUS\_A is shown in [Figure 16-1238](#) and described in [Table 16-1252](#).

**Figure 16-1238. QUEUE\_101\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1252. QUEUE\_101\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.963 QUEUE\_101\_STATUS\_B Register (offset = 3654h) [reset = 0h]**

QUEUE\_101\_STATUS\_B is shown in [Figure 16-1239](#) and described in [Table 16-1253](#).

**Figure 16-1239. QUEUE\_101\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1253. QUEUE\_101\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.964 QUEUE\_101\_STATUS\_C Register (offset = 3658h) [reset = 0h]**

QUEUE\_101\_STATUS\_C is shown in [Figure 16-1240](#) and described in [Table 16-1254](#).

**Figure 16-1240. QUEUE\_101\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1254. QUEUE\_101\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.965 QUEUE\_102\_STATUS\_A Register (offset = 3660h) [reset = 0h]**

QUEUE\_102\_STATUS\_A is shown in [Figure 16-1241](#) and described in [Table 16-1255](#).

**Figure 16-1241. QUEUE\_102\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1255. QUEUE\_102\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.966 QUEUE\_102\_STATUS\_B Register (offset = 3664h) [reset = 0h]**

QUEUE\_102\_STATUS\_B is shown in [Figure 16-1242](#) and described in [Table 16-1256](#).

**Figure 16-1242. QUEUE\_102\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1256. QUEUE\_102\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.967 QUEUE\_102\_STATUS\_C Register (offset = 3668h) [reset = 0h]**

QUEUE\_102\_STATUS\_C is shown in [Figure 16-1243](#) and described in [Table 16-1257](#).

**Figure 16-1243. QUEUE\_102\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1257. QUEUE\_102\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.968 QUEUE\_103\_STATUS\_A Register (offset = 3670h) [reset = 0h]**

QUEUE\_103\_STATUS\_A is shown in [Figure 16-1244](#) and described in [Table 16-1258](#).

**Figure 16-1244. QUEUE\_103\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1258. QUEUE\_103\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.969 QUEUE\_103\_STATUS\_B Register (offset = 3674h) [reset = 0h]**

QUEUE\_103\_STATUS\_B is shown in [Figure 16-1245](#) and described in [Table 16-1259](#).

**Figure 16-1245. QUEUE\_103\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1259. QUEUE\_103\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.970 QUEUE\_103\_STATUS\_C Register (offset = 3678h) [reset = 0h]**

QUEUE\_103\_STATUS\_C is shown in [Figure 16-1246](#) and described in [Table 16-1260](#).

**Figure 16-1246. QUEUE\_103\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1260. QUEUE\_103\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.971 QUEUE\_104\_STATUS\_A Register (offset = 3680h) [reset = 0h]**

QUEUE\_104\_STATUS\_A is shown in [Figure 16-1247](#) and described in [Table 16-1261](#).

**Figure 16-1247. QUEUE\_104\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1261. QUEUE\_104\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.972 QUEUE\_104\_STATUS\_B Register (offset = 3684h) [reset = 0h]**

QUEUE\_104\_STATUS\_B is shown in [Figure 16-1248](#) and described in [Table 16-1262](#).

**Figure 16-1248. QUEUE\_104\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1262. QUEUE\_104\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.973 QUEUE\_104\_STATUS\_C Register (offset = 3688h) [reset = 0h]**

QUEUE\_104\_STATUS\_C is shown in [Figure 16-1249](#) and described in [Table 16-1263](#).

**Figure 16-1249. QUEUE\_104\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1263. QUEUE\_104\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.974 QUEUE\_105\_STATUS\_A Register (offset = 3690h) [reset = 0h]**

QUEUE\_105\_STATUS\_A is shown in [Figure 16-1250](#) and described in [Table 16-1264](#).

**Figure 16-1250. QUEUE\_105\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1264. QUEUE\_105\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.975 QUEUE\_105\_STATUS\_B Register (offset = 3694h) [reset = 0h]**

QUEUE\_105\_STATUS\_B is shown in [Figure 16-1251](#) and described in [Table 16-1265](#).

**Figure 16-1251. QUEUE\_105\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1265. QUEUE\_105\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.976 QUEUE\_105\_STATUS\_C Register (offset = 3698h) [reset = 0h]**

QUEUE\_105\_STATUS\_C is shown in [Figure 16-1252](#) and described in [Table 16-1266](#).

**Figure 16-1252. QUEUE\_105\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1266. QUEUE\_105\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.977 QUEUE\_106\_STATUS\_A Register (offset = 36A0h) [reset = 0h]**

QUEUE\_106\_STATUS\_A is shown in [Figure 16-1253](#) and described in [Table 16-1267](#).

**Figure 16-1253. QUEUE\_106\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1267. QUEUE\_106\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.978 QUEUE\_106\_STATUS\_B Register (offset = 36A4h) [reset = 0h]**

QUEUE\_106\_STATUS\_B is shown in [Figure 16-1254](#) and described in [Table 16-1268](#).

**Figure 16-1254. QUEUE\_106\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1268. QUEUE\_106\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.979 QUEUE\_106\_STATUS\_C Register (offset = 36A8h) [reset = 0h]**

QUEUE\_106\_STATUS\_C is shown in [Figure 16-1255](#) and described in [Table 16-1269](#).

**Figure 16-1255. QUEUE\_106\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1269. QUEUE\_106\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.980 QUEUE\_107\_STATUS\_A Register (offset = 36B0h) [reset = 0h]**

QUEUE\_107\_STATUS\_A is shown in [Figure 16-1256](#) and described in [Table 16-1270](#).

**Figure 16-1256. QUEUE\_107\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1270. QUEUE\_107\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.981 QUEUE\_107\_STATUS\_B Register (offset = 36B4h) [reset = 0h]**

QUEUE\_107\_STATUS\_B is shown in [Figure 16-1257](#) and described in [Table 16-1271](#).

**Figure 16-1257. QUEUE\_107\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1271. QUEUE\_107\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.982 QUEUE\_107\_STATUS\_C Register (offset = 36B8h) [reset = 0h]**

QUEUE\_107\_STATUS\_C is shown in [Figure 16-1258](#) and described in [Table 16-1272](#).

**Figure 16-1258. QUEUE\_107\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1272. QUEUE\_107\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.983 QUEUE\_108\_STATUS\_A Register (offset = 36C0h) [reset = 0h]**

QUEUE\_108\_STATUS\_A is shown in [Figure 16-1259](#) and described in [Table 16-1273](#).

**Figure 16-1259. QUEUE\_108\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1273. QUEUE\_108\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.984 QUEUE\_108\_STATUS\_B Register (offset = 36C4h) [reset = 0h]**

QUEUE\_108\_STATUS\_B is shown in [Figure 16-1260](#) and described in [Table 16-1274](#).

**Figure 16-1260. QUEUE\_108\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1274. QUEUE\_108\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.985 QUEUE\_108\_STATUS\_C Register (offset = 36C8h) [reset = 0h]**

QUEUE\_108\_STATUS\_C is shown in [Figure 16-1261](#) and described in [Table 16-1275](#).

**Figure 16-1261. QUEUE\_108\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1275. QUEUE\_108\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.986 QUEUE\_109\_STATUS\_A Register (offset = 36D0h) [reset = 0h]**

QUEUE\_109\_STATUS\_A is shown in [Figure 16-1262](#) and described in [Table 16-1276](#).

**Figure 16-1262. QUEUE\_109\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1276. QUEUE\_109\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.987 QUEUE\_109\_STATUS\_B Register (offset = 36D4h) [reset = 0h]**

QUEUE\_109\_STATUS\_B is shown in [Figure 16-1263](#) and described in [Table 16-1277](#).

**Figure 16-1263. QUEUE\_109\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1277. QUEUE\_109\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.988 QUEUE\_109\_STATUS\_C Register (offset = 36D8h) [reset = 0h]**

QUEUE\_109\_STATUS\_C is shown in [Figure 16-1264](#) and described in [Table 16-1278](#).

**Figure 16-1264. QUEUE\_109\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1278. QUEUE\_109\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.989 QUEUE\_110\_STATUS\_A Register (offset = 36E0h) [reset = 0h]**

QUEUE\_110\_STATUS\_A is shown in [Figure 16-1265](#) and described in [Table 16-1279](#).

**Figure 16-1265. QUEUE\_110\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1279. QUEUE\_110\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.990 QUEUE\_110\_STATUS\_B Register (offset = 36E4h) [reset = 0h]**

QUEUE\_110\_STATUS\_B is shown in [Figure 16-1266](#) and described in [Table 16-1280](#).

**Figure 16-1266. QUEUE\_110\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1280. QUEUE\_110\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.991 QUEUE\_110\_STATUS\_C Register (offset = 36E8h) [reset = 0h]**

QUEUE\_110\_STATUS\_C is shown in [Figure 16-1267](#) and described in [Table 16-1281](#).

**Figure 16-1267. QUEUE\_110\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1281. QUEUE\_110\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.992 QUEUE\_111\_STATUS\_A Register (offset = 36F0h) [reset = 0h]**

QUEUE\_111\_STATUS\_A is shown in [Figure 16-1268](#) and described in [Table 16-1282](#).

**Figure 16-1268. QUEUE\_111\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1282. QUEUE\_111\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.993 QUEUE\_111\_STATUS\_B Register (offset = 36F4h) [reset = 0h]**

QUEUE\_111\_STATUS\_B is shown in [Figure 16-1269](#) and described in [Table 16-1283](#).

**Figure 16-1269. QUEUE\_111\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1283. QUEUE\_111\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.994 QUEUE\_111\_STATUS\_C Register (offset = 36F8h) [reset = 0h]**

QUEUE\_111\_STATUS\_C is shown in [Figure 16-1270](#) and described in [Table 16-1284](#).

**Figure 16-1270. QUEUE\_111\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1284. QUEUE\_111\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.995 QUEUE\_112\_STATUS\_A Register (offset = 3700h) [reset = 0h]**

QUEUE\_112\_STATUS\_A is shown in [Figure 16-1271](#) and described in [Table 16-1285](#).

**Figure 16-1271. QUEUE\_112\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1285. QUEUE\_112\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.996 QUEUE\_112\_STATUS\_B Register (offset = 3704h) [reset = 0h]**

QUEUE\_112\_STATUS\_B is shown in [Figure 16-1272](#) and described in [Table 16-1286](#).

**Figure 16-1272. QUEUE\_112\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1286. QUEUE\_112\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.997 QUEUE\_112\_STATUS\_C Register (offset = 3708h) [reset = 0h]**

QUEUE\_112\_STATUS\_C is shown in [Figure 16-1273](#) and described in [Table 16-1287](#).

**Figure 16-1273. QUEUE\_112\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1287. QUEUE\_112\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.998 QUEUE\_113\_STATUS\_A Register (offset = 3710h) [reset = 0h]**

QUEUE\_113\_STATUS\_A is shown in [Figure 16-1274](#) and described in [Table 16-1288](#).

**Figure 16-1274. QUEUE\_113\_STATUS\_A Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |     |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-----|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15                | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   | R-0 |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1288. QUEUE\_113\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.999 QUEUE\_113\_STATUS\_B Register (offset = 3714h) [reset = 0h]**

QUEUE\_113\_STATUS\_B is shown in [Figure 16-1275](#) and described in [Table 16-1289](#).

**Figure 16-1275. QUEUE\_113\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1289. QUEUE\_113\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1000 QUEUE\_113\_STATUS\_C Register (offset = 3718h) [reset = 0h]**

QUEUE\_113\_STATUS\_C is shown in [Figure 16-1276](#) and described in [Table 16-1290](#).

**Figure 16-1276. QUEUE\_113\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1290. QUEUE\_113\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1001 QUEUE\_114\_STATUS\_A Register (offset = 3720h) [reset = 0h]**

QUEUE\_114\_STATUS\_A is shown in Figure 16-1277 and described in Table 16-1291.

**Figure 16-1277. QUEUE\_114\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1291. QUEUE\_114\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1002 QUEUE\_114\_STATUS\_B Register (offset = 3724h) [reset = 0h]**

QUEUE\_114\_STATUS\_B is shown in Figure 16-1278 and described in Table 16-1292.

**Figure 16-1278. QUEUE\_114\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1292. QUEUE\_114\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1003 QUEUE\_114\_STATUS\_C Register (offset = 3728h) [reset = 0h]**

QUEUE\_114\_STATUS\_C is shown in Figure 16-1279 and described in Table 16-1293.

**Figure 16-1279. QUEUE\_114\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1293. QUEUE\_114\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1004 QUEUE\_115\_STATUS\_A Register (offset = 3730h) [reset = 0h]**

QUEUE\_115\_STATUS\_A is shown in Figure 16-1280 and described in Table 16-1294.

**Figure 16-1280. QUEUE\_115\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1294. QUEUE\_115\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1005 QUEUE\_115\_STATUS\_B Register (offset = 3734h) [reset = 0h]**

QUEUE\_115\_STATUS\_B is shown in Figure 16-1281 and described in Table 16-1295.

**Figure 16-1281. QUEUE\_115\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1295. QUEUE\_115\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1006 QUEUE\_115\_STATUS\_C Register (offset = 3738h) [reset = 0h]**

QUEUE\_115\_STATUS\_C is shown in [Figure 16-1282](#) and described in [Table 16-1296](#).

**Figure 16-1282. QUEUE\_115\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1296. QUEUE\_115\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1007 QUEUE\_116\_STATUS\_A Register (offset = 3740h) [reset = 0h]**

QUEUE\_116\_STATUS\_A is shown in Figure 16-1283 and described in Table 16-1297.

**Figure 16-1283. QUEUE\_116\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1297. QUEUE\_116\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1008 QUEUE\_116\_STATUS\_B Register (offset = 3744h) [reset = 0h]**

QUEUE\_116\_STATUS\_B is shown in Figure 16-1284 and described in Table 16-1298.

**Figure 16-1284. QUEUE\_116\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1298. QUEUE\_116\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1009 QUEUE\_116\_STATUS\_C Register (offset = 3748h) [reset = 0h]**

QUEUE\_116\_STATUS\_C is shown in Figure 16-1285 and described in Table 16-1299.

**Figure 16-1285. QUEUE\_116\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1299. QUEUE\_116\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1010 QUEUE\_117\_STATUS\_A Register (offset = 3750h) [reset = 0h]**

QUEUE\_117\_STATUS\_A is shown in Figure 16-1286 and described in Table 16-1300.

**Figure 16-1286. QUEUE\_117\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1300. QUEUE\_117\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1011 QUEUE\_117\_STATUS\_B Register (offset = 3754h) [reset = 0h]**

QUEUE\_117\_STATUS\_B is shown in Figure 16-1287 and described in Table 16-1301.

**Figure 16-1287. QUEUE\_117\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1301. QUEUE\_117\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1012 QUEUE\_117\_STATUS\_C Register (offset = 3758h) [reset = 0h]**

QUEUE\_117\_STATUS\_C is shown in Figure 16-1288 and described in Table 16-1302.

**Figure 16-1288. QUEUE\_117\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1302. QUEUE\_117\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1013 QUEUE\_118\_STATUS\_A Register (offset = 3760h) [reset = 0h]**

QUEUE\_118\_STATUS\_A is shown in Figure 16-1289 and described in Table 16-1303.

**Figure 16-1289. QUEUE\_118\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1303. QUEUE\_118\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1014 QUEUE\_118\_STATUS\_B Register (offset = 3764h) [reset = 0h]**

QUEUE\_118\_STATUS\_B is shown in Figure 16-1290 and described in Table 16-1304.

**Figure 16-1290. QUEUE\_118\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1304. QUEUE\_118\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1015 QUEUE\_118\_STATUS\_C Register (offset = 3768h) [reset = 0h]**

QUEUE\_118\_STATUS\_C is shown in Figure 16-1291 and described in Table 16-1305.

**Figure 16-1291. QUEUE\_118\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1305. QUEUE\_118\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1016 QUEUE\_119\_STATUS\_A Register (offset = 3770h) [reset = 0h]**

QUEUE\_119\_STATUS\_A is shown in Figure 16-1292 and described in Table 16-1306.

**Figure 16-1292. QUEUE\_119\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1306. QUEUE\_119\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1017 QUEUE\_119\_STATUS\_B Register (offset = 3774h) [reset = 0h]**

QUEUE\_119\_STATUS\_B is shown in Figure 16-1293 and described in Table 16-1307.

**Figure 16-1293. QUEUE\_119\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1307. QUEUE\_119\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1018 QUEUE\_119\_STATUS\_C Register (offset = 3778h) [reset = 0h]**

QUEUE\_119\_STATUS\_C is shown in [Figure 16-1294](#) and described in [Table 16-1308](#).

**Figure 16-1294. QUEUE\_119\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1308. QUEUE\_119\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1019 QUEUE\_120\_STATUS\_A Register (offset = 3780h) [reset = 0h]**

QUEUE\_120\_STATUS\_A is shown in Figure 16-1295 and described in Table 16-1309.

**Figure 16-1295. QUEUE\_120\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1309. QUEUE\_120\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1020 QUEUE\_120\_STATUS\_B Register (offset = 3784h) [reset = 0h]**

QUEUE\_120\_STATUS\_B is shown in [Figure 16-1296](#) and described in [Table 16-1310](#).

**Figure 16-1296. QUEUE\_120\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1310. QUEUE\_120\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1021 QUEUE\_120\_STATUS\_C Register (offset = 3788h) [reset = 0h]**

QUEUE\_120\_STATUS\_C is shown in Figure 16-1297 and described in Table 16-1311.

**Figure 16-1297. QUEUE\_120\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1311. QUEUE\_120\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1022 QUEUE\_121\_STATUS\_A Register (offset = 3790h) [reset = 0h]**

QUEUE\_121\_STATUS\_A is shown in Figure 16-1298 and described in Table 16-1312.

**Figure 16-1298. QUEUE\_121\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1312. QUEUE\_121\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1023 QUEUE\_121\_STATUS\_B Register (offset = 3794h) [reset = 0h]**

QUEUE\_121\_STATUS\_B is shown in Figure 16-1299 and described in Table 16-1313.

**Figure 16-1299. QUEUE\_121\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1313. QUEUE\_121\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1024 QUEUE\_121\_STATUS\_C Register (offset = 3798h) [reset = 0h]**

QUEUE\_121\_STATUS\_C is shown in [Figure 16-1300](#) and described in [Table 16-1314](#).

**Figure 16-1300. QUEUE\_121\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1314. QUEUE\_121\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1025 QUEUE\_122\_STATUS\_A Register (offset = 37A0h) [reset = 0h]**

QUEUE\_122\_STATUS\_A is shown in Figure 16-1301 and described in Table 16-1315.

**Figure 16-1301. QUEUE\_122\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1315. QUEUE\_122\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1026 QUEUE\_122\_STATUS\_B Register (offset = 37A4h) [reset = 0h]**

QUEUE\_122\_STATUS\_B is shown in Figure 16-1302 and described in Table 16-1316.

**Figure 16-1302. QUEUE\_122\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1316. QUEUE\_122\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1027 QUEUE\_122\_STATUS\_C Register (offset = 37A8h) [reset = 0h]**

QUEUE\_122\_STATUS\_C is shown in Figure 16-1303 and described in Table 16-1317.

**Figure 16-1303. QUEUE\_122\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1317. QUEUE\_122\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1028 QUEUE\_123\_STATUS\_A Register (offset = 37B0h) [reset = 0h]**

QUEUE\_123\_STATUS\_A is shown in [Figure 16-1304](#) and described in [Table 16-1318](#).

**Figure 16-1304. QUEUE\_123\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1318. QUEUE\_123\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1029 QUEUE\_123\_STATUS\_B Register (offset = 37B4h) [reset = 0h]**

QUEUE\_123\_STATUS\_B is shown in Figure 16-1305 and described in Table 16-1319.

**Figure 16-1305. QUEUE\_123\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1319. QUEUE\_123\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1030 QUEUE\_123\_STATUS\_C Register (offset = 37B8h) [reset = 0h]**

QUEUE\_123\_STATUS\_C is shown in [Figure 16-1306](#) and described in [Table 16-1320](#).

**Figure 16-1306. QUEUE\_123\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15          | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1320. QUEUE\_123\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1031 QUEUE\_124\_STATUS\_A Register (offset = 37C0h) [reset = 0h]**

QUEUE\_124\_STATUS\_A is shown in Figure 16-1307 and described in Table 16-1321.

**Figure 16-1307. QUEUE\_124\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1321. QUEUE\_124\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1032 QUEUE\_124\_STATUS\_B Register (offset = 37C4h) [reset = 0h]**

QUEUE\_124\_STATUS\_B is shown in Figure 16-1308 and described in Table 16-1322.

**Figure 16-1308. QUEUE\_124\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1322. QUEUE\_124\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1033 QUEUE\_124\_STATUS\_C Register (offset = 37C8h) [reset = 0h]**

QUEUE\_124\_STATUS\_C is shown in Figure 16-1309 and described in Table 16-1323.

**Figure 16-1309. QUEUE\_124\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1323. QUEUE\_124\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1034 QUEUE\_125\_STATUS\_A Register (offset = 37D0h) [reset = 0h]**

QUEUE\_125\_STATUS\_A is shown in Figure 16-1310 and described in Table 16-1324.

**Figure 16-1310. QUEUE\_125\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1324. QUEUE\_125\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1035 QUEUE\_125\_STATUS\_B Register (offset = 37D4h) [reset = 0h]**

QUEUE\_125\_STATUS\_B is shown in Figure 16-1311 and described in Table 16-1325.

**Figure 16-1311. QUEUE\_125\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1325. QUEUE\_125\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1036 QUEUE\_125\_STATUS\_C Register (offset = 37D8h) [reset = 0h]**

QUEUE\_125\_STATUS\_C is shown in [Figure 16-1312](#) and described in [Table 16-1326](#).

**Figure 16-1312. QUEUE\_125\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1326. QUEUE\_125\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1037 QUEUE\_126\_STATUS\_A Register (offset = 37E0h) [reset = 0h]**

QUEUE\_126\_STATUS\_A is shown in Figure 16-1313 and described in Table 16-1327.

**Figure 16-1313. QUEUE\_126\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1327. QUEUE\_126\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1038 QUEUE\_126\_STATUS\_B Register (offset = 37E4h) [reset = 0h]**

QUEUE\_126\_STATUS\_B is shown in [Figure 16-1314](#) and described in [Table 16-1328](#).

**Figure 16-1314. QUEUE\_126\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1328. QUEUE\_126\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1039 QUEUE\_126\_STATUS\_C Register (offset = 37E8h) [reset = 0h]**

QUEUE\_126\_STATUS\_C is shown in [Figure 16-1315](#) and described in [Table 16-1329](#).

**Figure 16-1315. QUEUE\_126\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1329. QUEUE\_126\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1040 QUEUE\_127\_STATUS\_A Register (offset = 37F0h) [reset = 0h]**

QUEUE\_127\_STATUS\_A is shown in Figure 16-1316 and described in Table 16-1330.

**Figure 16-1316. QUEUE\_127\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1330. QUEUE\_127\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1041 QUEUE\_127\_STATUS\_B Register (offset = 37F4h) [reset = 0h]**

QUEUE\_127\_STATUS\_B is shown in Figure 16-1317 and described in Table 16-1331.

**Figure 16-1317. QUEUE\_127\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1331. QUEUE\_127\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1042 QUEUE\_127\_STATUS\_C Register (offset = 37F8h) [reset = 0h]**

QUEUE\_127\_STATUS\_C is shown in [Figure 16-1318](#) and described in [Table 16-1332](#).

**Figure 16-1318. QUEUE\_127\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1332. QUEUE\_127\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1043 QUEUE\_128\_STATUS\_A Register (offset = 3800h) [reset = 0h]**

QUEUE\_128\_STATUS\_A is shown in Figure 16-1319 and described in Table 16-1333.

**Figure 16-1319. QUEUE\_128\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |                   |    |    |    |    |    |    |   |   |   |   |   |   |   |   |     |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1333. QUEUE\_128\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1044 QUEUE\_128\_STATUS\_B Register (offset = 3804h) [reset = 0h]**

QUEUE\_128\_STATUS\_B is shown in Figure 16-1320 and described in Table 16-1334.

**Figure 16-1320. QUEUE\_128\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1334. QUEUE\_128\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1045 QUEUE\_128\_STATUS\_C Register (offset = 3808h) [reset = 0h]**

QUEUE\_128\_STATUS\_C is shown in Figure 16-1321 and described in Table 16-1335.

**Figure 16-1321. QUEUE\_128\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1335. QUEUE\_128\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1046 QUEUE\_129\_STATUS\_A Register (offset = 3810h) [reset = 0h]**

QUEUE\_129\_STATUS\_A is shown in Figure 16-1322 and described in Table 16-1336.

**Figure 16-1322. QUEUE\_129\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1336. QUEUE\_129\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1047 QUEUE\_129\_STATUS\_B Register (offset = 3814h) [reset = 0h]**

QUEUE\_129\_STATUS\_B is shown in Figure 16-1323 and described in Table 16-1337.

**Figure 16-1323. QUEUE\_129\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1337. QUEUE\_129\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1048 QUEUE\_129\_STATUS\_C Register (offset = 3818h) [reset = 0h]**

QUEUE\_129\_STATUS\_C is shown in [Figure 16-1324](#) and described in [Table 16-1338](#).

**Figure 16-1324. QUEUE\_129\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1338. QUEUE\_129\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1049 QUEUE\_130\_STATUS\_A Register (offset = 3820h) [reset = 0h]**

QUEUE\_130\_STATUS\_A is shown in Figure 16-1325 and described in Table 16-1339.

**Figure 16-1325. QUEUE\_130\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1339. QUEUE\_130\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1050 QUEUE\_130\_STATUS\_B Register (offset = 3824h) [reset = 0h]**

QUEUE\_130\_STATUS\_B is shown in Figure 16-1326 and described in Table 16-1340.

**Figure 16-1326. QUEUE\_130\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1340. QUEUE\_130\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1051 QUEUE\_130\_STATUS\_C Register (offset = 3828h) [reset = 0h]**

QUEUE\_130\_STATUS\_C is shown in Figure 16-1327 and described in Table 16-1341.

**Figure 16-1327. QUEUE\_130\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    | R-0         |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1341. QUEUE\_130\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1052 QUEUE\_131\_STATUS\_A Register (offset = 3830h) [reset = 0h]**

QUEUE\_131\_STATUS\_A is shown in Figure 16-1328 and described in Table 16-1342.

**Figure 16-1328. QUEUE\_131\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1342. QUEUE\_131\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1053 QUEUE\_131\_STATUS\_B Register (offset = 3834h) [reset = 0h]**

QUEUE\_131\_STATUS\_B is shown in Figure 16-1329 and described in Table 16-1343.

**Figure 16-1329. QUEUE\_131\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1343. QUEUE\_131\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1054 QUEUE\_131\_STATUS\_C Register (offset = 3838h) [reset = 0h]**

QUEUE\_131\_STATUS\_C is shown in Figure 16-1330 and described in Table 16-1344.

**Figure 16-1330. QUEUE\_131\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1344. QUEUE\_131\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1055 QUEUE\_132\_STATUS\_A Register (offset = 3840h) [reset = 0h]**

QUEUE\_132\_STATUS\_A is shown in Figure 16-1331 and described in Table 16-1345.

**Figure 16-1331. QUEUE\_132\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1345. QUEUE\_132\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1056 QUEUE\_132\_STATUS\_B Register (offset = 3844h) [reset = 0h]**

QUEUE\_132\_STATUS\_B is shown in Figure 16-1332 and described in Table 16-1346.

**Figure 16-1332. QUEUE\_132\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1346. QUEUE\_132\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1057 QUEUE\_132\_STATUS\_C Register (offset = 3848h) [reset = 0h]**

QUEUE\_132\_STATUS\_C is shown in Figure 16-1333 and described in Table 16-1347.

**Figure 16-1333. QUEUE\_132\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1347. QUEUE\_132\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1058 QUEUE\_133\_STATUS\_A Register (offset = 3850h) [reset = 0h]**

QUEUE\_133\_STATUS\_A is shown in Figure 16-1334 and described in Table 16-1348.

**Figure 16-1334. QUEUE\_133\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1348. QUEUE\_133\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1059 QUEUE\_133\_STATUS\_B Register (offset = 3854h) [reset = 0h]**

QUEUE\_133\_STATUS\_B is shown in Figure 16-1335 and described in Table 16-1349.

**Figure 16-1335. QUEUE\_133\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1349. QUEUE\_133\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1060 QUEUE\_133\_STATUS\_C Register (offset = 3858h) [reset = 0h]**

QUEUE\_133\_STATUS\_C is shown in [Figure 16-1336](#) and described in [Table 16-1350](#).

**Figure 16-1336. QUEUE\_133\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1350. QUEUE\_133\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1061 QUEUE\_134\_STATUS\_A Register (offset = 3860h) [reset = 0h]**

QUEUE\_134\_STATUS\_A is shown in Figure 16-1337 and described in Table 16-1351.

**Figure 16-1337. QUEUE\_134\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1351. QUEUE\_134\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1062 QUEUE\_134\_STATUS\_B Register (offset = 3864h) [reset = 0h]**

QUEUE\_134\_STATUS\_B is shown in Figure 16-1338 and described in Table 16-1352.

**Figure 16-1338. QUEUE\_134\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1352. QUEUE\_134\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1063 QUEUE\_134\_STATUS\_C Register (offset = 3868h) [reset = 0h]**

QUEUE\_134\_STATUS\_C is shown in Figure 16-1339 and described in Table 16-1353.

**Figure 16-1339. QUEUE\_134\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1353. QUEUE\_134\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1064 QUEUE\_135\_STATUS\_A Register (offset = 3870h) [reset = 0h]**

QUEUE\_135\_STATUS\_A is shown in Figure 16-1340 and described in Table 16-1354.

**Figure 16-1340. QUEUE\_135\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1354. QUEUE\_135\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1065 QUEUE\_135\_STATUS\_B Register (offset = 3874h) [reset = 0h]**

QUEUE\_135\_STATUS\_B is shown in Figure 16-1341 and described in Table 16-1355.

**Figure 16-1341. QUEUE\_135\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1355. QUEUE\_135\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1066 QUEUE\_135\_STATUS\_C Register (offset = 3878h) [reset = 0h]**

QUEUE\_135\_STATUS\_C is shown in Figure 16-1342 and described in Table 16-1356.

**Figure 16-1342. QUEUE\_135\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1356. QUEUE\_135\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1067 QUEUE\_136\_STATUS\_A Register (offset = 3880h) [reset = 0h]**

QUEUE\_136\_STATUS\_A is shown in Figure 16-1343 and described in Table 16-1357.

**Figure 16-1343. QUEUE\_136\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1357. QUEUE\_136\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1068 QUEUE\_136\_STATUS\_B Register (offset = 3884h) [reset = 0h]**

QUEUE\_136\_STATUS\_B is shown in Figure 16-1344 and described in Table 16-1358.

**Figure 16-1344. QUEUE\_136\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1358. QUEUE\_136\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1069 QUEUE\_136\_STATUS\_C Register (offset = 3888h) [reset = 0h]**

QUEUE\_136\_STATUS\_C is shown in Figure 16-1345 and described in Table 16-1359.

**Figure 16-1345. QUEUE\_136\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1359. QUEUE\_136\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1070 QUEUE\_137\_STATUS\_A Register (offset = 3890h) [reset = 0h]**

QUEUE\_137\_STATUS\_A is shown in Figure 16-1346 and described in Table 16-1360.

**Figure 16-1346. QUEUE\_137\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1360. QUEUE\_137\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1071 QUEUE\_137\_STATUS\_B Register (offset = 3894h) [reset = 0h]**

QUEUE\_137\_STATUS\_B is shown in Figure 16-1347 and described in Table 16-1361.

**Figure 16-1347. QUEUE\_137\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1361. QUEUE\_137\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1072 QUEUE\_137\_STATUS\_C Register (offset = 3898h) [reset = 0h]**

QUEUE\_137\_STATUS\_C is shown in Figure 16-1348 and described in Table 16-1362.

**Figure 16-1348. QUEUE\_137\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1362. QUEUE\_137\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1073 QUEUE\_138\_STATUS\_A Register (offset = 38A0h) [reset = 0h]**

QUEUE\_138\_STATUS\_A is shown in Figure 16-1349 and described in Table 16-1363.

**Figure 16-1349. QUEUE\_138\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1363. QUEUE\_138\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1074 QUEUE\_138\_STATUS\_B Register (offset = 38A4h) [reset = 0h]**

QUEUE\_138\_STATUS\_B is shown in Figure 16-1350 and described in Table 16-1364.

**Figure 16-1350. QUEUE\_138\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1364. QUEUE\_138\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1075 QUEUE\_138\_STATUS\_C Register (offset = 38A8h) [reset = 0h]**

QUEUE\_138\_STATUS\_C is shown in Figure 16-1351 and described in Table 16-1365.

**Figure 16-1351. QUEUE\_138\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1365. QUEUE\_138\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1076 QUEUE\_139\_STATUS\_A Register (offset = 38B0h) [reset = 0h]**

QUEUE\_139\_STATUS\_A is shown in Figure 16-1352 and described in Table 16-1366.

**Figure 16-1352. QUEUE\_139\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1366. QUEUE\_139\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1077 QUEUE\_139\_STATUS\_B Register (offset = 38B4h) [reset = 0h]**

QUEUE\_139\_STATUS\_B is shown in Figure 16-1353 and described in Table 16-1367.

**Figure 16-1353. QUEUE\_139\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1367. QUEUE\_139\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1078 QUEUE\_139\_STATUS\_C Register (offset = 38B8h) [reset = 0h]**

QUEUE\_139\_STATUS\_C is shown in [Figure 16-1354](#) and described in [Table 16-1368](#).

**Figure 16-1354. QUEUE\_139\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1368. QUEUE\_139\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1079 QUEUE\_140\_STATUS\_A Register (offset = 38C0h) [reset = 0h]**

QUEUE\_140\_STATUS\_A is shown in Figure 16-1355 and described in Table 16-1369.

**Figure 16-1355. QUEUE\_140\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1369. QUEUE\_140\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1080 QUEUE\_140\_STATUS\_B Register (offset = 38C4h) [reset = 0h]**

QUEUE\_140\_STATUS\_B is shown in Figure 16-1356 and described in Table 16-1370.

**Figure 16-1356. QUEUE\_140\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1370. QUEUE\_140\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1081 QUEUE\_140\_STATUS\_C Register (offset = 38C8h) [reset = 0h]**

QUEUE\_140\_STATUS\_C is shown in Figure 16-1357 and described in Table 16-1371.

**Figure 16-1357. QUEUE\_140\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1371. QUEUE\_140\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1082 QUEUE\_141\_STATUS\_A Register (offset = 38D0h) [reset = 0h]**

QUEUE\_141\_STATUS\_A is shown in Figure 16-1358 and described in Table 16-1372.

**Figure 16-1358. QUEUE\_141\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1372. QUEUE\_141\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1083 QUEUE\_141\_STATUS\_B Register (offset = 38D4h) [reset = 0h]**

QUEUE\_141\_STATUS\_B is shown in Figure 16-1359 and described in Table 16-1373.

**Figure 16-1359. QUEUE\_141\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1373. QUEUE\_141\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1084 QUEUE\_141\_STATUS\_C Register (offset = 38D8h) [reset = 0h]**

QUEUE\_141\_STATUS\_C is shown in [Figure 16-1360](#) and described in [Table 16-1374](#).

**Figure 16-1360. QUEUE\_141\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1374. QUEUE\_141\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1085 QUEUE\_142\_STATUS\_A Register (offset = 38E0h) [reset = 0h]**

QUEUE\_142\_STATUS\_A is shown in Figure 16-1361 and described in Table 16-1375.

**Figure 16-1361. QUEUE\_142\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1375. QUEUE\_142\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1086 QUEUE\_142\_STATUS\_B Register (offset = 38E4h) [reset = 0h]**

QUEUE\_142\_STATUS\_B is shown in Figure 16-1362 and described in Table 16-1376.

**Figure 16-1362. QUEUE\_142\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1376. QUEUE\_142\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1087 QUEUE\_142\_STATUS\_C Register (offset = 38E8h) [reset = 0h]**

QUEUE\_142\_STATUS\_C is shown in Figure 16-1363 and described in Table 16-1377.

**Figure 16-1363. QUEUE\_142\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1377. QUEUE\_142\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1088 QUEUE\_143\_STATUS\_A Register (offset = 38F0h) [reset = 0h]**

QUEUE\_143\_STATUS\_A is shown in Figure 16-1364 and described in Table 16-1378.

**Figure 16-1364. QUEUE\_143\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1378. QUEUE\_143\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1089 QUEUE\_143\_STATUS\_B Register (offset = 38F4h) [reset = 0h]**

QUEUE\_143\_STATUS\_B is shown in Figure 16-1365 and described in Table 16-1379.

**Figure 16-1365. QUEUE\_143\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1379. QUEUE\_143\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1090 QUEUE\_143\_STATUS\_C Register (offset = 38F8h) [reset = 0h]**

QUEUE\_143\_STATUS\_C is shown in [Figure 16-1366](#) and described in [Table 16-1380](#).

**Figure 16-1366. QUEUE\_143\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1380. QUEUE\_143\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1091 QUEUE\_144\_STATUS\_A Register (offset = 3900h) [reset = 0h]**

QUEUE\_144\_STATUS\_A is shown in Figure 16-1367 and described in Table 16-1381.

**Figure 16-1367. QUEUE\_144\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1381. QUEUE\_144\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1092 QUEUE\_144\_STATUS\_B Register (offset = 3904h) [reset = 0h]**

QUEUE\_144\_STATUS\_B is shown in Figure 16-1368 and described in Table 16-1382.

**Figure 16-1368. QUEUE\_144\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1382. QUEUE\_144\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1093 QUEUE\_144\_STATUS\_C Register (offset = 3908h) [reset = 0h]**

QUEUE\_144\_STATUS\_C is shown in Figure 16-1369 and described in Table 16-1383.

**Figure 16-1369. QUEUE\_144\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1383. QUEUE\_144\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1094 QUEUE\_145\_STATUS\_A Register (offset = 3910h) [reset = 0h]**

QUEUE\_145\_STATUS\_A is shown in Figure 16-1370 and described in Table 16-1384.

**Figure 16-1370. QUEUE\_145\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1384. QUEUE\_145\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1095 QUEUE\_145\_STATUS\_B Register (offset = 3914h) [reset = 0h]**

QUEUE\_145\_STATUS\_B is shown in Figure 16-1371 and described in Table 16-1385.

**Figure 16-1371. QUEUE\_145\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1385. QUEUE\_145\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1096 QUEUE\_145\_STATUS\_C Register (offset = 3918h) [reset = 0h]**

QUEUE\_145\_STATUS\_C is shown in Figure 16-1372 and described in Table 16-1386.

**Figure 16-1372. QUEUE\_145\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1386. QUEUE\_145\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1097 QUEUE\_146\_STATUS\_A Register (offset = 3920h) [reset = 0h]**

QUEUE\_146\_STATUS\_A is shown in Figure 16-1373 and described in Table 16-1387.

**Figure 16-1373. QUEUE\_146\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1387. QUEUE\_146\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1098 QUEUE\_146\_STATUS\_B Register (offset = 3924h) [reset = 0h]**

QUEUE\_146\_STATUS\_B is shown in Figure 16-1374 and described in Table 16-1388.

**Figure 16-1374. QUEUE\_146\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1388. QUEUE\_146\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1099 QUEUE\_146\_STATUS\_C Register (offset = 3928h) [reset = 0h]**

QUEUE\_146\_STATUS\_C is shown in Figure 16-1375 and described in Table 16-1389.

**Figure 16-1375. QUEUE\_146\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1389. QUEUE\_146\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1100 QUEUE\_147\_STATUS\_A Register (offset = 3930h) [reset = 0h]**

QUEUE\_147\_STATUS\_A is shown in Figure 16-1376 and described in Table 16-1390.

**Figure 16-1376. QUEUE\_147\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1390. QUEUE\_147\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1101 QUEUE\_147\_STATUS\_B Register (offset = 3934h) [reset = 0h]**

QUEUE\_147\_STATUS\_B is shown in Figure 16-1377 and described in Table 16-1391.

**Figure 16-1377. QUEUE\_147\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1391. QUEUE\_147\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1102 QUEUE\_147\_STATUS\_C Register (offset = 3938h) [reset = 0h]**

QUEUE\_147\_STATUS\_C is shown in Figure 16-1378 and described in Table 16-1392.

**Figure 16-1378. QUEUE\_147\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17          | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3   | 2 | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|----|---|---|---|---|---|---|-----|---|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |    |   |   |   |   |   |   | R-0 |   |   |   |
|          |    |    |    |    |    |    |    |    |    |    |    |    |    |             |    |    |    |    |    |    |    |   |   |   |   |   |   |     |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1392. QUEUE\_147\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1103 QUEUE\_148\_STATUS\_A Register (offset = 3940h) [reset = 0h]**

QUEUE\_148\_STATUS\_A is shown in Figure 16-1379 and described in Table 16-1393.

**Figure 16-1379. QUEUE\_148\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1393. QUEUE\_148\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1104 QUEUE\_148\_STATUS\_B Register (offset = 3944h) [reset = 0h]**

QUEUE\_148\_STATUS\_B is shown in Figure 16-1380 and described in Table 16-1394.

**Figure 16-1380. QUEUE\_148\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1394. QUEUE\_148\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1105 QUEUE\_148\_STATUS\_C Register (offset = 3948h) [reset = 0h]**

QUEUE\_148\_STATUS\_C is shown in Figure 16-1381 and described in Table 16-1395.

**Figure 16-1381. QUEUE\_148\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1395. QUEUE\_148\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1106 QUEUE\_149\_STATUS\_A Register (offset = 3950h) [reset = 0h]**

QUEUE\_149\_STATUS\_A is shown in Figure 16-1382 and described in Table 16-1396.

**Figure 16-1382. QUEUE\_149\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1396. QUEUE\_149\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1107 QUEUE\_149\_STATUS\_B Register (offset = 3954h) [reset = 0h]**

QUEUE\_149\_STATUS\_B is shown in Figure 16-1383 and described in Table 16-1397.

**Figure 16-1383. QUEUE\_149\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1397. QUEUE\_149\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1108 QUEUE\_149\_STATUS\_C Register (offset = 3958h) [reset = 0h]**

QUEUE\_149\_STATUS\_C is shown in Figure 16-1384 and described in Table 16-1398.

**Figure 16-1384. QUEUE\_149\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1398. QUEUE\_149\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1109 QUEUE\_150\_STATUS\_A Register (offset = 3960h) [reset = 0h]**

QUEUE\_150\_STATUS\_A is shown in Figure 16-1385 and described in Table 16-1399.

**Figure 16-1385. QUEUE\_150\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1399. QUEUE\_150\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1110 QUEUE\_150\_STATUS\_B Register (offset = 3964h) [reset = 0h]**

QUEUE\_150\_STATUS\_B is shown in Figure 16-1386 and described in Table 16-1400.

**Figure 16-1386. QUEUE\_150\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1400. QUEUE\_150\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1111 QUEUE\_150\_STATUS\_C Register (offset = 3968h) [reset = 0h]**

QUEUE\_150\_STATUS\_C is shown in Figure 16-1387 and described in Table 16-1401.

**Figure 16-1387. QUEUE\_150\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1401. QUEUE\_150\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1112 QUEUE\_151\_STATUS\_A Register (offset = 3970h) [reset = 0h]**

QUEUE\_151\_STATUS\_A is shown in Figure 16-1388 and described in Table 16-1402.

**Figure 16-1388. QUEUE\_151\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1402. QUEUE\_151\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1113 QUEUE\_151\_STATUS\_B Register (offset = 3974h) [reset = 0h]**

QUEUE\_151\_STATUS\_B is shown in Figure 16-1389 and described in Table 16-1403.

**Figure 16-1389. QUEUE\_151\_STATUS\_B Register**

|          |    |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29               | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved |    | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1403. QUEUE\_151\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1114 QUEUE\_151\_STATUS\_C Register (offset = 3978h) [reset = 0h]**

QUEUE\_151\_STATUS\_C is shown in Figure 16-1390 and described in Table 16-1404.

**Figure 16-1390. QUEUE\_151\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1404. QUEUE\_151\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1115 QUEUE\_152\_STATUS\_A Register (offset = 3980h) [reset = 0h]**

QUEUE\_152\_STATUS\_A is shown in Figure 16-1391 and described in Table 16-1405.

**Figure 16-1391. QUEUE\_152\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1405. QUEUE\_152\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1116 QUEUE\_152\_STATUS\_B Register (offset = 3984h) [reset = 0h]**

QUEUE\_152\_STATUS\_B is shown in Figure 16-1392 and described in Table 16-1406.

**Figure 16-1392. QUEUE\_152\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1406. QUEUE\_152\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1117 QUEUE\_152\_STATUS\_C Register (offset = 3988h) [reset = 0h]**

QUEUE\_152\_STATUS\_C is shown in Figure 16-1393 and described in Table 16-1407.

**Figure 16-1393. QUEUE\_152\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1407. QUEUE\_152\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1118 QUEUE\_153\_STATUS\_A Register (offset = 3990h) [reset = 0h]**

QUEUE\_153\_STATUS\_A is shown in Figure 16-1394 and described in Table 16-1408.

**Figure 16-1394. QUEUE\_153\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1408. QUEUE\_153\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1119 QUEUE\_153\_STATUS\_B Register (offset = 3994h) [reset = 0h]**

QUEUE\_153\_STATUS\_B is shown in Figure 16-1395 and described in Table 16-1409.

**Figure 16-1395. QUEUE\_153\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1409. QUEUE\_153\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1120 QUEUE\_153\_STATUS\_C Register (offset = 3998h) [reset = 0h]**

QUEUE\_153\_STATUS\_C is shown in [Figure 16-1396](#) and described in [Table 16-1410](#).

**Figure 16-1396. QUEUE\_153\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1410. QUEUE\_153\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1121 QUEUE\_154\_STATUS\_A Register (offset = 39A0h) [reset = 0h]**

QUEUE\_154\_STATUS\_A is shown in Figure 16-1397 and described in Table 16-1411.

**Figure 16-1397. QUEUE\_154\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1411. QUEUE\_154\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1122 QUEUE\_154\_STATUS\_B Register (offset = 39A4h) [reset = 0h]**

QUEUE\_154\_STATUS\_B is shown in Figure 16-1398 and described in Table 16-1412.

**Figure 16-1398. QUEUE\_154\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1412. QUEUE\_154\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1123 QUEUE\_154\_STATUS\_C Register (offset = 39A8h) [reset = 0h]**

QUEUE\_154\_STATUS\_C is shown in Figure 16-1399 and described in Table 16-1413.

**Figure 16-1399. QUEUE\_154\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2   | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|-----|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   | R-0 |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1413. QUEUE\_154\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

**16.5.7.1124 QUEUE\_155\_STATUS\_A Register (offset = 39B0h) [reset = 0h]**

QUEUE\_155\_STATUS\_A is shown in Figure 16-1400 and described in Table 16-1414.

**Figure 16-1400. QUEUE\_155\_STATUS\_A Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16                | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | QUEUE_ENTRY_COUNT |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1414. QUEUE\_155\_STATUS\_A Register Field Descriptions**

| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 13-0 | QUEUE_ENTRY_COUNT | R-0  | 0     | This field indicates how many packets are currently queued on the queue.<br>Queue Manager Queue N Status Registers A |

**16.5.7.1125 QUEUE\_155\_STATUS\_B Register (offset = 39B4h) [reset = 0h]**

QUEUE\_155\_STATUS\_B is shown in Figure 16-1401 and described in Table 16-1415.

**Figure 16-1401. QUEUE\_155\_STATUS\_B Register**

| 31       | 30               | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Reserved | QUEUE_BYTE_COUNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1415. QUEUE\_155\_STATUS\_B Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description   |
|------|------------------|------|-------|---|
| 27-0 | QUEUE_BYTE_COUNT | R-0  | 0     | This field indicates how many bytes total are contained in all of the packets which are currently queued on this queue.<br>Queue_Manager_Queue_n_Status_B Registers B |

**16.5.7.1126 QUEUE\_155\_STATUS\_C Register (offset = 39B8h) [reset = 0h]**

QUEUE\_155\_STATUS\_C is shown in [Figure 16-1402](#) and described in [Table 16-1416](#).

**Figure 16-1402. QUEUE\_155\_STATUS\_C Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16          | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|-----|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    | PACKET_SIZE |    |    |    |    |    |    |   |   |   |   |   |   |   |   | R-0 |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 16-1416. QUEUE\_155\_STATUS\_C Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 13-0 | PACKET_SIZE | R-0  | 0     | This field indicates packet size of the head element of a queue.<br>Queue_Manager_Queue_N_Status_C Registers C |

## ***Interprocessor Communication***

This chapter describes the interprocessor communication of the device.

| Topic               | Page |
|---------------------|------|
| 17.1 Mailbox .....  | 3223 |
| 17.2 Spinlock ..... | 3293 |

## 17.1 Mailbox

### 17.1.1 *Introduction*

#### 17.1.1.1 Features

Global features of the Mailbox module are:

- OCP slave interface (L4) supports:
  - 32-bit data bus width
  - 8/16/32 bit access supported
  - 9-bit address bus width
  - Burst not supported
- 8 mailbox sub-modules
- Each mailbox sub module allows 1-way communication between 2 initiators
- Flexible mailbox/initiators assignment scheme
- 4 messages per mailbox sub-module
- 32-bit message width
- Support of 16/32-bit addressing scheme
- Non-intrusive emulation
- 4 interrupts (one per user: 1 to MPU Subsystem, 2 to PRU-ICSS, and 1 to WakeM3)

#### 17.1.1.2 Unsupported Features

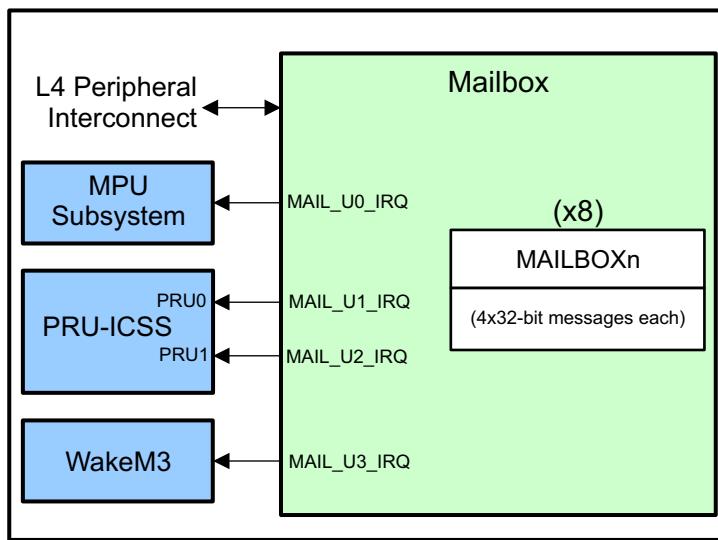
There are no unsupported features for Mailbox on this device.

## 17.1.2 Integration

This device contains a single instantiation of the Mailbox module at the system level. The mailbox function is made of eight sub-module mailboxes each supporting a 1-way communication between two initiators. The communication protocol from the sender to the receiver is implemented with mailbox registers using interrupts. The sender sends information to the receiver by writing to the mailbox. Interrupt signaling is used to notify the receiver a message has been queued or the sender for overflow situation.

The eight mailboxes are enough to handle communications between the MPU Subsystem, PRU-ICSS PRUs, and WakeM3. Note that because the WakeM3 has access only to L4\_Wakeup peripherals it does not have access to the Mailbox registers. A mailbox interrupt can still be sent to the M3 to trigger message notification. The actual message payload must be placed in either M3 internal memory or in the Control Module Interprocessor Message registers (IPC\_MSG\_REG{0-7}).

**Figure 17-1. Mailbox Integration**



### 17.1.2.1 Mailbox Connectivity Attributes

The general connectivity for the Mailbox is shown in [Table 17-1](#).

**Table 17-1. Mailbox Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | L4PER_L4LS_GCLK  |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 4 Interrupts<br>mail_u0 (MBINT0) – to MPU Subsystem<br>mail_u1 – to PRU-ICSS (PRU0)<br>mail_u2 – to PRU-ICSS (PRU1)<br>mail_u3 – to WakeM3 |
| DMA Requests        | None   |
| Physical Address    | L4 Peripheral slave port   |

### 17.1.2.2 Mailbox Clock and Reset Management

The mailbox function operates from the L4 interface clock.

**Table 17-2. Mailbox Clock Signals**

| Clock Signal                 | Max Freq | Reference / Source | Comments                      |
|------------------------------|----------|--------------------|-------------------------------|
| Functional / Interface clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>From PRCM |

### 17.1.2.3 Mailbox Pin List

The Mailbox module does not include any external interface pins.

### 17.1.3 Functional Description

This device has the following mailbox instances:

- System mailbox

**Table 17-3** shows Mailbox Implementation in this device, where u is the user number and m is the mailbox number.

**Table 17-3. Mailbox Implementation**

| Mailbox Type   | User Number(u) | Mailbox Number(m) | Messages per Mailbox |
|----------------|----------------|-------------------|----------------------|
| System mailbox | 0 to 3         | 0 to 7            | 4                    |

The mailbox module provides a means of communication through message queues among the users (depending on the mailbox module instance). The individual mailbox modules (8 for the system mailbox instance), or FIFOs, can associate (or de-associate) with any of the processors using the MAILBOX\_IRQENABLE\_SET\_u (or MAILBOX\_IRQENABLE\_CLR\_u) register.

The system mailbox module includes the following user subsystems:

- User 0: MPU Subsystem (u = 0)
- User 1: PRU\_ICSS PRU0 (u = 1)
- User 2: PRU\_ICSS PRU1 (u = 2)
- User 3: WakeM3 (u = 3)

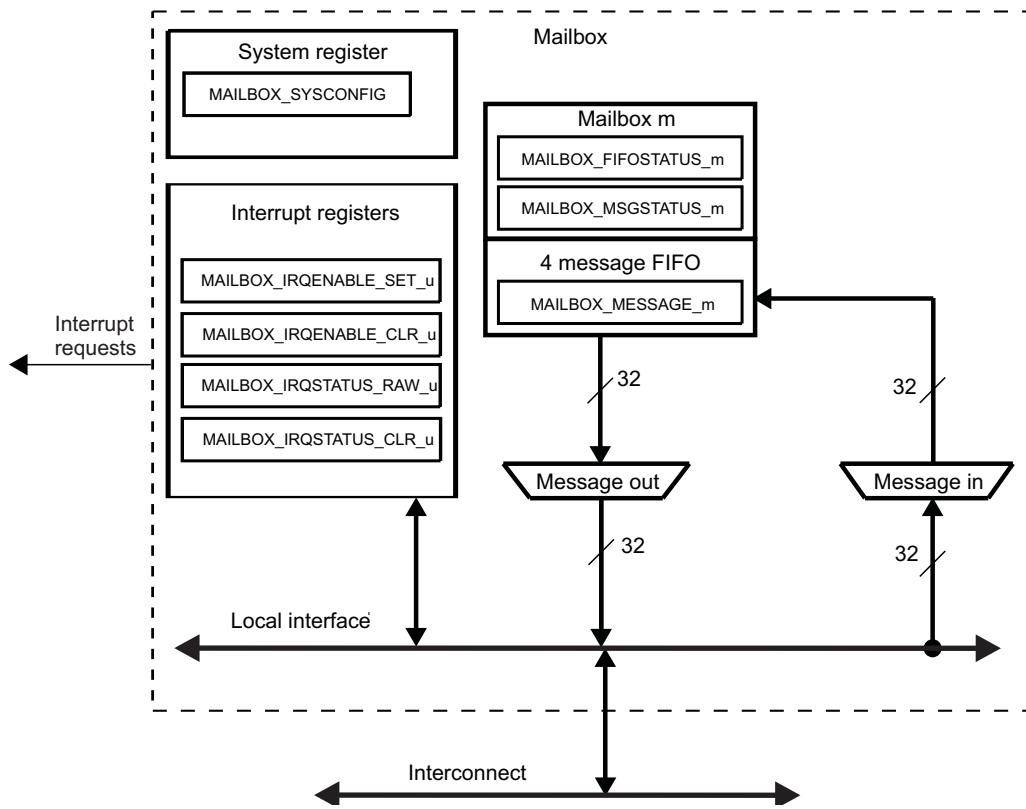
Each user has a dedicated interrupt signal from the corresponding mailbox module instance and dedicated interrupt enabling and status registers. Each MAILBOX\_IRQSTATUS\_RAW\_u/MAILBOX\_IRQSTATUS\_CLR\_u interrupt status register corresponds to a particular user.

For the system mailbox instance, a user can query its interrupt status register through the L4\_STANDARD interconnect.

#### 17.1.3.1 Mailbox Block Diagram

[Figure 17-2](#) shows the mailbox block diagram.

Figure 17-2. Mailbox Block Diagram



### 17.1.3.2 Software Reset

The mailbox module supports a software reset through the MAILBOX\_SYSCONFIG[0].SOFTRESET bit. Setting this bit to 1 enables an active software reset that is functionally equivalent to a hardware reset. Reading the MAILBOX\_SYSCONFIG[0] SOFTRESET bit gives the status of the software reset:

- Read 1: the software reset is on-going
- Read 0: the software reset is complete

The software must ensure that the software reset completes before doing mailbox operations.

### 17.1.3.3 Power Management

Table 17-4 describes power-management features available for the mailbox module.

Table 17-4. Local Power Management Features

| Feature                | Registers                        | Description  |
|------------------------|----------------------------------|--|
| Clock autogating       | NA                               | Feature not available                                  |
| Slave idle modes       | MAILBOX_SYSCONFIG[3:2].SIDLEMODE | Force-idle, no-idle and smart-idle modes are available |
| Clock activity         | NA                               | Feature not available                                  |
| Master standby modes   | NA                               | Feature not available                                  |
| Global wake-up enable  | NA                               | Feature not available                                  |
| Wake-up sources enable | NA                               | Feature not available                                  |

The mailbox module can be configured using the MAILBOX\_SYSConfig[3:2] SIDLEMODE bit field to one of the following acknowledgment modes:

- Force-idle mode (SIDLEMODE = 0x0): The mailbox module immediately enters the idle state on receiving a low-power-mode request from the PRCM module. In this mode, the software must ensure that there are no asserted output interrupts before requesting this mode to go into the idle state.
- No-idle mode (SIDLEMODE = 0x1): The mailbox module never enters the idle state.
- Smart-idle mode (SIDLEMODE = 0x2): After receiving a low-power-mode request from the PRCM module, the mailbox module enters the idle state only after all asserted output interrupts are acknowledged.

#### 17.1.3.4 Interrupt Requests

An interrupt request allows the user of the mailbox to be notified when a message is received or when the message queue is not full. There is one interrupt per user. [Table 17-5](#) lists the event flags, and their mask, that can cause module interrupts.

**Table 17-5. Interrupt Events**

| Non-Maskable Event Flag <sup>(1)</sup>            | Maskable Event Flag                               | Event Mask Bit  | Event Unmask Bit                                      | Description                         |
|---|---|---|---|-------------------------------------|
| MAILBOX_IRQSTATUS_RAW_u[0+m*2].NEWMSGSTATUSUUMBm  | MAILBOX_IRQSTATUS_CLR_u[0+m*2].NEWMSGSTATUSUUMBm  | MAILBOX_IRQENABLE_CLR_u[0+m*2].                       |   |                                     |
| MAILBOX_IRQSTATUS_RAW_u[0+m*2].NEWMSGSTATUSUUMBm  | MAILBOX_IRQSTATUS_CLR_u[0+m*2].NEWMSGSTATUSUUMBm  | MAILBOX_IRQENABLE_CLR_u[0+m*2].<br>NEWMSGSTATUSUUMBm  | MAILBOX_IRQENABLE_SET_u[0+m*2].<br>NEWMSGSTATUSUUMBm  | Mailbox m receives a new message    |
| MAILBOX_IRQSTATUS_RAW_u[1+m*2].NOTFULLSTATUSUUMBm | MAILBOX_IRQSTATUS_CLR_u[1+m*2].NOTFULLSTATUSUUMBm | MAILBOX_IRQENABLE_CLR_u[1+m*2].<br>NOTFULLSTATUSUUMBm | MAILBOX_IRQENABLE_SET_u[1+m*2].<br>NOTFULLSTATUSUUMBm | Mailbox m message queue is not full |

<sup>(1)</sup> MAILBOX.MAILBOX\_IRQSTATUS\_RAW\_u register is mostly used for debug purposes.

**CAUTION**

Once an event generating the interrupt request has been processed by the software, it must be cleared by writing a logical 1 in the corresponding bit of the MAILBOX\_IRQSTATUS\_CLR\_u register. Writing a logical 1 in a bit of the MAILBOX\_IRQSTATUS\_CLR\_u register will also clear to 0 the corresponding bit in the appropriate MAILBOX\_IRQSTATUS\_RAW\_u register.

An event can generate an interrupt request when a logical 1 is written to the corresponding unmask bit in the MAILBOX\_IRQENABLE\_SET\_u register. Events are reported in the appropriate MAILBOX\_IRQSTATUS\_CLR\_u and MAILBOX\_IRQSTATUS\_RAW\_u registers.

An event stops generating interrupt requests when a logical 1 is written to the corresponding mask bit in the MAILBOX\_IRQENABLE\_CLR\_u register. Events are only reported in the appropriate MAILBOX\_IRQSTATUS\_RAW\_u register.

In case of the MAILBOX\_IRQSTATUS\_RAW\_u register, the event is reported in the corresponding bit even if the interrupt request generation is disabled for this event.

### 17.1.3.5 Assignment

#### 17.1.3.5.1 Description

To assign a receiver to a mailbox, set the new message interrupt enable bit corresponding to the desired mailbox in the MAILBOX\_IRQENABLE\_SET\_u register. The receiver reads the MAILBOX\_MESSAGE\_m register to retrieve a message from the mailbox.

An alternate method for the receiver that does not use the interrupts is to poll the MAILBOX\_FIFOSTATUS\_m and/or MAILBOX\_MSGSTATUS\_m registers to know when to send or retrieve a message to or from the mailbox. This method does not require assigning a receiver to a mailbox. Because this method does not include the explicit assignment of the mailbox, the software must avoid having multiple receivers use the same mailbox, which can result in incoherency.

To assign a sender to a mailbox, set the queue-not-full interrupt enable bit of the desired mailbox in the MAILBOX\_IRQENABLE\_SET\_u register, where u is the number of the sending user. However, direct allocation of a mailbox to a sender is not recommended because it can cause the sending processor to be constantly interrupted.

It is recommended that register polling be used to:

- Check the status of either the MAILBOX\_FIFOSTATUS\_m or MAILBOX\_MSGSTATUS\_m registers
- Write the message to the corresponding MAILBOX\_MESSAGE\_m register, if space is available

The sender might use the queue-not-full interrupt when the initial mailbox status check indicates the mailbox is full. In this case, the sender can enable the queue-not-full interrupt for its mailbox in the appropriate MAILBOX\_IRQENABLE\_SET\_u register. This allows the sender to be notified by interrupt only when a FIFO queue has at least one available entry.

Reading the MAILBOX\_IRQSTATUS\_CLR\_u register determines the status of the new message and the queue-not-full interrupts for a particular user. Writing 1 to the corresponding bit in the MAILBOX\_IRQSTATUS\_CLR\_u register acknowledges, and subsequently clears, an interrupt.

#### CAUTION

Assigning multiple senders or multiple receivers to the same mailbox is not recommended.

### 17.1.3.6 Sending and Receiving Messages

#### 17.1.3.6.1 Description

When a 32-bit message is written to the MAILBOX\_MESSAGE\_m register, the message is appended into the FIFO queue. This queue holds four messages. If the queue is full, the message is discarded. Queue overflow can be avoided by first reading the MAILBOX\_FIFOSTATUS\_m register to check that the mailbox message queue is not full before writing a new message to it. Reading the MAILBOX\_MESSAGE\_m register returns the message at the beginning of the FIFO queue and removes it from the queue. If the FIFO queue is empty when the MAILBOX\_MESSAGE\_m register is read, the value 0 is returned. The new message interrupt is asserted when at least one message is in the mailbox message FIFO queue. To determine the number of messages in the mailbox message FIFO queue, read the MAILBOX\_MSGSTATUS\_m register.

### 17.1.3.7 16-Bit Register Access

#### 17.1.3.7.1 Description

So that 16-bit processors can access the mailbox module, the module allows 16-bit register read and write access, with restrictions for the MAILBOX\_MESSAGE\_m registers. The 16-bit half-words are organized in little endian fashion; that is, the least-significant 16 bits are at the low address and the most-significant 16 bits are at the high address (low address + 0x02). All mailbox module registers can be read or written to directly using individual 16-bit accesses with no restriction on interleaving, except the MAILBOX\_MESSAGE\_m registers, which must always be accessed by either single 32-bit accesses or two consecutive 16-bit accesses.

**CAUTION**

When using 16-bit accesses to the MAILBOX\_MESSAGE\_m registers, the order of access must be the least-significant half-word first (low address) and the most-significant half-word last (high address). This requirement is because of the update operation by the message FIFO of the MAILBOX\_MSGSTATUS\_m registers. The update of the FIFO queue contents and the associated status registers and possible interrupt generation occurs only when the most-significant 16 bits of a MAILBOX\_MESSAGE\_m are accessed.

### 17.1.4 Programming Guide

#### 17.1.4.1 Low-level Programming Models

This section covers the low-level hardware programming sequences for configuration and usage of the mailbox module.

##### 17.1.4.1.1 Global Initialization

###### 17.1.4.1.1.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the mailbox module is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration of the mailbox.

See [Section 17.1.2](#) for further information.

**Table 17-6. Global Initialization of Surrounding Modules for System Mailbox**

| Surrounding Modules   | Comments   |
|-----------------------|--|
| PRCM                  | Mailbox functional/interface clock must be enabled.  |
| Interrupt Controllers | Cortex-A8 MPU interrupt controller must be configured to enable the interrupt request generation to the Cortex-A8 MPU. |

#### 17.1.4.1.1.2 Mailbox Global Initialization

##### 17.1.4.1.1.2.1 Main Sequence - Mailbox Global Initialization

This procedure initializes the mailbox module after a power-on or software reset.

**Table 17-7. Mailbox Global Initialization**

|                              | Register/Bitfield/Programming Model | Value |
|------------------------------|-------------------------------------|-------|
| Perform a software reset     | MAILBOX_SYS CONFIG[0].SOFTRESET     | 1     |
| Wait until reset is complete | MAILBOX_SYS CONFIG[0].SOFTRESET     | 0     |
| Set idle mode configuration  | MAILBOX_SYS CONFIG[3:2].SIDLEMOD    | 0xE   |

#### 17.1.4.1.2 Operational Modes Configuration

##### 17.1.4.1.2.1 Main Sequence - Sending a Message (Polling Method)

**Table 17-8. Sending a Message (Polling Method)**

| Step  | Register/Bitfield/Programming Model     | Value |
|---|---|-------|
| IF : Is FIFO full ?                               | MAILBOX_FIFOSTATUS_m[0].FIFOFULL_MB     | =1h   |
| Wait until at least one message slot is available | MAILBOX_FIFOSTATUS_m[0].FIFOFULL_MB     | =0h   |
| ELSE  |   |       |
| Write message                                     | MAILBOX_MESSAGE_m[31:0].MESSAGEVALUEMBM | ---h  |
| ENDIF   |   |       |

##### 17.1.4.1.2.2 Main Sequence - Sending a Message (Interrupt Method)

**Table 17-9. Sending a Message (Interrupt Method)**

| Step  | Register/Bitfield/Programming Model     | Value |
|---|---|-------|
| IF : Is FIFO full ?   | MAILBOX_FIFOSTATUS_m[0].FIFOFULL_MB     | =1h   |
| Enable interrupt event  | MAILBOX_IRQENABLE_SET_u[1+ m*2]         | 1h    |
| User(processor) can perform another task until interrupt occurs |   |       |
| ELSE  |   |       |
| Write message   | MAILBOX_MESSAGE_m[31:0].MESSAGEVALUEMBM | ---h  |
| ENDIF   |   |       |

**17.1.4.1.2.3 Main Sequence - Receiving a Message (Polling Method)**
**Table 17-10. Receiving a Message (Polling Method)**

| Step                                      | Register/Bitfield/Programming Model         | Value |
|---|---|-------|
| IF : Number of messages is not equal to 0 | MAILBOX_MSGSTATUS_m[2:0].NBOFM<br>SGMB      | !=0h  |
| Read message                              | MAILBOX_MESSAGE_m[31:0].MESSAG<br>EVALUEMBM | ---h  |
| ENDIF                                     |   |       |

**17.1.4.1.2.4 Main Sequence - Receiving a Message (Interrupt Method)**
**Table 17-11. Receiving a Message (Interrupt Method)**

| Step  | Register/Bitfield/Programming Model | Value |
|---|-------------------------------------|-------|
| Enable interrupt event  | MAILBOX_IRQENABLE_SET_u[0 + m*2]    | 1h    |
| User(processor) can perform another task until interrupt occurs |                                     |       |

**17.1.4.1.3 Events Servicing**
**17.1.4.1.3.1 Sending Mode**

Table 17-12 describes the events servicing in sending mode.

**Table 17-12. Events Servicing in Sending Mode**

| Step                             | Register/Bitfield/Programming Model         | Value |
|----------------------------------|---|-------|
| Read interrupt status bit        | MAILBOX_IRQSTATUS_CLR_u[1 + m*2]            | 1     |
| Write message                    | MAILBOX_MESSAGE_m[31:0].MESSAG<br>EVALUEMBM | ---h  |
| Write 1 to acknowledge interrupt | MAILBOX_IRQSTATUS_CLR_u[1 + m*2]            | 1     |

**17.1.4.1.3.2 Receiving Mode**

Table 17-13 describes the events servicing in receiving mode.

**Table 17-13. Events Servicing in Receiving Mode**

| Step   | Register/Bitfield/Programming Model         | Value |
|--|---|-------|
| Read interrupt status bit                      | MAILBOX_IRQSTATUS_CLR_u[0 + m*2]            | 1     |
| IF : Number of messages is not equal to 0<br>? | MAILBOX_MSGSTATUS_m[2:0].NBOFM<br>SGMB      | !=0h  |
| Read message                                   | MAILBOX_MESSAGE_m[31:0].MESSAG<br>EVALUEMBM | ---h  |
| ELSE   |   |       |
| Write 1 to acknowledge interrupt               | MAILBOX_IRQSTATUS_CLR_u[0 + m*2]            | 1     |
| ENDIF  |   |       |

### 17.1.5 MAILBOX Registers

Table 17-14 lists the memory-mapped registers for the MAILBOX. All register offset addresses not listed in Table 17-14 should be considered as reserved locations and the register contents should not be modified.

**Table 17-14. MAILBOX REGISTERS**

| Offset | Acronym      | Register Name   | Section                           |
|--------|--------------|---|-----------------------------------|
| 0h     | REVISION     | This register contains the IP revision code   | <a href="#">Section 17.1.5.1</a>  |
| 10h    | SYSCONFIG    | This register controls the various parameters of the OCP interface  | <a href="#">Section 17.1.5.2</a>  |
| 40h    | MESSAGE_0    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.3</a>  |
| 44h    | MESSAGE_1    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.4</a>  |
| 48h    | MESSAGE_2    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.5</a>  |
| 4Ch    | MESSAGE_3    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.6</a>  |
| 50h    | MESSAGE_4    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.7</a>  |
| 54h    | MESSAGE_5    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.8</a>  |
| 58h    | MESSAGE_6    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.9</a>  |
| 5Ch    | MESSAGE_7    | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.10</a> |
| 80h    | FIFOSTATUS_0 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.11</a> |
| 84h    | FIFOSTATUS_1 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.12</a> |
| 88h    | FIFOSTATUS_2 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.13</a> |
| 8Ch    | FIFOSTATUS_3 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.14</a> |
| 90h    | FIFOSTATUS_4 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.15</a> |
| 94h    | FIFOSTATUS_5 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.16</a> |
| 98h    | FIFOSTATUS_6 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.17</a> |
| 9Ch    | FIFOSTATUS_7 | The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue | <a href="#">Section 17.1.5.18</a> |
| C0h    | MSGSTATUS_0  | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.19</a> |
| C4h    | MSGSTATUS_1  | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.20</a> |

**Table 17-14. MAILBOX REGISTERS (continued)**

| Offset | Acronym         | Register Name   | Section                           |
|--------|-----------------|---|-----------------------------------|
| C8h    | MSGSTATUS_2     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.21</a> |
| CCh    | MSGSTATUS_3     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.22</a> |
| D0h    | MSGSTATUS_4     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.23</a> |
| D4h    | MSGSTATUS_5     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.24</a> |
| D8h    | MSGSTATUS_6     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.25</a> |
| DCh    | MSGSTATUS_7     | The message status register has the status of the messages in the mailbox   | <a href="#">Section 17.1.5.26</a> |
| 100h   | IRQSTATUS_RAW_0 | The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.<br>This register is mainly used for debug purpose. | <a href="#">Section 17.1.5.27</a> |
| 104h   | IRQSTATUS_CLR_0 | The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.                           | <a href="#">Section 17.1.5.28</a> |
| 108h   | IRQENABLE_SET_0 | The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to set.  | <a href="#">Section 17.1.5.29</a> |
| 10Ch   | IRQENABLE_CLR_0 | The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to clear.  | <a href="#">Section 17.1.5.30</a> |
| 110h   | IRQSTATUS_RAW_1 | The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.<br>This register is mainly used for debug purpose. | <a href="#">Section 17.1.5.31</a> |
| 114h   | IRQSTATUS_CLR_1 | The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.                           | <a href="#">Section 17.1.5.32</a> |
| 118h   | IRQENABLE_SET_1 | The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to set.  | <a href="#">Section 17.1.5.33</a> |
| 11Ch   | IRQENABLE_CLR_1 | The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to clear.  | <a href="#">Section 17.1.5.34</a> |
| 120h   | IRQSTATUS_RAW_2 | The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.<br>This register is mainly used for debug purpose. | <a href="#">Section 17.1.5.35</a> |
| 124h   | IRQSTATUS_CLR_2 | The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.                           | <a href="#">Section 17.1.5.36</a> |
| 128h   | IRQENABLE_SET_2 | The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to set.  | <a href="#">Section 17.1.5.37</a> |

**Table 17-14. MAILBOX REGISTERS (continued)**

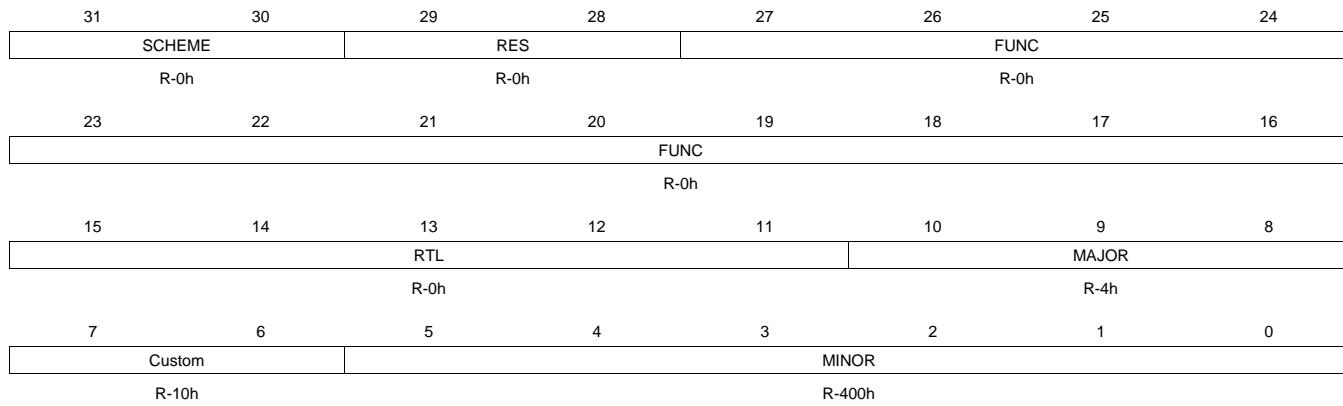
| Offset | Acronym         | Register Name   | Section                           |
|--------|-----------------|---|-----------------------------------|
| 12Ch   | IRQENABLE_CLR_2 | The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to clear.  | <a href="#">Section 17.1.5.38</a> |
| 130h   | IRQSTATUS_RAW_3 | The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.<br>This register is mainly used for debug purpose. | <a href="#">Section 17.1.5.39</a> |
| 134h   | IRQSTATUS_CLR_3 | The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.                           | <a href="#">Section 17.1.5.40</a> |
| 138h   | IRQENABLE_SET_3 | The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to set.  | <a href="#">Section 17.1.5.41</a> |
| 13Ch   | IRQENABLE_CLR_3 | The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user.<br>This register is write 1 to clear.  | <a href="#">Section 17.1.5.42</a> |

### 17.1.5.1 REVISION Register (offset = 0h) [reset = 400h]

REVISION is shown in [Figure 17-3](#) and described in [Table 17-15](#).

This register contains the IP revision code

**Figure 17-3. REVISION Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-15. REVISION Register Field Descriptions**

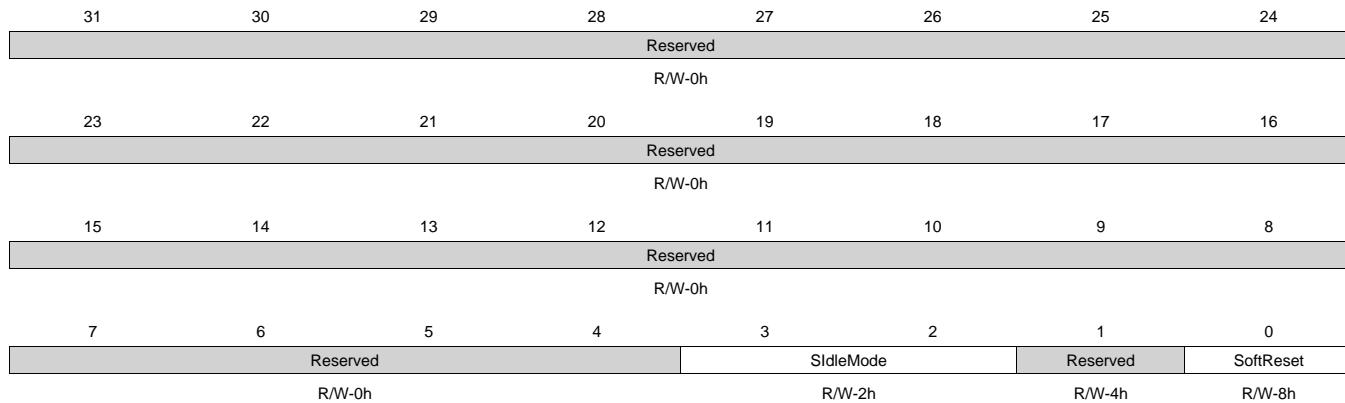
| Bit   | Field  | Type | Reset | Description       |
|-------|--------|------|-------|-------------------|
| 31-30 | SCHEME | R    | 0h    | Not defined yet   |
| 29-28 | RES    | R    | 0h    | Reserved          |
| 27-16 | FUNC   | R    | 0h    | Not defined yet   |
| 15-11 | RTL    | R    | 0h    | Not defined yet   |
| 10-8  | MAJOR  | R    | 4h    | IP-Major Revision |
| 7-6   | Custom | R    | 10h   | Not Defined Yet   |
| 5-0   | MINOR  | R    | 400h  | IP-Minor Revision |

### 17.1.5.2 SYS CONFIG Register (offset = 10h) [reset = 8h]

SYS CONFIG is shown in [Figure 17-4](#) and described in [Table 17-16](#).

This register controls the various parameters of the OCP interface

**Figure 17-4. SYS CONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-16. SYS CONFIG Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-4 | Reserved  | R/W  | 0h    | Write 0's for future compatibility Reads returns 0   |
| 3-2  | SIdleMode | R/W  | 2h    |  |
| 1    | Reserved  | R/W  | 4h    | Write 0's for future compatibility Read returns 0  |
| 0    | SoftReset | R/W  | 8h    | Software reset.<br>This bit is automatically reset by the hardware.<br>During reads, it always return 0<br>0 = Normal : Normal mode<br>1 = Reset : The module is reset |

### 17.1.5.3 MESSAGE\_0 Register (offset = 40h) [reset = 0h]

MESSAGE\_0 is shown in [Figure 17-5](#) and described in [Table 17-17](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-5. MESSAGE\_0 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-17. MESSAGE\_0 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

#### 17.1.5.4 MESSAGE\_1 Register (offset = 44h) [reset = 0h]

MESSAGE\_1 is shown in [Figure 17-6](#) and described in [Table 17-18](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-6. MESSAGE\_1 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-18. MESSAGE\_1 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.5 MESSAGE\_2 Register (offset = 48h) [reset = 0h]

MESSAGE\_2 is shown in [Figure 17-7](#) and described in [Table 17-19](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-7. MESSAGE\_2 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-19. MESSAGE\_2 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.6 MESSAGE\_3 Register (offset = 4Ch) [reset = 0h]

MESSAGE\_3 is shown in [Figure 17-8](#) and described in [Table 17-20](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-8. MESSAGE\_3 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-20. MESSAGE\_3 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.7 MESSAGE\_4 Register (offset = 50h) [reset = 0h]

MESSAGE\_4 is shown in [Figure 17-9](#) and described in [Table 17-21](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-9. MESSAGE\_4 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-21. MESSAGE\_4 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.8 MESSAGE\_5 Register (offset = 54h) [reset = 0h]

MESSAGE\_5 is shown in [Figure 17-10](#) and described in [Table 17-22](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-10. MESSAGE\_5 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-22. MESSAGE\_5 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### **17.1.5.9 MESSAGE\_6 Register (offset = 58h) [reset = 0h]**

MESSAGE\_6 is shown in [Figure 17-11](#) and described in [Table 17-23](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-11. MESSAGE\_6 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-23. MESSAGE\_6 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.10 MESSAGE\_7 Register (offset = 5Ch) [reset = 0h]

MESSAGE\_7 is shown in [Figure 17-12](#) and described in [Table 17-24](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-12. MESSAGE\_7 Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| MESSAGEVALUEMBM |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-24. MESSAGE\_7 Register Field Descriptions**

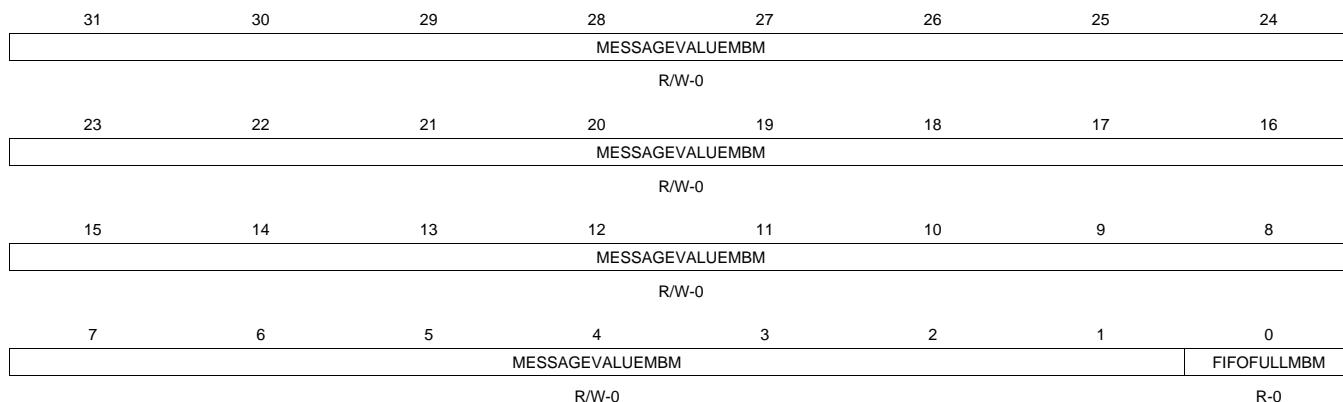
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-0 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |

### 17.1.5.11 FIFOSTATUS\_0 Register (offset = 80h) [reset = 0h]

FIFOSTATUS\_0 is shown in [Figure 17-13](#) and described in [Table 17-25](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-13. FIFOSTATUS\_0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-25. FIFOSTATUS\_0 Register Field Descriptions**

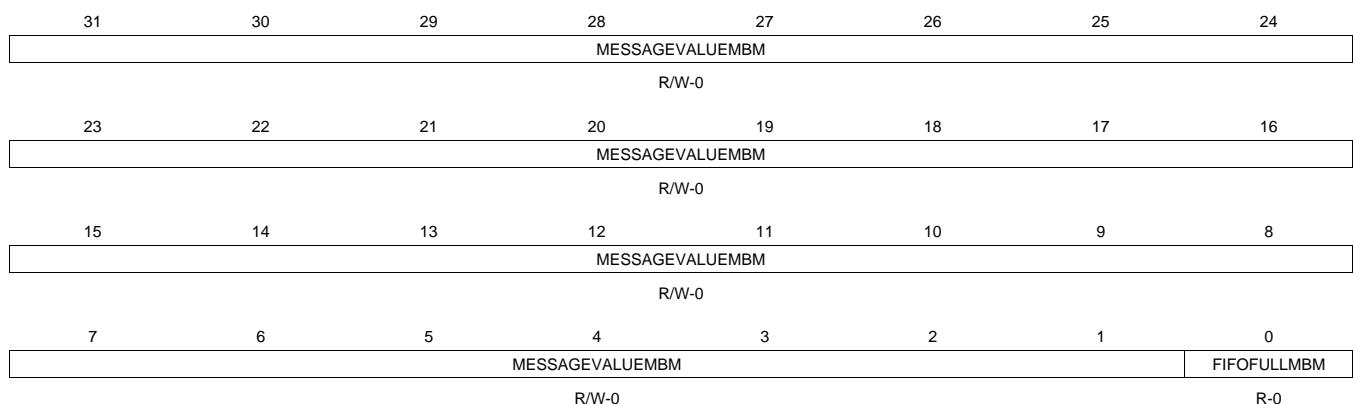
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.12 FIFOSTATUS\_1 Register (offset = 84h) [reset = 0h]

FIFOSTATUS\_1 is shown in [Figure 17-14](#) and described in [Table 17-26](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-14. FIFOSTATUS\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-26. FIFOSTATUS\_1 Register Field Descriptions**

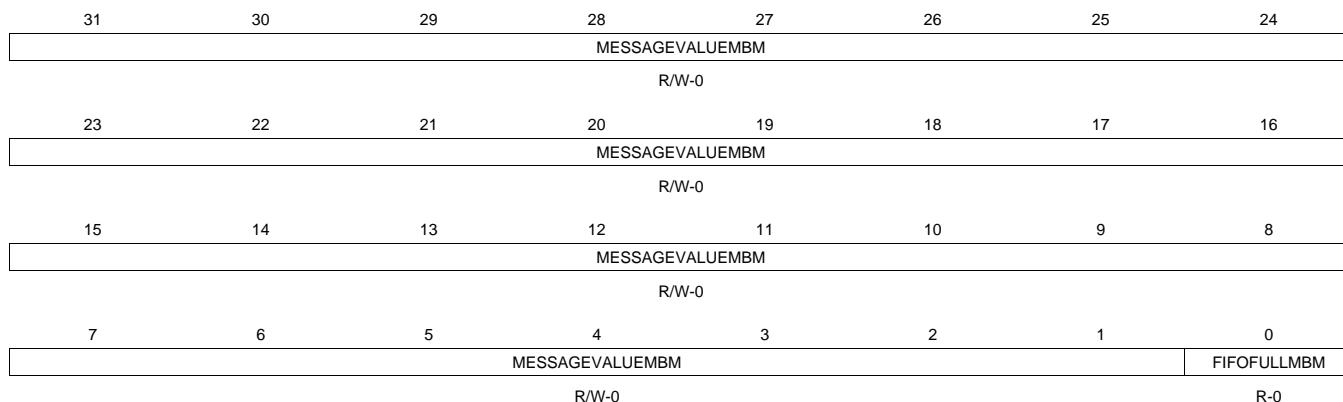
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.13 FIFOSTATUS\_2 Register (offset = 88h) [reset = 0h]

FIFOSTATUS\_2 is shown in [Figure 17-15](#) and described in [Table 17-27](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-15. FIFOSTATUS\_2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-27. FIFOSTATUS\_2 Register Field Descriptions**

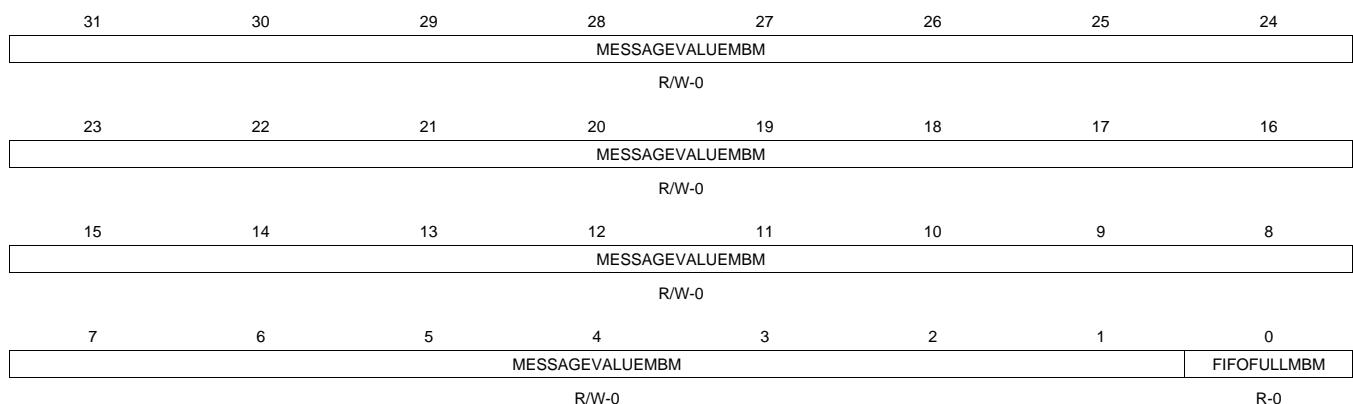
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.14 FIFOSTATUS\_3 Register (offset = 8Ch) [reset = 0h]

FIFOSTATUS\_3 is shown in [Figure 17-16](#) and described in [Table 17-28](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-16. FIFOSTATUS\_3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-28. FIFOSTATUS\_3 Register Field Descriptions**

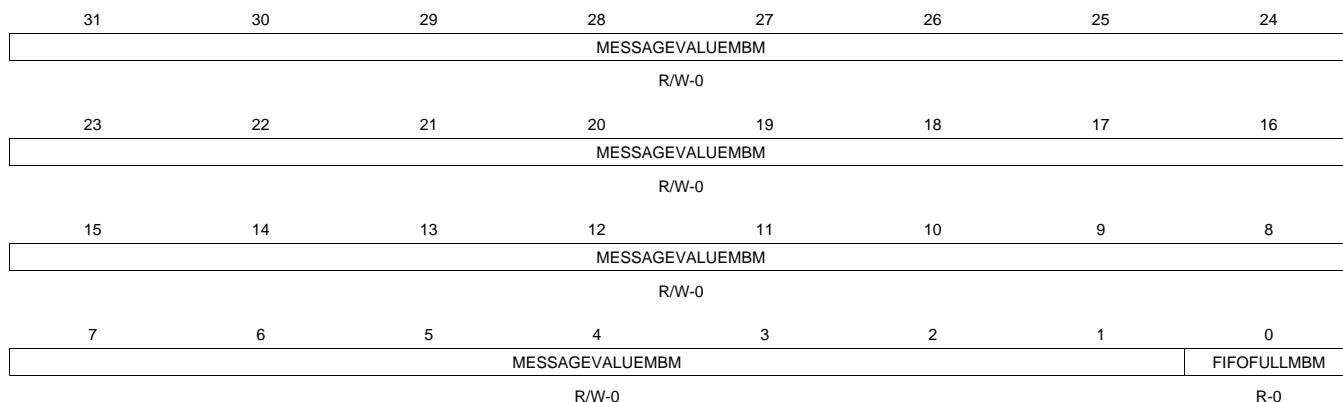
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.15 FIFOSTATUS\_4 Register (offset = 90h) [reset = 0h]

FIFOSTATUS\_4 is shown in [Figure 17-17](#) and described in [Table 17-29](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-17. FIFOSTATUS\_4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-29. FIFOSTATUS\_4 Register Field Descriptions**

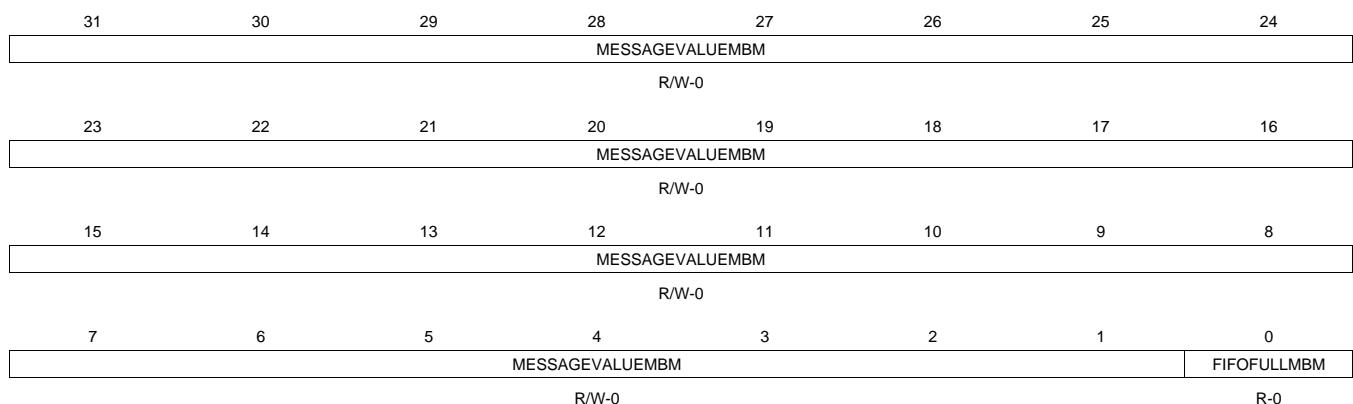
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.16 FIFOSTATUS\_5 Register (offset = 94h) [reset = 0h]

FIFOSTATUS\_5 is shown in [Figure 17-18](#) and described in [Table 17-30](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-18. FIFOSTATUS\_5 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-30. FIFOSTATUS\_5 Register Field Descriptions**

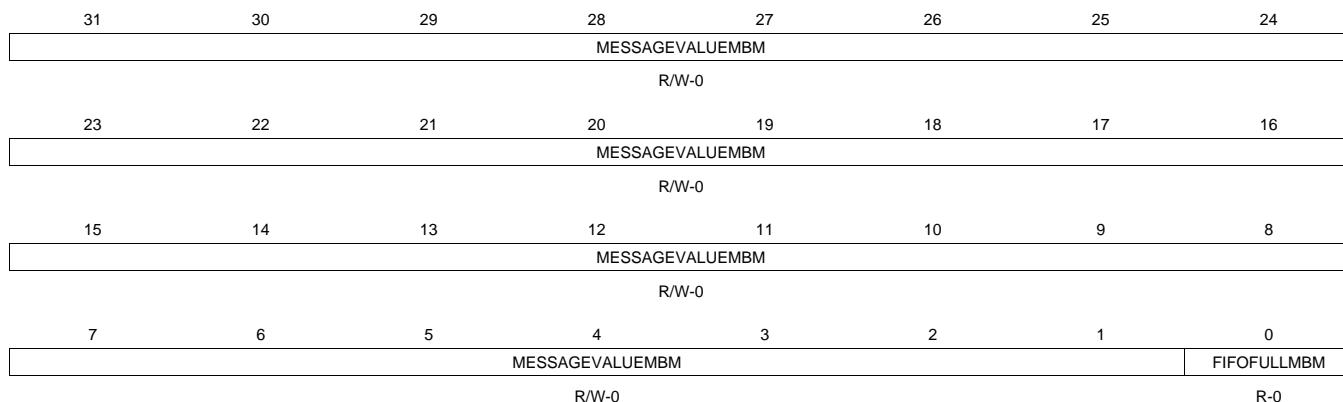
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.17 FIFOSTATUS\_6 Register (offset = 98h) [reset = 0h]

FIFOSTATUS\_6 is shown in [Figure 17-19](#) and described in [Table 17-31](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue

**Figure 17-19. FIFOSTATUS\_6 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-31. FIFOSTATUS\_6 Register Field Descriptions**

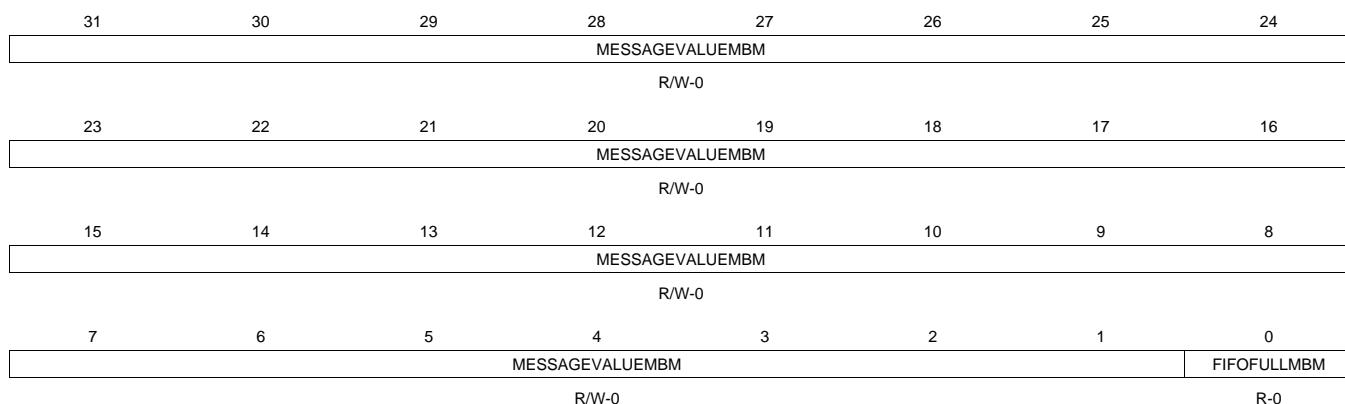
| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.18 FIFOSTATUS\_7 Register (offset = 9Ch) [reset = 0h]

FIFOSTATUS\_7 is shown in [Figure 17-20](#) and described in [Table 17-32](#).

The message register stores the next to be read message of the mailbox. Reads remove the message from the FIFO queue.

**Figure 17-20. FIFOSTATUS\_7 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-32. FIFOSTATUS\_7 Register Field Descriptions**

| Bit  | Field           | Type  | Reset | Description   |
|------|-----------------|-------|-------|---|
| 31-1 | MESSAGEVALUEMBM | R/W-0 | 0     | Message in Mailbox.<br>The message register stores the next to be read message of the mailbox.<br>Reads remove the message from the FIFO queue. |
| 0    | FIFOFULLMBM     | R-0   | 0     | Full flag for Mailbox<br>0 = NotFull : Mailbox FIFO is not full<br>1 = Full : Mailbox FIFO is full  |

### 17.1.5.19 MSGSTATUS\_0 Register (offset = C0h) [reset = 0h]

MSGSTATUS\_0 is shown in [Figure 17-21](#) and described in [Table 17-33](#).

The message status register has the status of the messages in the mailbox

**Figure 17-21. MSGSTATUS\_0 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-33. MSGSTATUS\_0 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.20 MSGSTATUS\_1 Register (offset = C4h) [reset = 0h]

MSGSTATUS\_1 is shown in [Figure 17-22](#) and described in [Table 17-34](#).

The message status register has the status of the messages in the mailbox

**Figure 17-22. MSGSTATUS\_1 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |            |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-34. MSGSTATUS\_1 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.21 MSGSTATUS\_2 Register (offset = C8h) [reset = 0h]

MSGSTATUS\_2 is shown in [Figure 17-23](#) and described in [Table 17-35](#).

The message status register has the status of the messages in the mailbox

**Figure 17-23. MSGSTATUS\_2 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-35. MSGSTATUS\_2 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.22 MSGSTATUS\_3 Register (offset = CCh) [reset = 0h]

MSGSTATUS\_3 is shown in [Figure 17-24](#) and described in [Table 17-36](#).

The message status register has the status of the messages in the mailbox

**Figure 17-24. MSGSTATUS\_3 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |            |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-36. MSGSTATUS\_3 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.23 MSGSTATUS\_4 Register (offset = D0h) [reset = 0h]

MSGSTATUS\_4 is shown in [Figure 17-25](#) and described in [Table 17-37](#).

The message status register has the status of the messages in the mailbox

**Figure 17-25. MSGSTATUS\_4 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-37. MSGSTATUS\_4 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.24 MSGSTATUS\_5 Register (offset = D4h) [reset = 0h]

MSGSTATUS\_5 is shown in [Figure 17-26](#) and described in [Table 17-38](#).

The message status register has the status of the messages in the mailbox

**Figure 17-26. MSGSTATUS\_5 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |            |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-38. MSGSTATUS\_5 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.25 MSGSTATUS\_6 Register (offset = D8h) [reset = 0h]

MSGSTATUS\_6 is shown in [Figure 17-27](#) and described in [Table 17-39](#).

The message status register has the status of the messages in the mailbox

**Figure 17-27. MSGSTATUS\_6 Register**

| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2          | 1 | 0 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|------------|---|---|
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   | NBOFMSGMBM |   |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-39. MSGSTATUS\_6 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.26 MSGSTATUS\_7 Register (offset = DCh) [reset = 0h]

MSGSTATUS\_7 is shown in [Figure 17-28](#) and described in [Table 17-40](#).

The message status register has the status of the messages in the mailbox

**Figure 17-28. MSGSTATUS\_7 Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |            |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|------------|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
| Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   | NBOFMSGMBM |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-40. MSGSTATUS\_7 Register Field Descriptions**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2-0 | NBOFMSGMBM | R    | 0h    | Number of unread messages in Mailbox.<br>Limited to four messages per mailbox. |

### 17.1.5.27 IRQSTATUS\_RAW\_0 Register (offset = 100h) [reset = 0h]

IRQSTATUS\_RAW\_0 is shown in [Figure 17-29](#) and described in [Table 17-41](#).

The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit. This register is mainly used for debug purpose.

**Figure 17-29. IRQSTATUS\_RAW\_0 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-41. IRQSTATUS\_RAW\_0 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-41. IRQSTATUS\_RAW\_0 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.28 IRQSTATUS\_CLR\_0 Register (offset = 104h) [reset = 0h]

IRQSTATUS\_CLR\_0 is shown in [Figure 17-30](#) and described in [Table 17-42](#).

The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.

**Figure 17-30. IRQSTATUS\_CLR\_0 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-42. IRQSTATUS\_CLR\_0 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-42. IRQSTATUS\_CLR\_0 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.29 IRQENABLE\_SET\_0 Register (offset = 108h) [reset = 0h]

IRQENABLE\_SET\_0 is shown in [Figure 17-31](#) and described in [Table 17-43](#).

The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user. This register is write 1 to set.

**Figure 17-31. IRQENABLE\_SET\_0 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-43. IRQENABLE\_SET\_0 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-43. IRQENABLE\_SET\_0 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.30 IRQENABLE\_CLR\_0 Register (offset = 10Ch) [reset = 0h]

IRQENABLE\_CLR\_0 is shown in [Figure 17-32](#) and described in [Table 17-44](#).

The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user. This register is write 1 to clear.

**Figure 17-32. IRQENABLE\_CLR\_0 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-44. IRQENABLE\_CLR\_0 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-44. IRQENABLE\_CLR\_0 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.31 IRQSTATUS\_RAW\_1 Register (offset = 110h) [reset = 0h]

IRQSTATUS\_RAW\_1 is shown in [Figure 17-33](#) and described in [Table 17-45](#).

The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit. This register is mainly used for debug purpose.

**Figure 17-33. IRQSTATUS\_RAW\_1 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-45. IRQSTATUS\_RAW\_1 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-45. IRQSTATUS\_RAW\_1 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.32 IRQSTATUS\_CLR\_1 Register (offset = 114h) [reset = 0h]

IRQSTATUS\_CLR\_1 is shown in [Figure 17-34](#) and described in [Table 17-46](#).

The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.

**Figure 17-34. IRQSTATUS\_CLR\_1 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-46. IRQSTATUS\_CLR\_1 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-46. IRQSTATUS\_CLR\_1 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.33 IRQENABLE\_SET\_1 Register (offset = 118h) [reset = 0h]

IRQENABLE\_SET\_1 is shown in [Figure 17-35](#) and described in [Table 17-47](#).

The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user. This register is write 1 to set.

**Figure 17-35. IRQENABLE\_SET\_1 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-47. IRQENABLE\_SET\_1 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-47. IRQENABLE\_SET\_1 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.34 IRQENABLE\_CLR\_1 Register (offset = 11Ch) [reset = 0h]

IRQENABLE\_CLR\_1 is shown in [Figure 17-36](#) and described in [Table 17-48](#).

The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user. This register is write 1 to clear.

**Figure 17-36. IRQENABLE\_CLR\_1 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-48. IRQENABLE\_CLR\_1 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-48. IRQENABLE\_CLR\_1 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.35 IRQSTATUS\_RAW\_2 Register (offset = 120h) [reset = 0h]

IRQSTATUS\_RAW\_2 is shown in [Figure 17-37](#) and described in [Table 17-49](#).

The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit. This register is mainly used for debug purpose.

**Figure 17-37. IRQSTATUS\_RAW\_2 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-49. IRQSTATUS\_RAW\_2 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-49. IRQSTATUS\_RAW\_2 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.36 IRQSTATUS\_CLR\_2 Register (offset = 124h) [reset = 0h]

IRQSTATUS\_CLR\_2 is shown in [Figure 17-38](#) and described in [Table 17-50](#).

The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.

**Figure 17-38. IRQSTATUS\_CLR\_2 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-50. IRQSTATUS\_CLR\_2 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-50. IRQSTATUS\_CLR\_2 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.37 IRQENABLE\_SET\_2 Register (offset = 128h) [reset = 0h]

IRQENABLE\_SET\_2 is shown in [Figure 17-39](#) and described in [Table 17-51](#).

The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user. This register is write 1 to set.

**Figure 17-39. IRQENABLE\_SET\_2 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-51. IRQENABLE\_SET\_2 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-51. IRQENABLE\_SET\_2 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.38 IRQENABLE\_CLR\_2 Register (offset = 12Ch) [reset = 0h]

IRQENABLE\_CLR\_2 is shown in [Figure 17-40](#) and described in [Table 17-52](#).

The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user. This register is write 1 to clear.

**Figure 17-40. IRQENABLE\_CLR\_2 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-52. IRQENABLE\_CLR\_2 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-52. IRQENABLE\_CLR\_2 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.39 IRQSTATUS\_RAW\_3 Register (offset = 130h) [reset = 0h]

IRQSTATUS\_RAW\_3 is shown in [Figure 17-41](#) and described in [Table 17-53](#).

The interrupt status register has the status for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit. This register is mainly used for debug purpose.

**Figure 17-41. IRQSTATUS\_RAW\_3 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-53. IRQSTATUS\_RAW\_3 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-53. IRQSTATUS\_RAW\_3 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.40 IRQSTATUS\_CLR\_3 Register (offset = 134h) [reset = 0h]

IRQSTATUS\_CLR\_3 is shown in [Figure 17-42](#) and described in [Table 17-54](#).

The interrupt status register has the status combined with irq-enable for each event that may be responsible for the generation of an interrupt to the corresponding user - write 1 to a given bit resets this bit.

**Figure 17-42. IRQSTATUS\_CLR\_3 Register**

|                         |                       |                         |                       |                         |                       |                         |                       |
|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|-------------------------|-----------------------|
| 31                      | 30                    | 29                      | 28                    | 27                      | 26                    | 25                      | 24                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 23                      | 22                    | 21                      | 20                    | 19                      | 18                    | 17                      | 16                    |
| Reserved                |                       |                         |                       |                         |                       |                         |                       |
| 15                      | 14                    | 13                      | 12                    | 11                      | 10                    | 9                       | 8                     |
| NotFullStatusUuMB7<br>7 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB6<br>6 | NewMSGStatusUuMB<br>6 | NotFullStatusUuMB5<br>5 | NewMSGStatusUuMB<br>5 | NotFullStatusUuMB4<br>4 | NewMSGStatusUuMB<br>4 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |
| 7                       | 6                     | 5                       | 4                     | 3                       | 2                     | 1                       | 0                     |
| NotFullStatusUuMB3<br>3 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB2<br>2 | NewMSGStatusUuMB<br>2 | NotFullStatusUuMB1<br>1 | NewMSGStatusUuMB<br>1 | NotFullStatusUuMB0<br>0 | NewMSGStatusUuMB<br>0 |
| R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                | R/W-0h                  | R/W-0h                |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-54. IRQSTATUS\_CLR\_3 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-54. IRQSTATUS\_CLR\_3 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.41 IRQENABLE\_SET\_3 Register (offset = 138h) [reset = 0h]

IRQENABLE\_SET\_3 is shown in [Figure 17-43](#) and described in [Table 17-55](#).

The interrupt enable register enables to unmask the module internal source of interrupt to the corresponding user. This register is write 1 to set.

**Figure 17-43. IRQENABLE\_SET\_3 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-55. IRQENABLE\_SET\_3 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-55. IRQENABLE\_SET\_3 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

### 17.1.5.42 IRQENABLE\_CLR\_3 Register (offset = 13Ch) [reset = 0h]

IRQENABLE\_CLR\_3 is shown in [Figure 17-44](#) and described in [Table 17-56](#).

The interrupt enable register enables to mask the module internal source of interrupt to the corresponding user. This register is write 1 to clear.

**Figure 17-44. IRQENABLE\_CLR\_3 Register**

|                    |                   |                    |                   |                    |                   |                    |                   |
|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|--------------------|-------------------|
| 31                 | 30                | 29                 | 28                | 27                 | 26                | 25                 | 24                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 23                 | 22                | 21                 | 20                | 19                 | 18                | 17                 | 16                |
| Reserved           |                   |                    |                   |                    |                   |                    |                   |
| 15                 | 14                | 13                 | 12                | 11                 | 10                | 9                  | 8                 |
| NotFullStatusUuMB7 | NewMSGStatusUuMB7 | NotFullStatusUuMB6 | NewMSGStatusUuMB6 | NotFullStatusUuMB5 | NewMSGStatusUuMB5 | NotFullStatusUuMB4 | NewMSGStatusUuMB4 |
| 7                  | 6                 | 5                  | 4                 | 3                  | 2                 | 1                  | 0                 |
| NotFullStatusUuMB3 | NewMSGStatusUuMB3 | NotFullStatusUuMB2 | NewMSGStatusUuMB2 | NotFullStatusUuMB1 | NewMSGStatusUuMB1 | NotFullStatusUuMB0 | NewMSGStatusUuMB0 |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |
| R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            | R/W-0h             | R/W-0h            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-56. IRQENABLE\_CLR\_3 Register Field Descriptions**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 15  | NotFullStatusUuMB7 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 14  | NewMSGStatusUuMB7  | R/W  | 0h    | New Message Status bit for User u, Mailbox 7<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 13  | NotFullStatusUuMB6 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 12  | NewMSGStatusUuMB6  | R/W  | 0h    | New Message Status bit for User u, Mailbox 6<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 11  | NotFullStatusUuMB5 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 10  | NewMSGStatusUuMB5  | R/W  | 0h    | New Message Status bit for User u, Mailbox 5<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 9   | NotFullStatusUuMB4 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 8   | NewMSGStatusUuMB4  | R/W  | 0h    | New Message Status bit for User u, Mailbox 4<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 7   | NotFullStatusUuMB3 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |

**Table 17-56. IRQENABLE\_CLR\_3 Register Field Descriptions (continued)**

| Bit | Field              | Type | Reset | Description  |
|-----|--------------------|------|-------|--|
| 6   | NewMSGStatusUuMB3  | R/W  | 0h    | New Message Status bit for User u, Mailbox 3<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 5   | NotFullStatusUuMB2 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 4   | NewMSGStatusUuMB2  | R/W  | 0h    | New Message Status bit for User u, Mailbox 2<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 3   | NotFullStatusUuMB1 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 2   | NewMSGStatusUuMB1  | R/W  | 0h    | New Message Status bit for User u, Mailbox 1<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |
| 1   | NotFullStatusUuMB0 | R/W  | 0h    | Not Full Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug)    |
| 0   | NewMSGStatusUuMB0  | R/W  | 0h    | New Message Status bit for User u, Mailbox 0<br>0 = NoAction : No action<br>1 = SetEvent : Set the event (for debug) |

## 17.2 Spinlock

### 17.2.1 SPINLOCK Registers

**Table 17-57** lists the memory-mapped registers for the SPINLOCK. All register offset addresses not listed in [Table 17-57](#) should be considered as reserved locations and the register contents should not be modified.

**Table 17-57. SPINLOCK REGISTERS**

| Offset | Acronym    | Register Name   | Section                           |
|--------|------------|---|-----------------------------------|
| 0h     | REV        | Read-only IP revision identifier (X.Y.R) used by software to determine features, bugs and compatibility of an instance of this the Spin Lock module.  | <a href="#">Section 17.2.1.1</a>  |
| 10h    | SYSCONFIG  | This register controls the various parameters of the OCP interface.<br>Note that several fields are present by read-only.   | <a href="#">Section 17.2.1.2</a>  |
| 14h    | SYSTATUS   | This register provides status information about this instance of the Spin Lock module.  | <a href="#">Section 17.2.1.3</a>  |
| 800h   | LOCK_REG_0 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.4</a>  |
| 804h   | LOCK_REG_1 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.5</a>  |
| 808h   | LOCK_REG_2 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.6</a>  |
| 80Ch   | LOCK_REG_3 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.7</a>  |
| 810h   | LOCK_REG_4 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.8</a>  |
| 814h   | LOCK_REG_5 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.9</a>  |
| 818h   | LOCK_REG_6 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.10</a> |
| 81Ch   | LOCK_REG_7 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.11</a> |
| 820h   | LOCK_REG_8 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.12</a> |

**Table 17-57. SPINLOCK REGISTERS (continued)**

| Offset | Acronym     | Register Name   | Section                           |
|--------|-------------|---|-----------------------------------|
| 824h   | LOCK_REG_9  | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.13</a> |
| 828h   | LOCK_REG_10 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.14</a> |
| 82Ch   | LOCK_REG_11 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.15</a> |
| 830h   | LOCK_REG_12 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.16</a> |
| 834h   | LOCK_REG_13 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.17</a> |
| 838h   | LOCK_REG_14 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.18</a> |
| 83Ch   | LOCK_REG_15 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.19</a> |
| 840h   | LOCK_REG_16 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.20</a> |
| 844h   | LOCK_REG_17 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.21</a> |
| 848h   | LOCK_REG_18 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.22</a> |
| 84Ch   | LOCK_REG_19 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.23</a> |
| 850h   | LOCK_REG_20 | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.24</a> |

**Table 17-57. SPINLOCK REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b> | <b>Register Name</b>  | <b>Section</b>                    |
|---------------|----------------|---|-----------------------------------|
| 854h          | LOCK_REG_21    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.25</a> |
| 858h          | LOCK_REG_22    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.26</a> |
| 85Ch          | LOCK_REG_23    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.27</a> |
| 860h          | LOCK_REG_24    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.28</a> |
| 864h          | LOCK_REG_25    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.29</a> |
| 868h          | LOCK_REG_26    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.30</a> |
| 86Ch          | LOCK_REG_27    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.31</a> |
| 870h          | LOCK_REG_28    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.32</a> |
| 874h          | LOCK_REG_29    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.33</a> |
| 878h          | LOCK_REG_30    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.34</a> |
| 87Ch          | LOCK_REG_31    | This register is read when attempting to acquire a lock.<br>The lock is automatically taken if it was not taken and the value returned by the read is zero.<br>If the lock was already taken, then the read returns one.<br>Writing a zero to this register frees the lock. | <a href="#">Section 17.2.1.35</a> |

### 17.2.1.1 REV Register (offset = 0h) [reset = 50020000h]

REV is shown in [Figure 17-45](#) and described in [Table 17-58](#).

Read-only IP revision identifier (X.Y.R) used by software to determine features, bugs and compatibility of an instance of this the Spin Lock module.

**Figure 17-45. REV Register**

|     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31  | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REV |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-50020000h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-58. REV Register Field Descriptions**

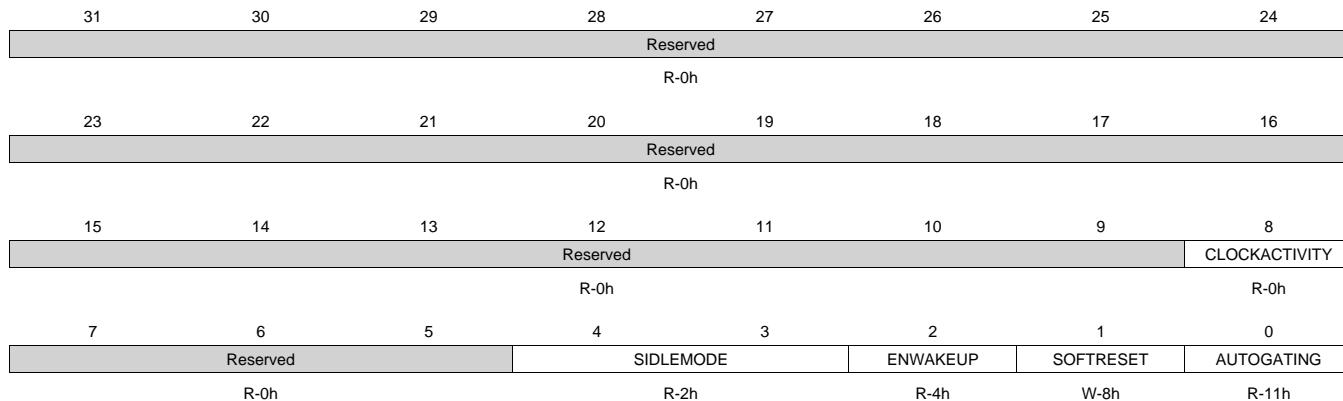
| Bit  | Field | Type | Reset     | Description       |
|------|-------|------|-----------|-------------------|
| 31-0 | REV   | R    | 50020000h | IP Revision Code. |

### 17.2.1.2 SYS CONFIG Register (offset = 10h) [reset = 11h]

SYS CONFIG is shown in [Figure 17-46](#) and described in [Table 17-59](#).

This register controls the various parameters of the OCP interface. Note that several fields are present by read-only.

**Figure 17-46. SYS CONFIG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-59. SYS CONFIG Register Field Descriptions**

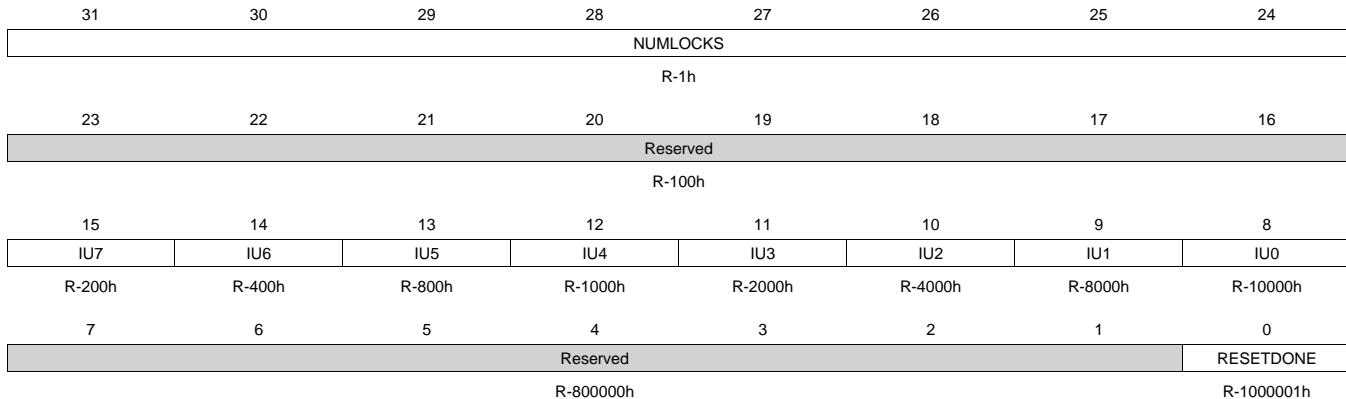
| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-9 | Reserved      | R    | 0h    |   |
| 8    | CLOCKACTIVITY | R    | 0h    | Indicates whether the module requires the OCP when in IDLE mode.<br>0 = NOTREQUIRED : OCP clock is not required by the module during IDLE mode and may be switched off.<br>1 = REQUIRED : OCP clock is required by the module, even during idle mode.   |
| 7-5  | Reserved      | R    | 0h    |   |
| 4-3  | SIDLEMODE     | R    | 2h    | Control of the slave interface power management IDLE request acknowledgement.<br>0 = FORCEIDLE : IDLE request is acknowledged unconditionally and immediately.<br>1 = NOIDLE : IDLE request is never acknowledged.<br>2 = SMARTIDLE : IDLE request acknowledgement is based on the internal module activity.<br>3 = unused3 : Reserved. Do not use. |
| 2    | ENWAKEUP      | R    | 4h    | Asynchronous wakeup generation.<br>0 = NOWAKEUPGEN : Wakeup generation is disabled.<br>1 = WAKEUPGEN1 : Enable wakeup generation.   |
| 1    | SOFTRESET     | W    | 8h    | Module software reset.<br>0 = NOOP : No Description<br>1 = DORESET : Start a soft reset sequence of the Spin Lock module.   |
| 0    | AUTOGATING    | R    | 11h   | Internal OCP clock gating strategy.<br>0 = FREERUNNING : OCP clock is not gated when OCP interface is idle.<br>1 = GATED : Automatic internal OCP clock gating strategy is applied, based on the OCP interface activity.  |

### 17.2.1.3 SYSTATUS Register (offset = 14h) [reset = 1000001h]

SYSTATUS is shown in [Figure 17-47](#) and described in [Table 17-60](#).

This register provides status information about this instance of the Spin Lock module.

**Figure 17-47. SYSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-60. SYSTATUS Register Field Descriptions**

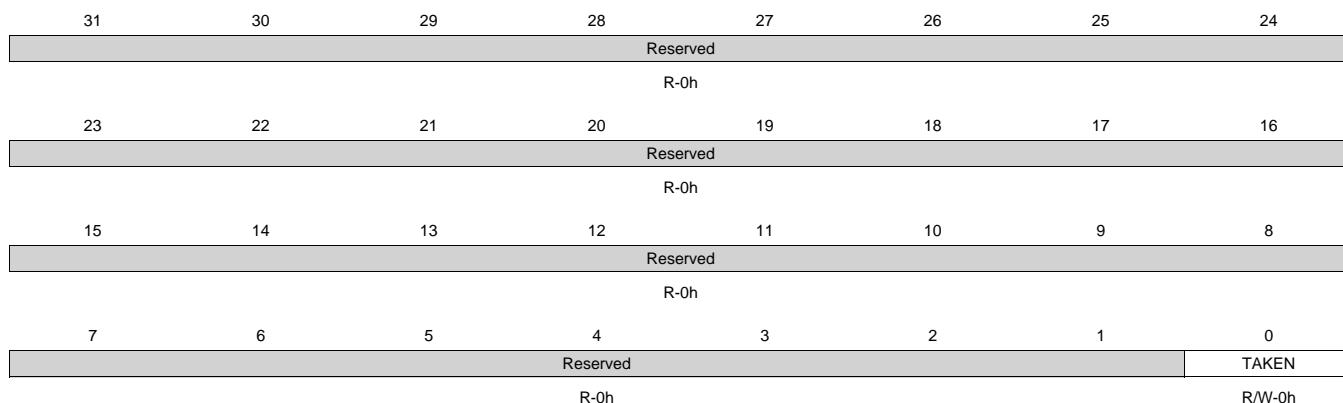
| Bit   | Field     | Type | Reset    | Description  |
|-------|-----------|------|----------|--|
| 31-24 | NUMLOCKS  | R    | 1h       |  |
| 23-16 | Reserved  | R    | 100h     |  |
| 15    | IU7       | R    | 200h     |  |
| 14    | IU6       | R    | 400h     |  |
| 13    | IU5       | R    | 800h     |  |
| 12    | IU4       | R    | 1000h    |  |
| 11    | IU3       | R    | 2000h    |  |
| 10    | IU2       | R    | 4000h    |  |
| 9     | IU1       | R    | 8000h    | In-Use flag 1, covering lock registers 32 - 63.<br>Reads as one only if one or more lock registers in this range are TAKEN.<br>If no lock registers are implemented in this range, then this flag always reads as 0. |
| 8     | IU0       | R    | 10000h   | In-Use flag 0, covering lock registers 0 - 31.<br>Reads as one only if one or more lock registers in this range are TAKEN.   |
| 7-1   | Reserved  | R    | 800000h  | reserved   |
| 0     | RESETDONE | R    | 1000001h | 0: Reset in progress.<br>1: Reset is completed.  |

#### 17.2.1.4 LOCK\_REG\_0 Register (offset = 800h) [reset = 0h]

LOCK\_REG\_0 is shown in [Figure 17-48](#) and described in [Table 17-61](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-48. LOCK\_REG\_0 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-61. LOCK\_REG\_0 Register Field Descriptions**

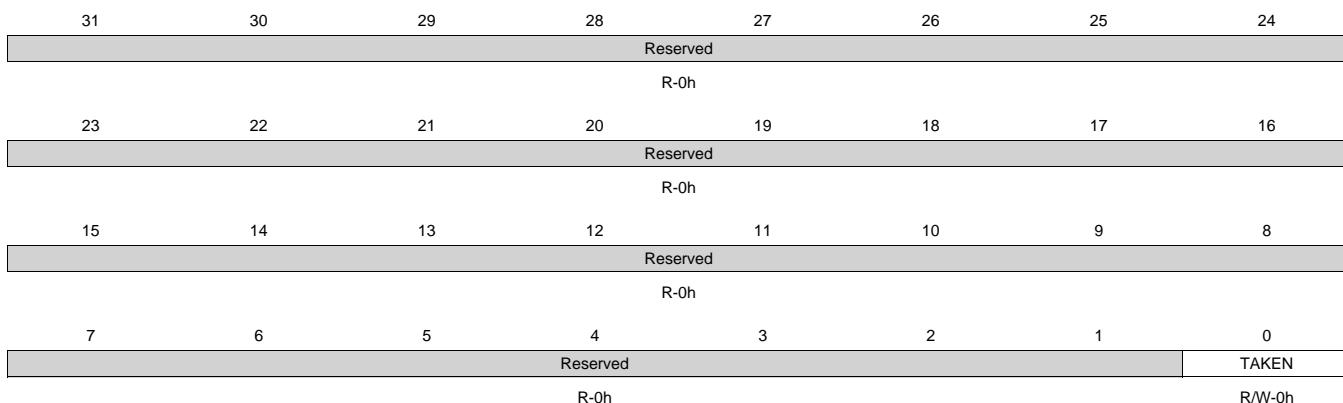
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.5 LOCK\_REG\_1 Register (offset = 804h) [reset = 0h]

LOCK\_REG\_1 is shown in [Figure 17-49](#) and described in [Table 17-62](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-49. LOCK\_REG\_1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-62. LOCK\_REG\_1 Register Field Descriptions**

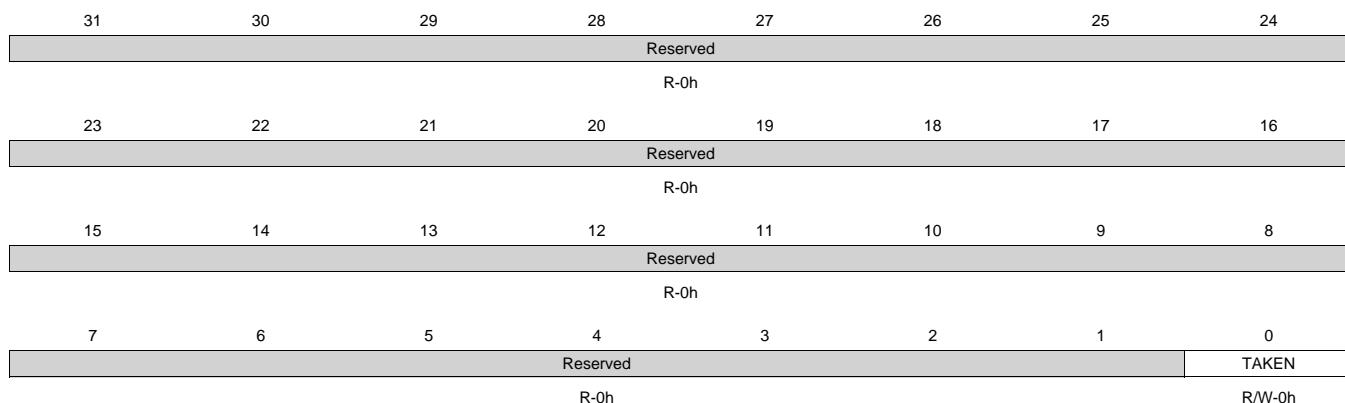
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.6 LOCK\_REG\_2 Register (offset = 808h) [reset = 0h]

LOCK\_REG\_2 is shown in [Figure 17-50](#) and described in [Table 17-63](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-50. LOCK\_REG\_2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-63. LOCK\_REG\_2 Register Field Descriptions**

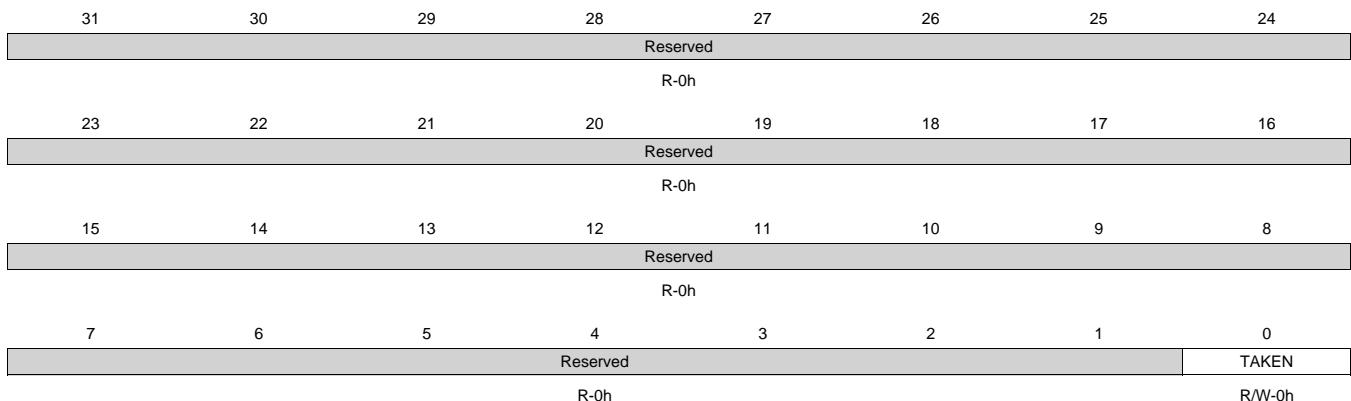
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.7 LOCK\_REG\_3 Register (offset = 80Ch) [reset = 0h]

LOCK\_REG\_3 is shown in [Figure 17-51](#) and described in [Table 17-64](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-51. LOCK\_REG\_3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-64. LOCK\_REG\_3 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

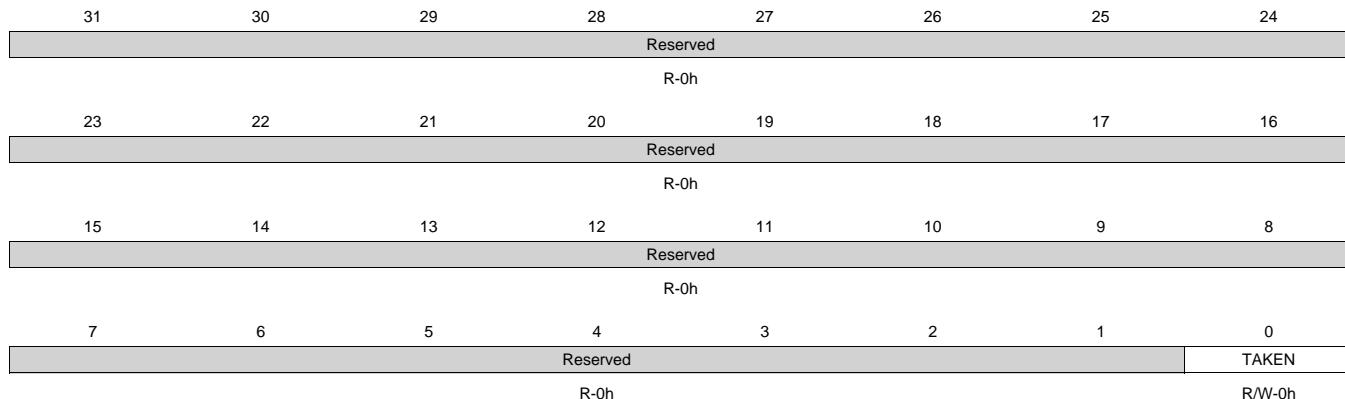
### 17.2.1.8 LOCK\_REG\_4 Register (offset = 810h) [reset = 0h]

LOCK\_REG\_4 is shown in [Figure 17-52](#) and described in [Table 17-65](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one.

Writing a zero to this register frees the lock.

**Figure 17-52. LOCK\_REG\_4 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-65. LOCK\_REG\_4 Register Field Descriptions**

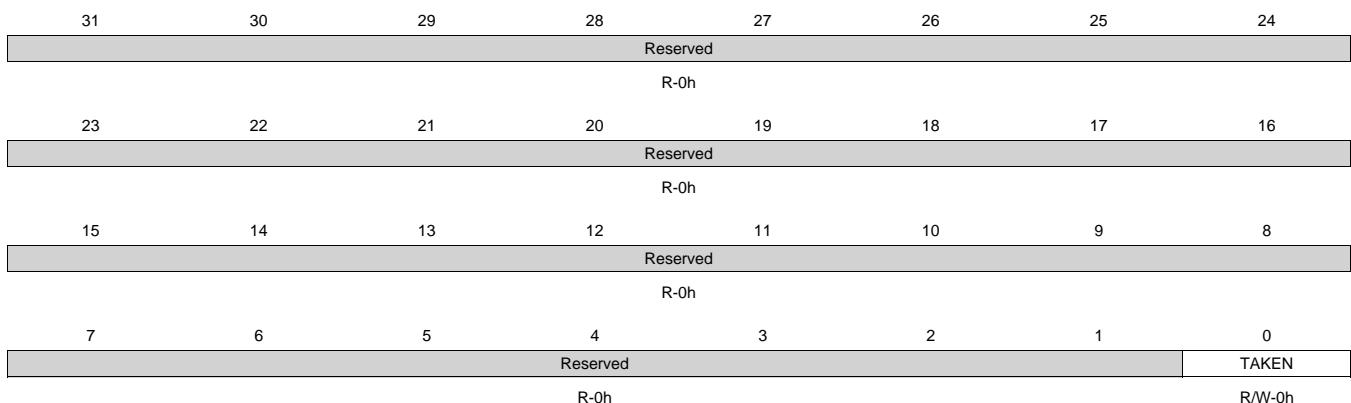
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.9 LOCK\_REG\_5 Register (offset = 814h) [reset = 0h]

LOCK\_REG\_5 is shown in [Figure 17-53](#) and described in [Table 17-66](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-53. LOCK\_REG\_5 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-66. LOCK\_REG\_5 Register Field Descriptions**

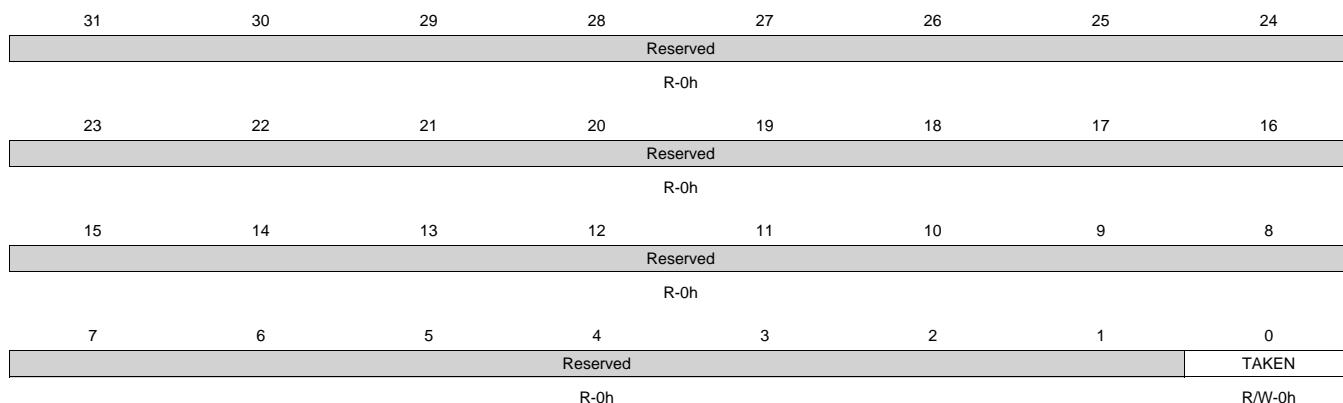
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.10 LOCK\_REG\_6 Register (offset = 818h) [reset = 0h]

LOCK\_REG\_6 is shown in [Figure 17-54](#) and described in [Table 17-67](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-54. LOCK\_REG\_6 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-67. LOCK\_REG\_6 Register Field Descriptions**

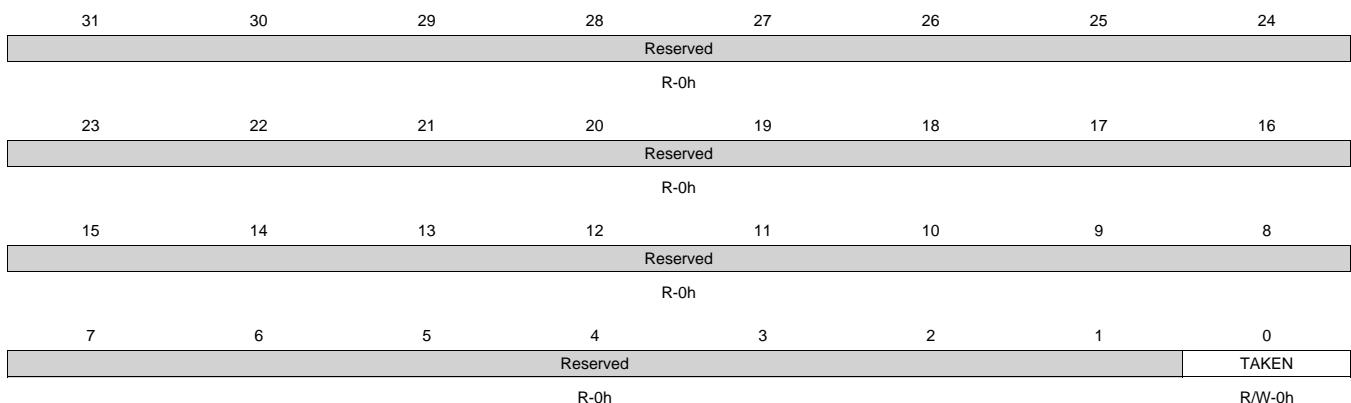
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.11 LOCK\_REG\_7 Register (offset = 81Ch) [reset = 0h]

LOCK\_REG\_7 is shown in [Figure 17-55](#) and described in [Table 17-68](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-55. LOCK\_REG\_7 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-68. LOCK\_REG\_7 Register Field Descriptions**

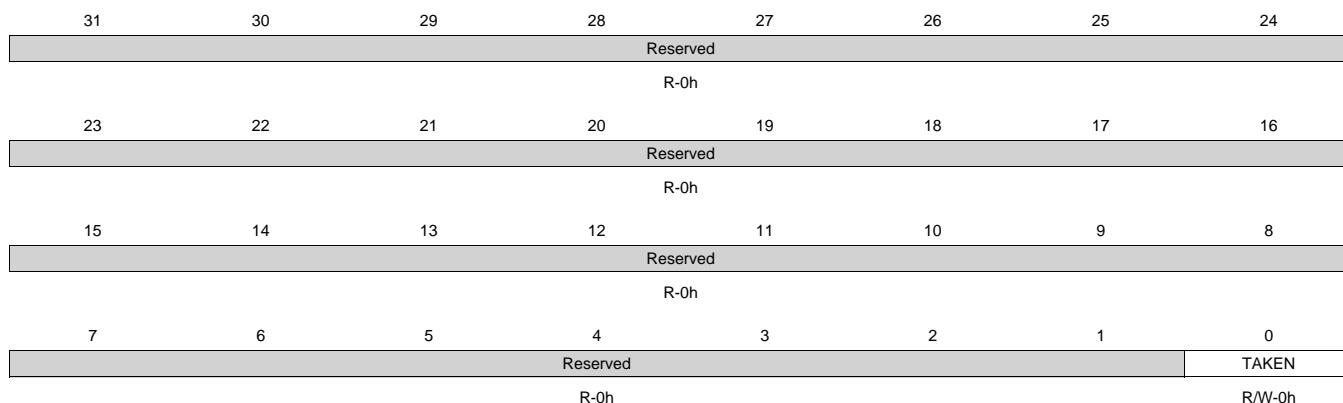
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.12 LOCK\_REG\_8 Register (offset = 820h) [reset = 0h]

LOCK\_REG\_8 is shown in [Figure 17-56](#) and described in [Table 17-69](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-56. LOCK\_REG\_8 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-69. LOCK\_REG\_8 Register Field Descriptions**

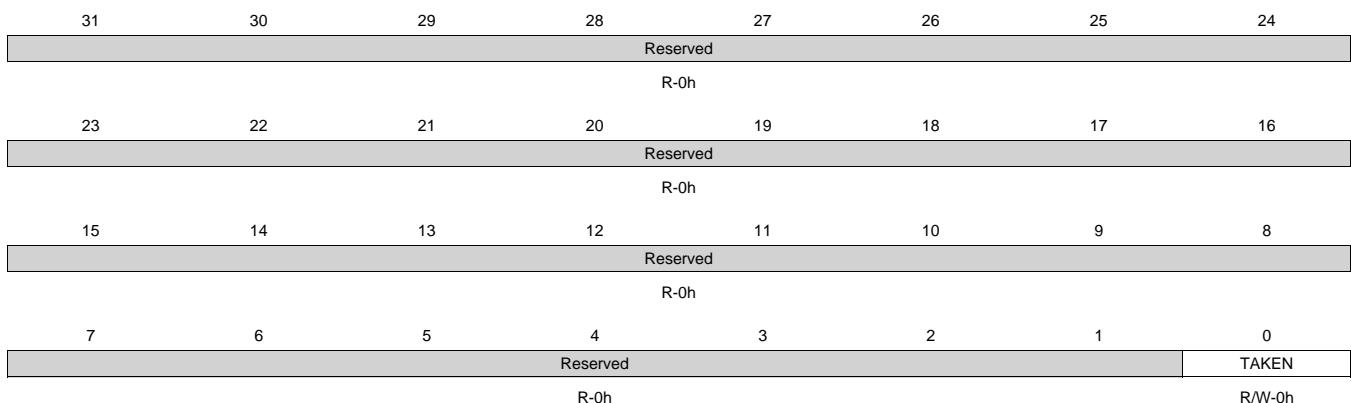
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.13 LOCK\_REG\_9 Register (offset = 824h) [reset = 0h]

LOCK\_REG\_9 is shown in [Figure 17-57](#) and described in [Table 17-70](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-57. LOCK\_REG\_9 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-70. LOCK\_REG\_9 Register Field Descriptions**

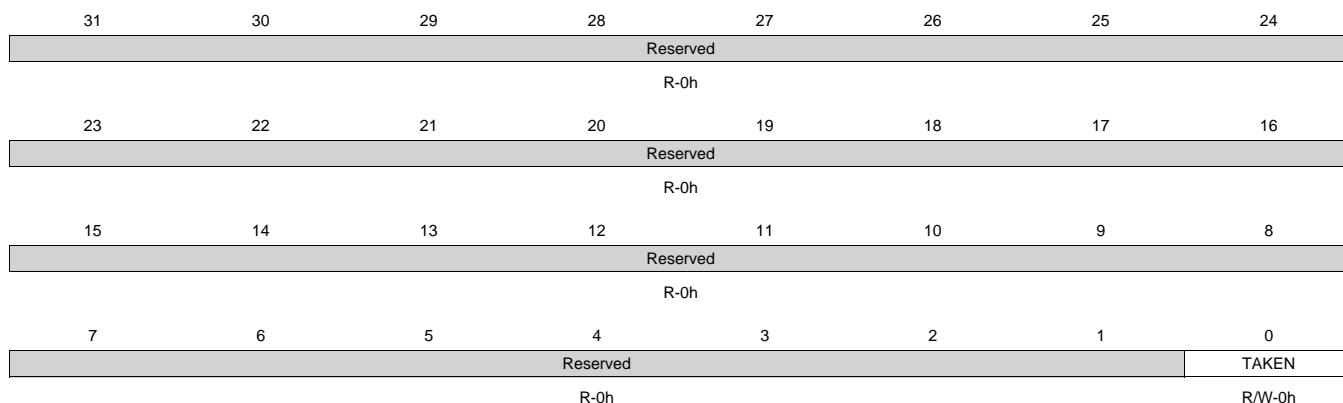
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.14 LOCK\_REG\_10 Register (offset = 828h) [reset = 0h]

LOCK\_REG\_10 is shown in [Figure 17-58](#) and described in [Table 17-71](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-58. LOCK\_REG\_10 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-71. LOCK\_REG\_10 Register Field Descriptions**

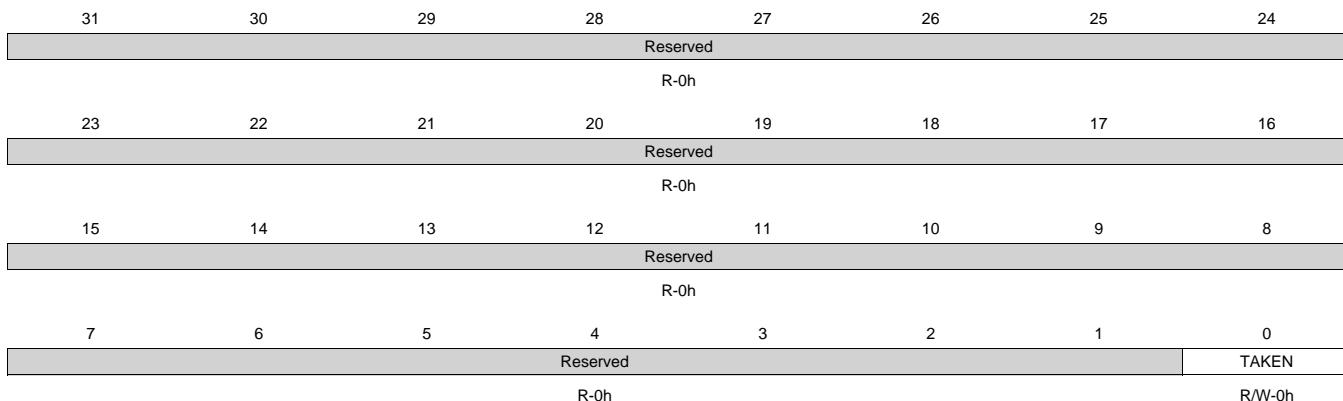
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.15 LOCK\_REG\_11 Register (offset = 82Ch) [reset = 0h]

LOCK\_REG\_11 is shown in [Figure 17-59](#) and described in [Table 17-72](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-59. LOCK\_REG\_11 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-72. LOCK\_REG\_11 Register Field Descriptions**

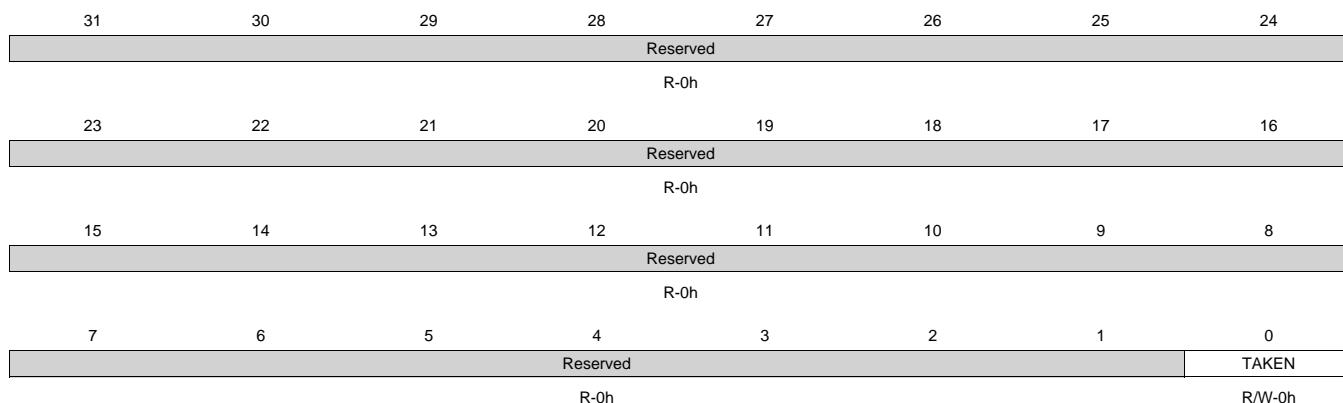
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.16 LOCK\_REG\_12 Register (offset = 830h) [reset = 0h]

LOCK\_REG\_12 is shown in [Figure 17-60](#) and described in [Table 17-73](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-60. LOCK\_REG\_12 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-73. LOCK\_REG\_12 Register Field Descriptions**

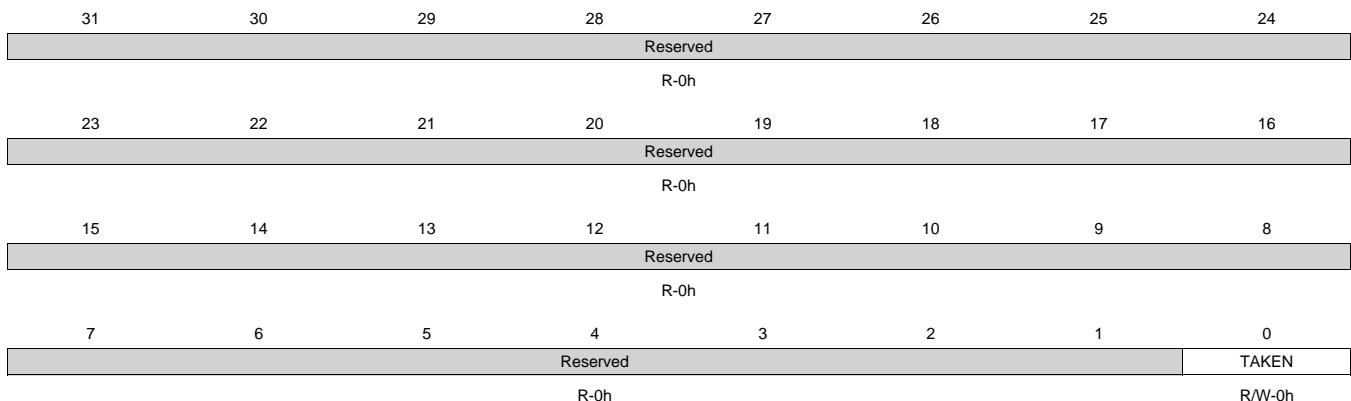
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.17 LOCK\_REG\_13 Register (offset = 834h) [reset = 0h]

LOCK\_REG\_13 is shown in [Figure 17-61](#) and described in [Table 17-74](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-61. LOCK\_REG\_13 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-74. LOCK\_REG\_13 Register Field Descriptions**

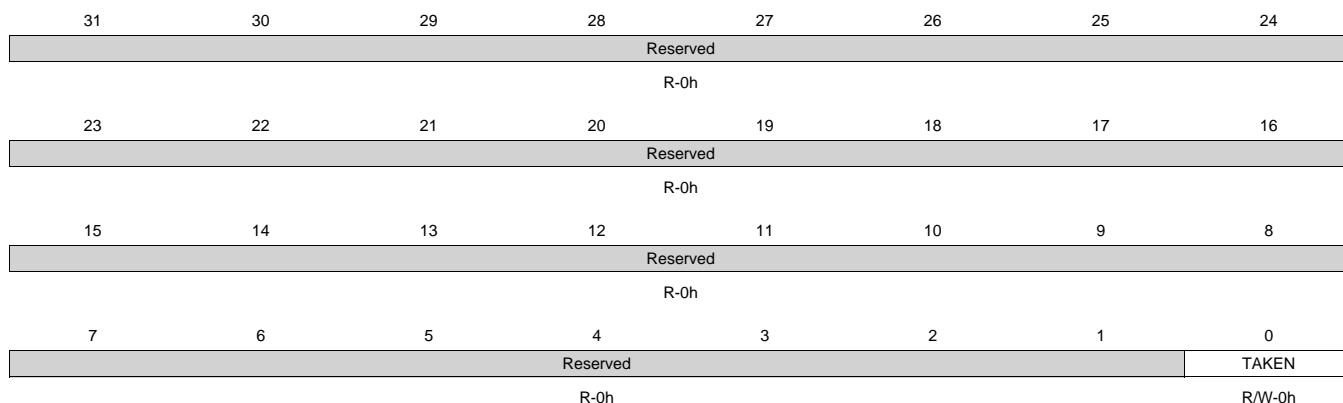
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.18 LOCK\_REG\_14 Register (offset = 838h) [reset = 0h]

LOCK\_REG\_14 is shown in [Figure 17-62](#) and described in [Table 17-75](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-62. LOCK\_REG\_14 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-75. LOCK\_REG\_14 Register Field Descriptions**

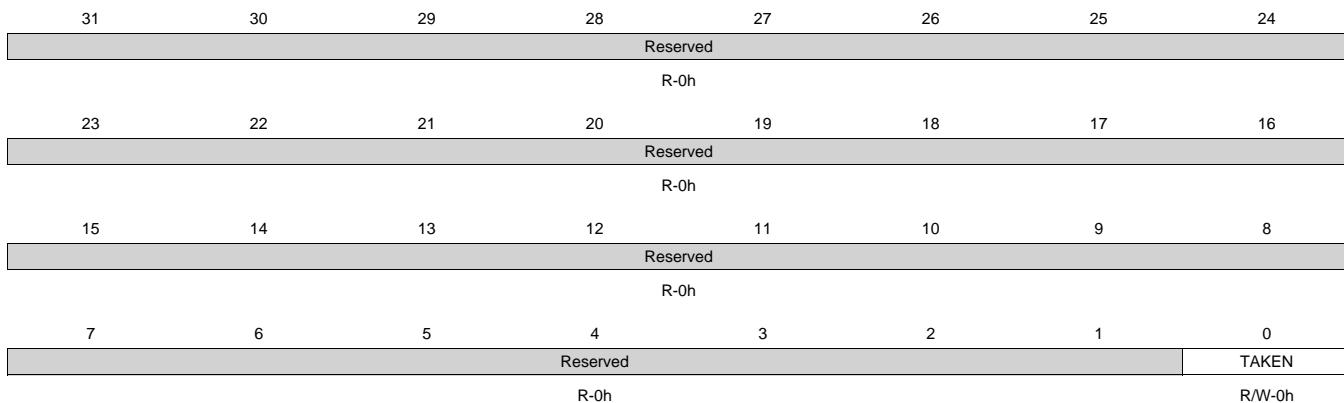
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.19 LOCK\_REG\_15 Register (offset = 83Ch) [reset = 0h]

LOCK\_REG\_15 is shown in [Figure 17-63](#) and described in [Table 17-76](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-63. LOCK\_REG\_15 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-76. LOCK\_REG\_15 Register Field Descriptions**

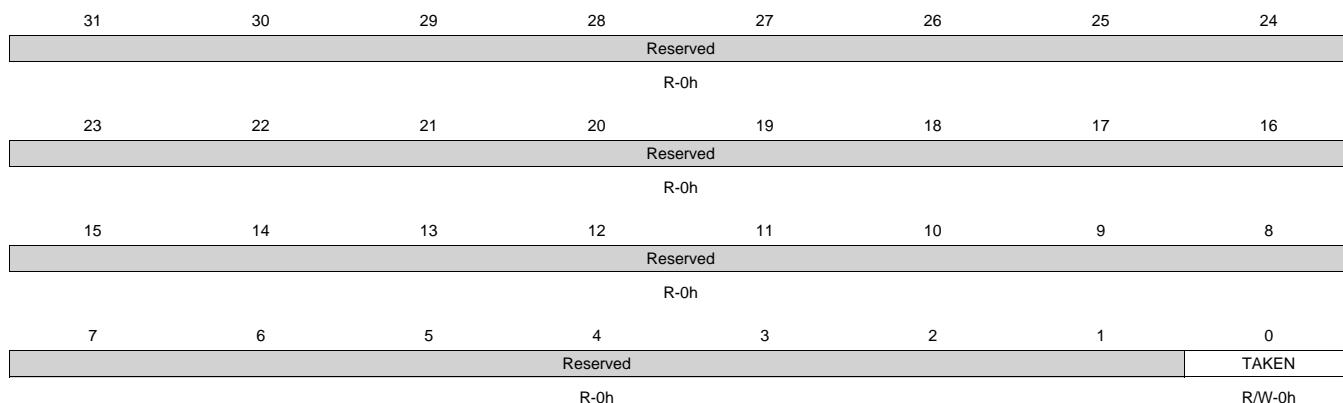
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.20 LOCK\_REG\_16 Register (offset = 840h) [reset = 0h]

LOCK\_REG\_16 is shown in [Figure 17-64](#) and described in [Table 17-77](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-64. LOCK\_REG\_16 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-77. LOCK\_REG\_16 Register Field Descriptions**

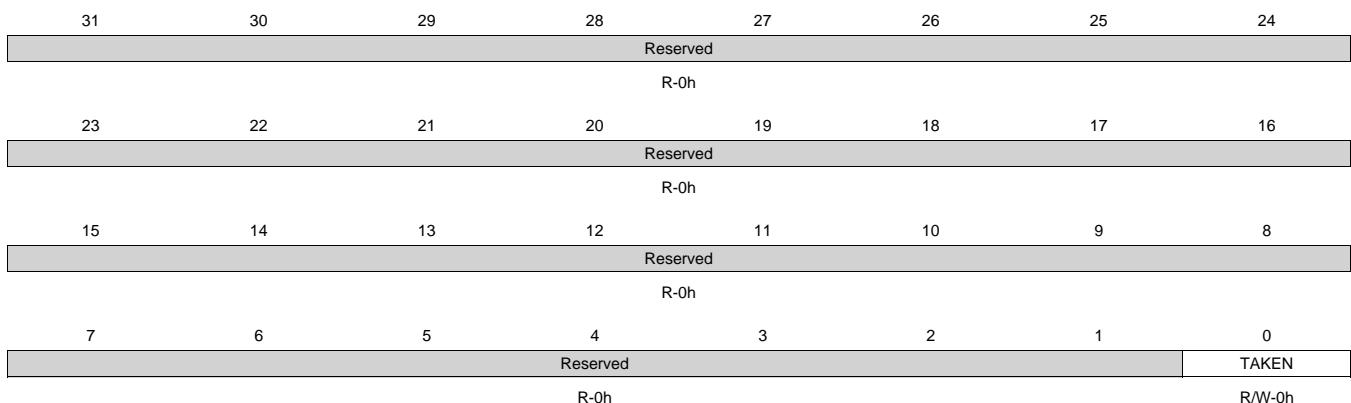
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.21 LOCK\_REG\_17 Register (offset = 844h) [reset = 0h]

LOCK\_REG\_17 is shown in [Figure 17-65](#) and described in [Table 17-78](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-65. LOCK\_REG\_17 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-78. LOCK\_REG\_17 Register Field Descriptions**

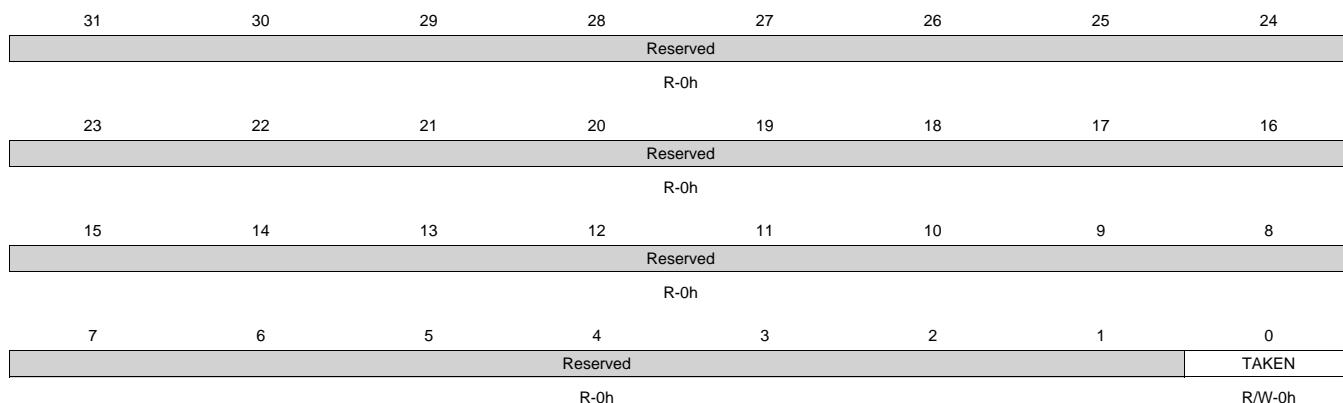
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.22 LOCK\_REG\_18 Register (offset = 848h) [reset = 0h]

LOCK\_REG\_18 is shown in [Figure 17-66](#) and described in [Table 17-79](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-66. LOCK\_REG\_18 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-79. LOCK\_REG\_18 Register Field Descriptions**

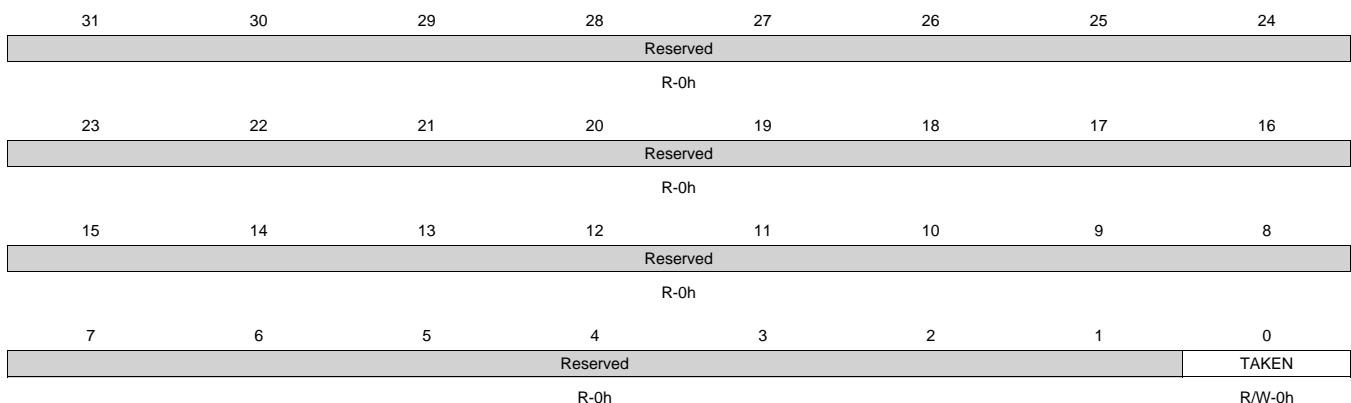
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.23 LOCK\_REG\_19 Register (offset = 84Ch) [reset = 0h]

LOCK\_REG\_19 is shown in [Figure 17-67](#) and described in [Table 17-80](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-67. LOCK\_REG\_19 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-80. LOCK\_REG\_19 Register Field Descriptions**

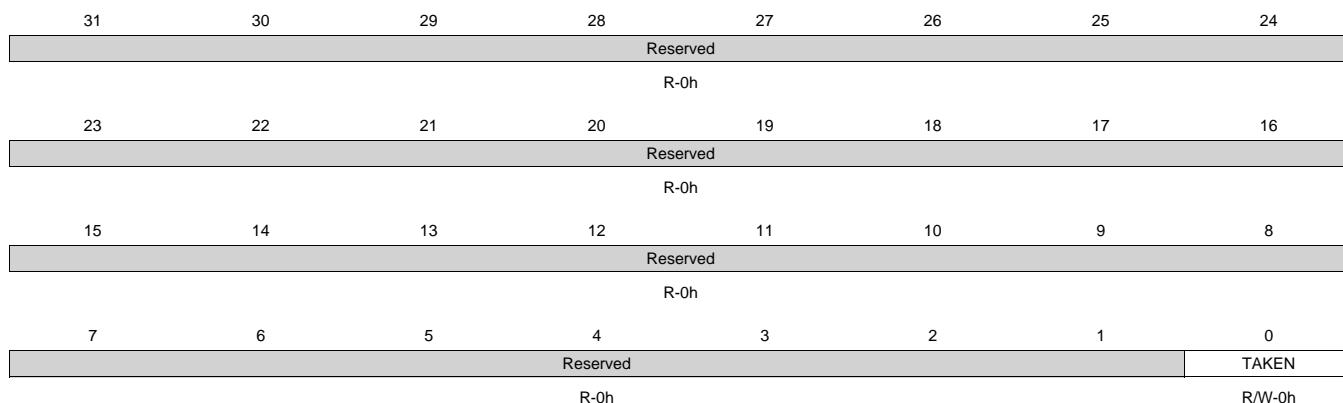
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.24 LOCK\_REG\_20 Register (offset = 850h) [reset = 0h]

LOCK\_REG\_20 is shown in [Figure 17-68](#) and described in [Table 17-81](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-68. LOCK\_REG\_20 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-81. LOCK\_REG\_20 Register Field Descriptions**

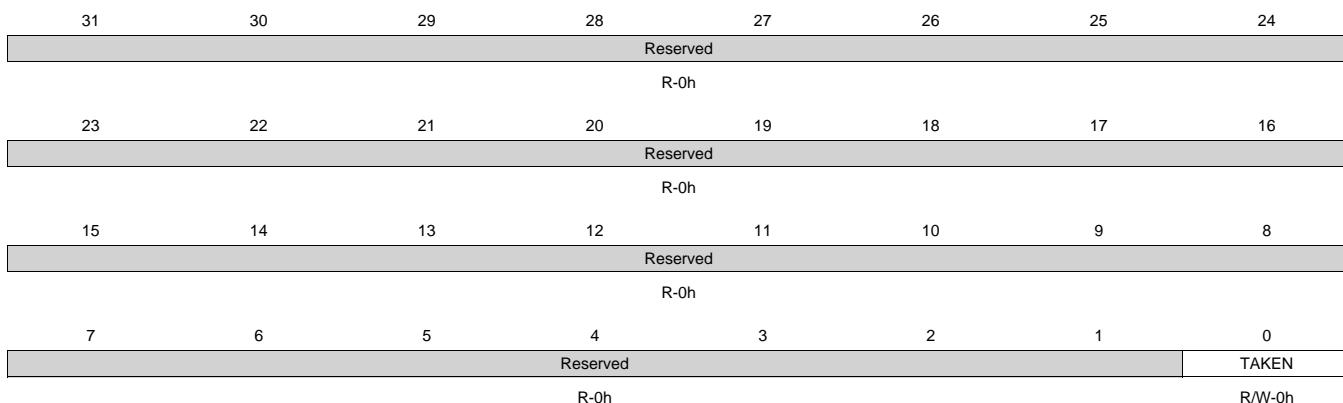
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.25 LOCK\_REG\_21 Register (offset = 854h) [reset = 0h]

LOCK\_REG\_21 is shown in [Figure 17-69](#) and described in [Table 17-82](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-69. LOCK\_REG\_21 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-82. LOCK\_REG\_21 Register Field Descriptions**

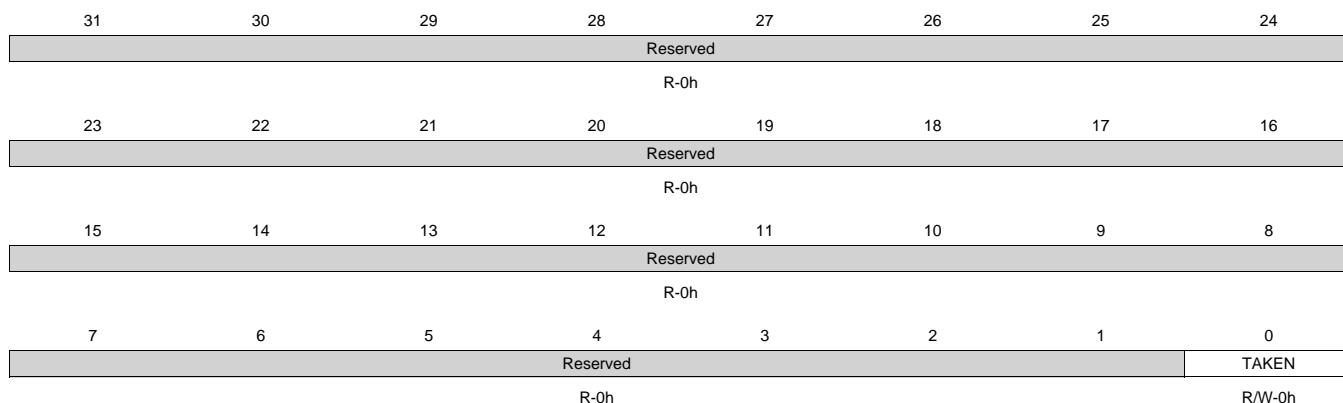
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.26 LOCK\_REG\_22 Register (offset = 858h) [reset = 0h]

LOCK\_REG\_22 is shown in [Figure 17-70](#) and described in [Table 17-83](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-70. LOCK\_REG\_22 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-83. LOCK\_REG\_22 Register Field Descriptions**

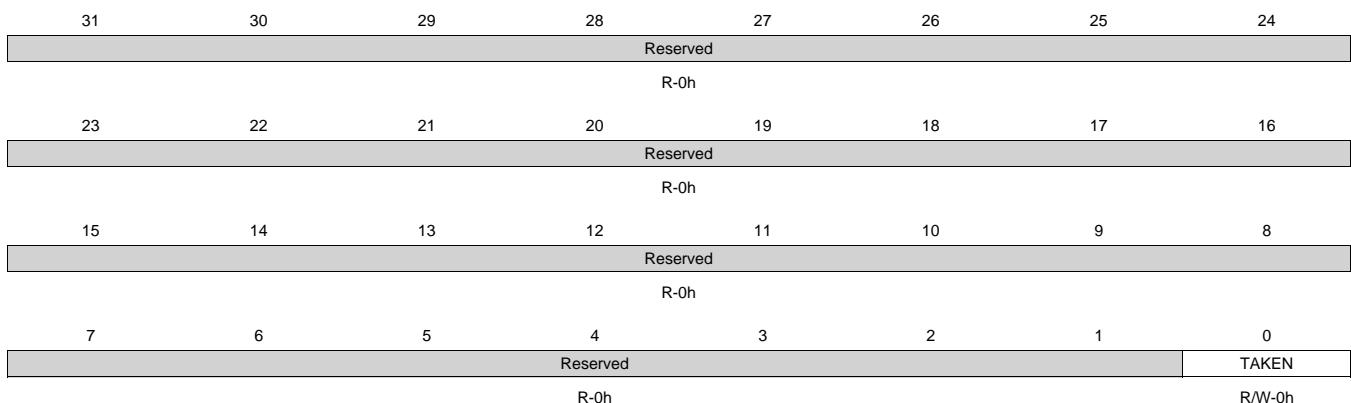
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.27 LOCK\_REG\_23 Register (offset = 85Ch) [reset = 0h]

LOCK\_REG\_23 is shown in [Figure 17-71](#) and described in [Table 17-84](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-71. LOCK\_REG\_23 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-84. LOCK\_REG\_23 Register Field Descriptions**

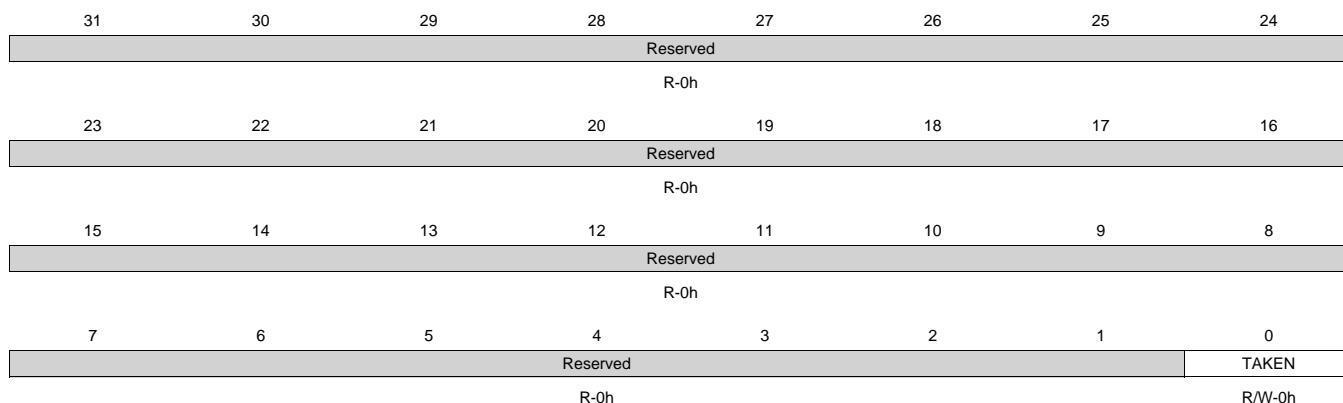
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.28 LOCK\_REG\_24 Register (offset = 860h) [reset = 0h]

LOCK\_REG\_24 is shown in [Figure 17-72](#) and described in [Table 17-85](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-72. LOCK\_REG\_24 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-85. LOCK\_REG\_24 Register Field Descriptions**

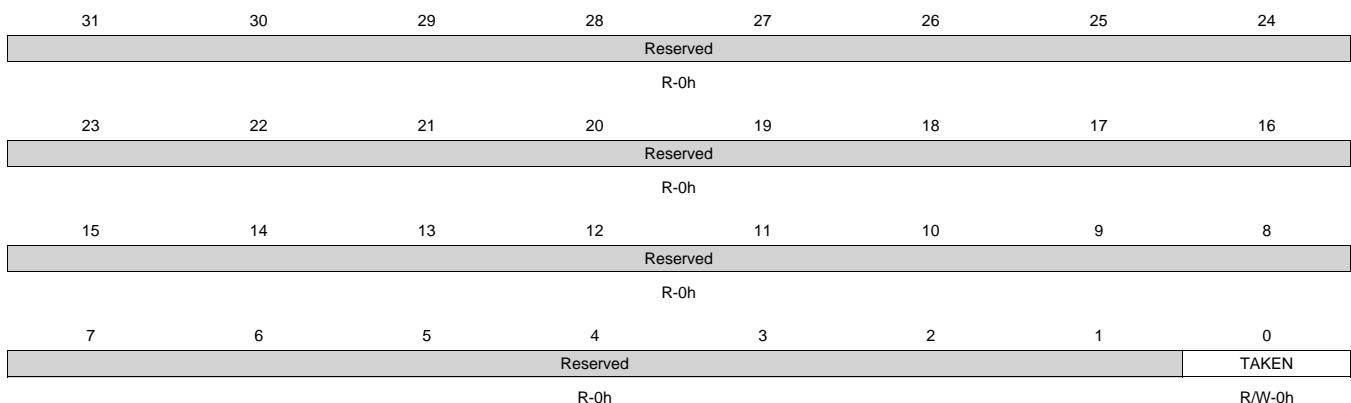
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.29 LOCK\_REG\_25 Register (offset = 864h) [reset = 0h]

LOCK\_REG\_25 is shown in [Figure 17-73](#) and described in [Table 17-86](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-73. LOCK\_REG\_25 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-86. LOCK\_REG\_25 Register Field Descriptions**

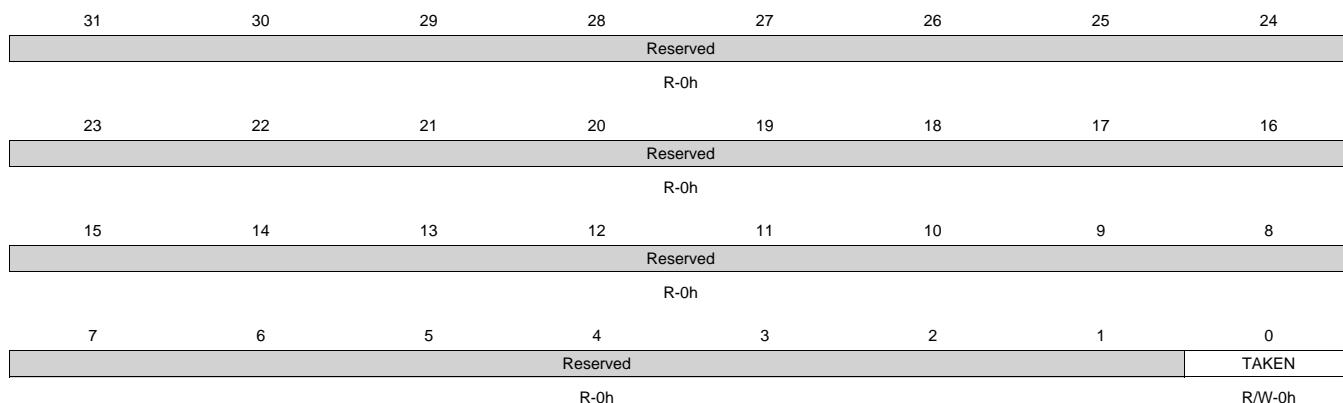
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.30 LOCK\_REG\_26 Register (offset = 868h) [reset = 0h]

LOCK\_REG\_26 is shown in [Figure 17-74](#) and described in [Table 17-87](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-74. LOCK\_REG\_26 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-87. LOCK\_REG\_26 Register Field Descriptions**

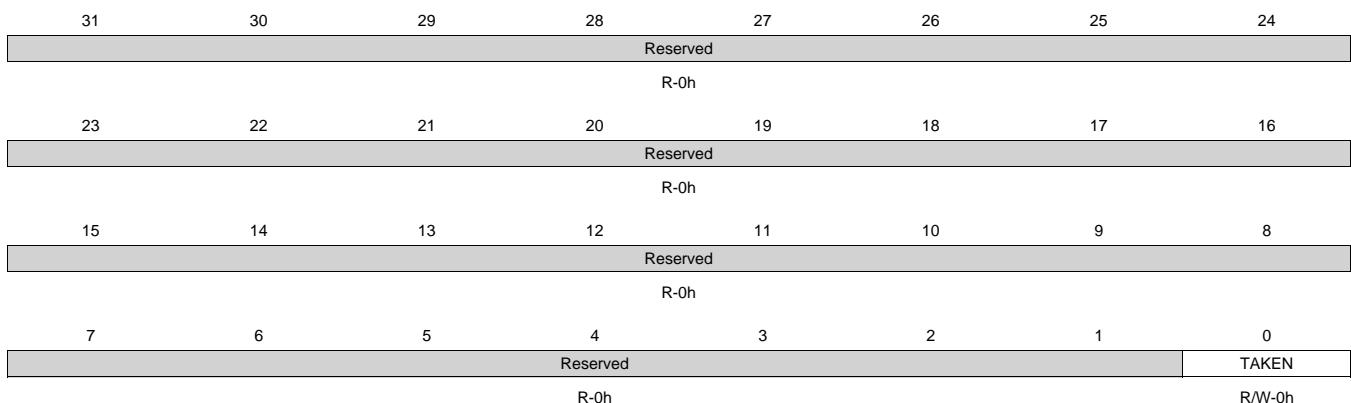
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.31 LOCK\_REG\_27 Register (offset = 86Ch) [reset = 0h]

LOCK\_REG\_27 is shown in [Figure 17-75](#) and described in [Table 17-88](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-75. LOCK\_REG\_27 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-88. LOCK\_REG\_27 Register Field Descriptions**

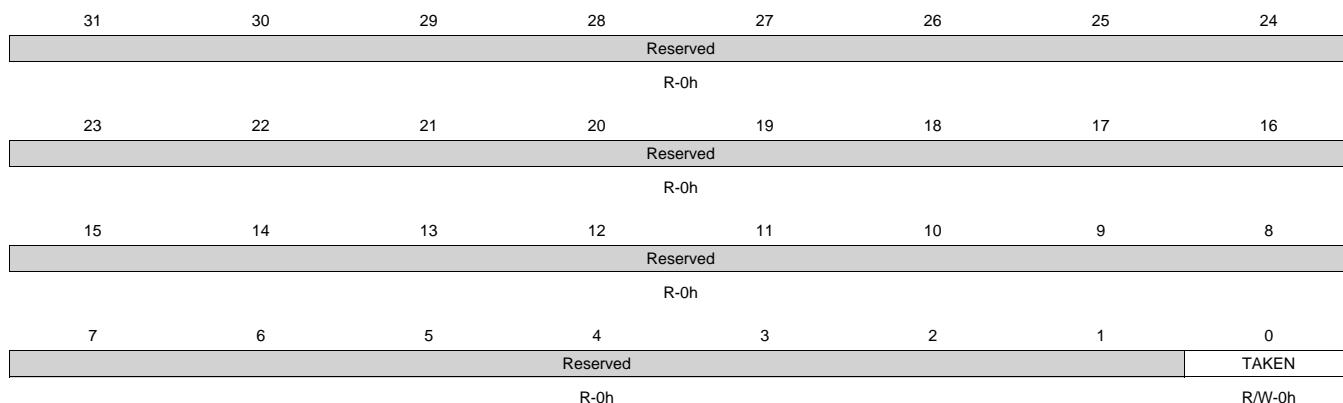
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.32 LOCK\_REG\_28 Register (offset = 870h) [reset = 0h]

LOCK\_REG\_28 is shown in [Figure 17-76](#) and described in [Table 17-89](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-76. LOCK\_REG\_28 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-89. LOCK\_REG\_28 Register Field Descriptions**

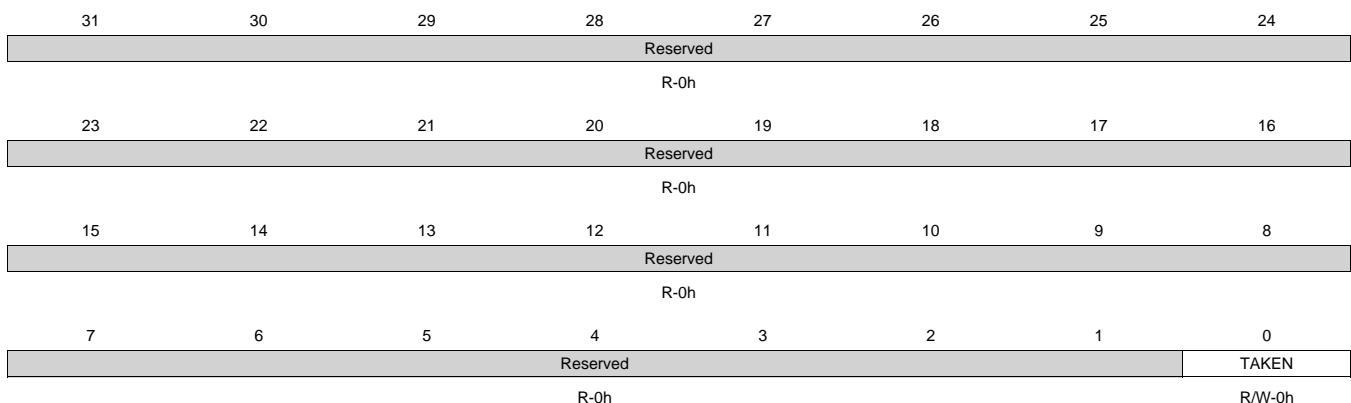
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.33 LOCK\_REG\_29 Register (offset = 874h) [reset = 0h]

LOCK\_REG\_29 is shown in [Figure 17-77](#) and described in [Table 17-90](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-77. LOCK\_REG\_29 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-90. LOCK\_REG\_29 Register Field Descriptions**

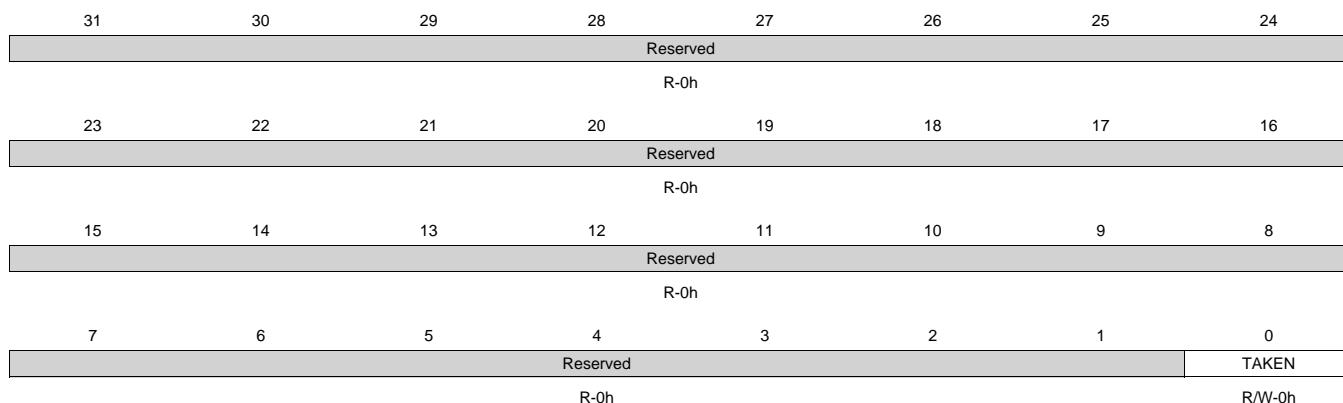
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.34 LOCK\_REG\_30 Register (offset = 878h) [reset = 0h]

LOCK\_REG\_30 is shown in [Figure 17-78](#) and described in [Table 17-91](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-78. LOCK\_REG\_30 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-91. LOCK\_REG\_30 Register Field Descriptions**

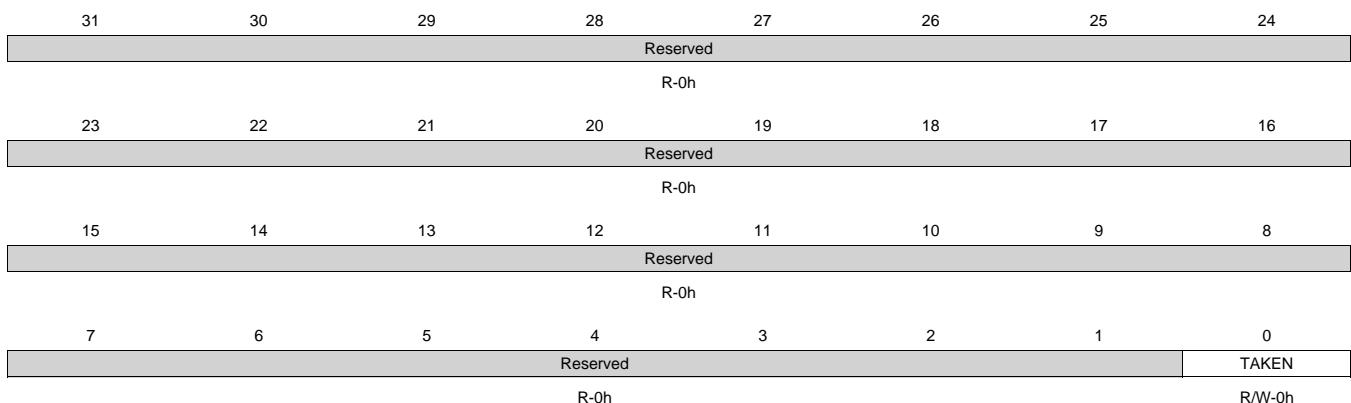
| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

### 17.2.1.35 LOCK\_REG\_31 Register (offset = 87Ch) [reset = 0h]

LOCK\_REG\_31 is shown in [Figure 17-79](#) and described in [Table 17-92](#).

This register is read when attempting to acquire a lock. The lock is automatically taken if it was not taken and the value returned by the read is zero. If the lock was already taken, then the read returns one. Writing a zero to this register frees the lock.

**Figure 17-79. LOCK\_REG\_31 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 17-92. LOCK\_REG\_31 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-1 | Reserved | R    | 0h    |             |
| 0    | TAKEN    | R/W  | 0h    |             |

## **Multimedia Card (MMC)**

This chapter describes the MMC of the device.

| Topic                                   | Page |
|---|------|
| 18.1 Introduction .....                 | 3332 |
| 18.2 Integration .....                  | 3333 |
| 18.3 Functional Description .....       | 3337 |
| 18.4 Low-Level Programming Models ..... | 3371 |
| 18.5 Multimedia Card Registers .....    | 3376 |

## 18.1 Introduction

### 18.1.1 MMCHS Features

The general features of the MMCHS host controller IP are:

- Built-in 1024-byte buffer for read or write
- Two DMA channels, one interrupt line
- Clock support
  - 96-MHz functional clock source input
  - up to 384Mbit/sec (48MByte/sec) in MMC mode 8-bit data transfer
  - up to 192Mbit/sec (24MByte/sec) in High-Speed SD mode 4-bit data transfer
  - up to 24Mbit/sec (3MByte/sec) in Default SD mode 1-bit data transfer
- Support for SDA 3.0 Part A2 programming model
- Serial link supports full compliance with:
  - MMC command/response sets as defined in the MMC standard specification v4.3.
  - SD command/response sets as defined in the SD Physical Layer specification v2.00
  - SDIO command/response sets and interrupt/read-wait suspend-resume operations as defined in the SD part E1 specification v 2.00
  - SD Host Controller Standard Specification sets as defined in the SD card specification Part A2 v2.00

### 18.1.2 Unsupported MMCHS Features

The MMCHS module features not supported in this device are shown in [Table 18-1](#).

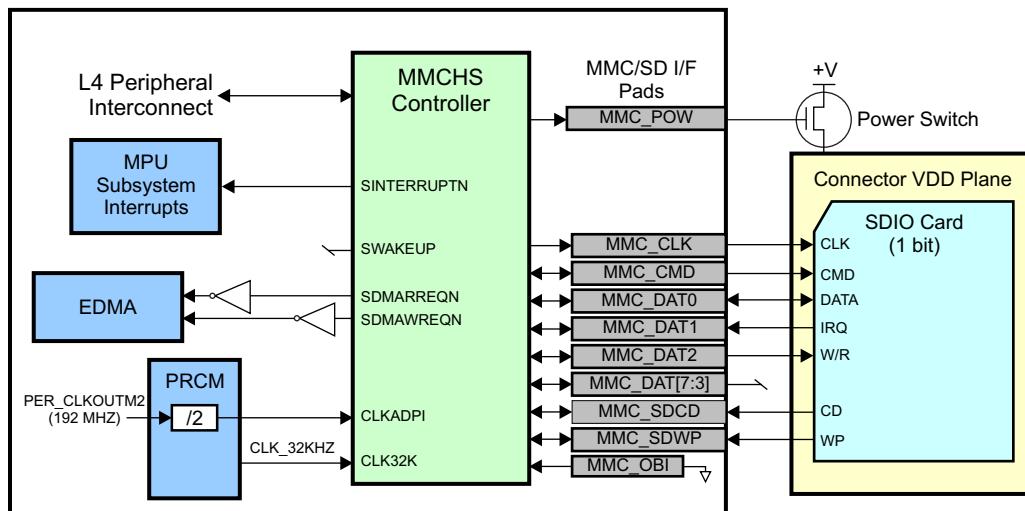
**Table 18-1. Unsupported MMCHS Features**

| Feature                          | Reason                               |
|----------------------------------|--------------------------------------|
| MMC Out-of-band interrupts       | MMC_OBI input tied low               |
| Master DMA operation             | Disabled through synthesis parameter |
| Card Supply Control (MMCHS(1-2)) | Signal not pinned out                |
| Dual Data Rate (DDR) mode        | Timing not supported                 |

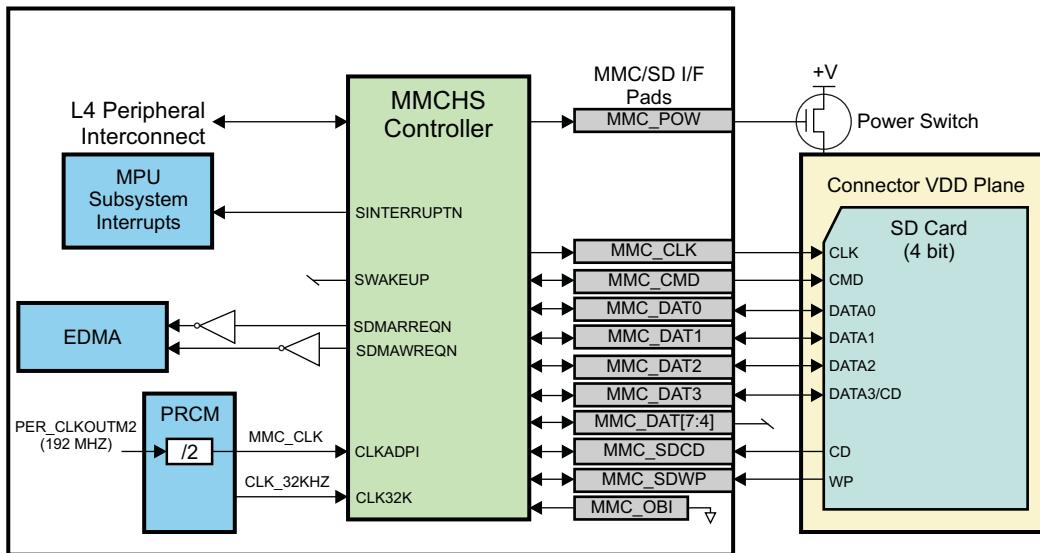
## 18.2 Integration

This device contains three instances of the Multimedia Card (MMC), Secure Digital (SD), and Secure Digital I/O (SDIO) high speed interface module (MMCHS). The controller provides an interface to an MMC, SD memory card or SDIO card.

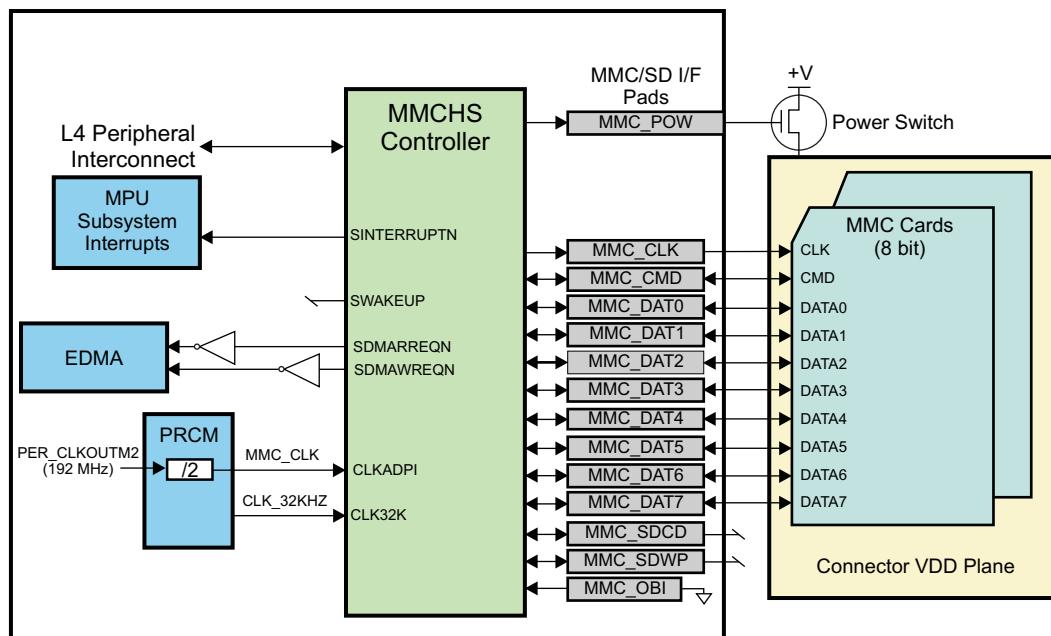
The application interface is responsible for managing transaction semantics; the MMC/SDIO host controller deals with MMC/SDIO protocol at transmission level, packing data, adding CRC, start/end bit and checking for syntactical correctness. [Figure 18-1](#) through [Figure 18-3](#) below show examples of systems using the MMCHS controller. Note that the power switch control is only available on the MMCHS0 interface.



**Figure 18-1. MMCHS Module SDIO Application**



**Figure 18-2. MMCHS SD (4-bit) Card Application**



**Figure 18-3. MMCHS Module MMC Application**

### 18.2.1 MMCHS Connectivity Attributes

The general connectivity attributes for the three MMCHS modules are shown in [Table 18-2](#).

**Table 18-2. MMCHS Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L4LS_GCLK (OCP)<br>PD_PER_MMC_FCLK (Func)<br>CLK_32KHZ (Debounce)                                       |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 1 interrupt per instance to MPU Subsystem (MMCSdXINT)  |
| DMA Requests        | 2 DMA requests per instance to EDMA (SDTXEVTx,<br>SDRXEVTx)<br>(Active low, need to be inverted in glue logic) |
| Physical Address    | L4 Peripheral slave port   |

### 18.2.2 MMCHS Clock and Reset Management

The MMCHS controller has separate bus interface and functional clocks. The debounce clock is created by dividing the 96-MHz (48 MHz @ OPP50) clock in the PRCM by two and then dividing the resulting 48-MHz (24 MHz @ OPP50) clock by a fixed 732.4219 (366.2109 @ OPP50) in the Control Module to get a 32-kHz clock. This clock is fed back into the PRCM for clock gating. (See the CLK32KDIVRATIO\_CTRL register in [Chapter 9, Control Module](#), for more details).

**Table 18-3. MMCHS Clock Signals**

| Clock Signal                | Max Freq   | Reference / Source | Comments                   |
|-----------------------------|------------|--------------------|----------------------------|
| CLK Interface clock         | 100 MHz    | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk from PRCM |
| CLKADPI Functional clock    | 96 MHz     | PER_CLKOUTM2 / 4   | pd_per_mmc_fclk from PRCM  |
| CLK32 Input de-bounce clock | 32.768 KHz | CLK_24 / 732.4219  | clk_32KHz from PRCM        |

### 18.2.3 MMCHS Pin List

The MMCHS interface pins are summarized in [Table 18-4](#).

**Table 18-4. MMCHS Pin List**

| Pin           | Type | Description                                |
|---------------|------|--|
| MMCx_CLK      | I/O  | MMC/SD serial clock output                 |
| MMCx_CMD      | I/O  | MMC/SD command signal                      |
| MMCx_DAT0     | I/O  | MMC/SD data signal                         |
| MMCx_DAT1     | I/O  | MMC/SD data signal, SDIO interrupt input   |
| MMCx_DAT2     | I/O  | MMC/SD data signal, SDIO read wait output  |
| MMCx_DAT[7:3] | I/O  | MMC/SD data signals                        |
| MMCx_POW      | O    | MMC/SD power supply control (MMCHS 0 only) |
| MMCx_SD_CD    | I    | SD card detect (from connector)            |
| MMCx_SD_WP    | I    | SD write protect (from connector)          |
| MMCx_OBI      | I    | MMC out of band interrupt                  |

The direction of the data lines depends on the selected data transfer mode as summarized in [Table 18-5](#).

**Table 18-5. DAT Line Direction for Data Transfer Modes**

|        | MMC/SD 1-bit mode | MMC/SD 4-bit mode | MMC/SD 8-bit mode | SDIO 1-bit mode    | SDIO 4-bit mode         |
|--------|-------------------|-------------------|-------------------|--------------------|-------------------------|
| DAT[0] | I/O               | I/O               | I/O               | I/O                | I/O                     |
| DAT[1] | I <sup>(1)</sup>  | I/O               | I/O               | I <sup>(2)</sup>   | I/O or I <sup>(2)</sup> |
| DAT[2] | I <sup>(1)</sup>  | I/O               | I/O               | I/O <sup>(3)</sup> | I/O or O <sup>(3)</sup> |
| DAT[3] | I <sup>(1)</sup>  | I/O               | I/O               | I <sup>(1)</sup>   | I/O                     |
| DAT[4] | I <sup>(1)</sup>  | I <sup>(1)</sup>  | I/O               | I <sup>(1)</sup>   | I <sup>(1)</sup>        |
| DAT[5] | I <sup>(1)</sup>  | I <sup>(1)</sup>  | I/O               | I <sup>(1)</sup>   | I <sup>(1)</sup>        |
| DAT[6] | I <sup>(1)</sup>  | I <sup>(1)</sup>  | I/O               | I <sup>(1)</sup>   | I <sup>(1)</sup>        |
| DAT[7] | I <sup>(1)</sup>  | I <sup>(1)</sup>  | I/O               | I <sup>(1)</sup>   | I <sup>(1)</sup>        |

<sup>(1)</sup> Hi-Z state to avoid bus conflict.

<sup>(2)</sup> To support incoming interrupt from the SDIO card.

<sup>(3)</sup> To support read wait to the SDIO card. By default it is Input, Output only in read wait period.

The direction of the MMCHS data buffers are controlled by ADPDATDIROQ signals. ADPDATDIROQ[i] = 1 sets the corresponding DAT signal(s) in read position (input) and ADPDATDIROQ[i] = 0 sets the corresponding DAT signal(s) in write position (output). Additionally, the ADPDATDIRLS signals are provided (with opposite polarity) to control the direction of external level shifters. The value of these control signals for the various data modes are summarized in [Table 18-6](#).

**Table 18-6. ADPDATDIROQ and ADPDATDIRLS Signal States**

|        | <b>MMC/SD<br/>1-bit mode</b>                     | <b>MMC/SD<br/>4-bit mode</b>                     | <b>MMC/SD<br/>8-bit mode</b>                     | <b>SDIO<br/>1-bit mode</b>                       | <b>SDIO<br/>4-bit mode</b>                       |
|--------|--|--|--|--|--|
| DAT[0] | ADPDATDIRLS[0] = 0 / 1<br>ADPDATDIROQ[0] = 1 / 0 | ADPDATDIRLS[0] = 0 / 1<br>ADPDATDIROQ[0] = 1 / 0 | ADPDATDIRLS[0] = 0 / 1<br>ADPDATDIROQ[0] = 1 / 0 | ADPDATDIRLS[0] = 0 / 1                           | ADPDATDIRLS[0] = 0 / 1<br>ADPDATDIROQ[0] = 1 / 0 |
| DAT[2] | ADPDATDIRLS[2] = 0<br>ADPDATDIROQ[2] = 1         | ADPDATDIRLS[2] = 0 / 1<br>ADPDATDIROQ[2] = 1 / 0 | ADPDATDIRLS[2] = 0 / 1<br>ADPDATDIROQ[2] = 1 / 0 | ADPDATDIRLS[2] = 0 / 1<br>ADPDATDIROQ[2] = 1 / 0 | ADPDATDIRLS[2] = 0 / 1<br>ADPDATDIROQ[2] = 1 / 0 |
| DAT[1] | ADPDATDIRLS[1] = 0<br>ADPDATDIROQ[1] = 1         | ADPDATDIRLS[1] = 0 / 1<br>ADPDATDIROQ[1] = 1 / 0 | ADPDATDIRLS[1] = 0 / 1<br>ADPDATDIROQ[1] = 1 / 0 | ADPDATDIRLS[1] = 0 / 1<br>ADPDATDIROQ[1] = 1     | ADPDATDIRLS[1] = 0 / 1<br>ADPDATDIROQ[1] = 1 / 0 |
| DAT[3] |  | ADPDATDIRLS[3] = 0<br>ADPDATDIROQ[3] = 1         |
| DAT[4] |  |  |  |  |  |
| DAT[5] |  |  |  |  |  |
| DAT[6] |  |  |  |  |  |
| DAT[7] |  |  |  |  |  |

ADPDATIRLSx = 0 for input and 1 for output — these signals are not pinned out on this device.

ADPDATIROQx = 1 for output and 1 for input.

Grayed cells indicate that the data line is not used in the selected transfer mode.

## 18.3 Functional Description

One MMC/SD/SDIO host controller can support one MMC memory card, one SD card, or one SDIO card.

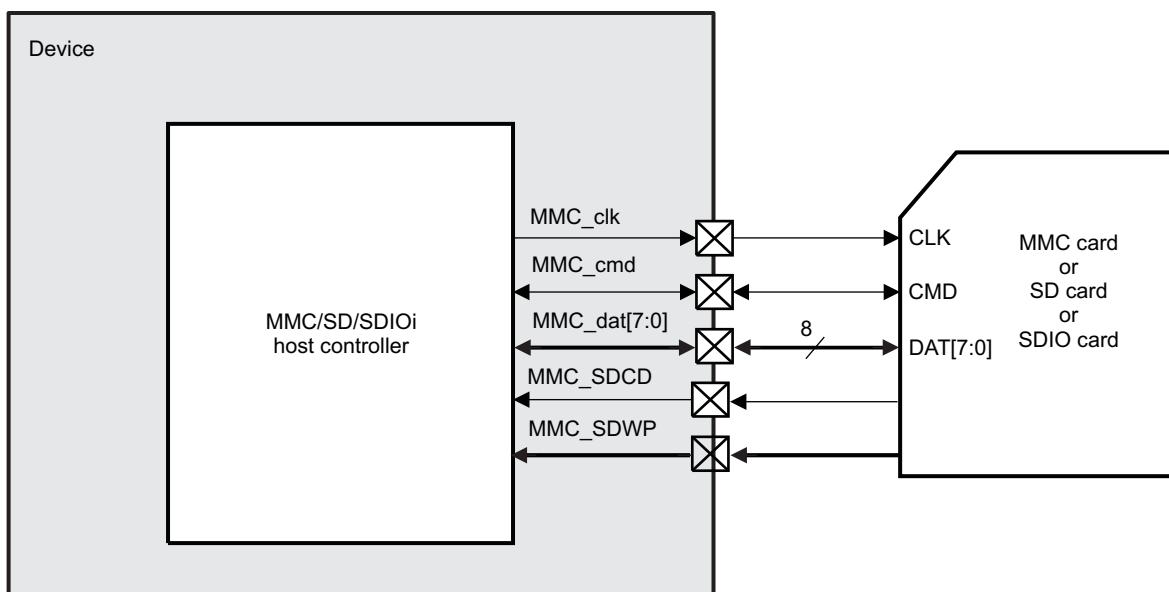
Other combinations (for example, two SD cards, one MMC card, and one SD card) are not supported through a single controller.

### 18.3.1 MMC/SD/SDIO Functional Modes

#### 18.3.1.1 MMC/SD/SDIO Connected to an MMC, an SD Card, or an SDIO Card

Figure 18-4 shows the MMC/SD/SDIO1 and MMC/SD/SDIO2 host controllers connected to an MMC, an SD, or an SDIO card and its related external connections.

**Figure 18-4. MMC/SD1/2 Connectivity to an MMC/SD Card**



**Figure 18-5. MMC/SD0 Connectivity to an MMC/SD Card**

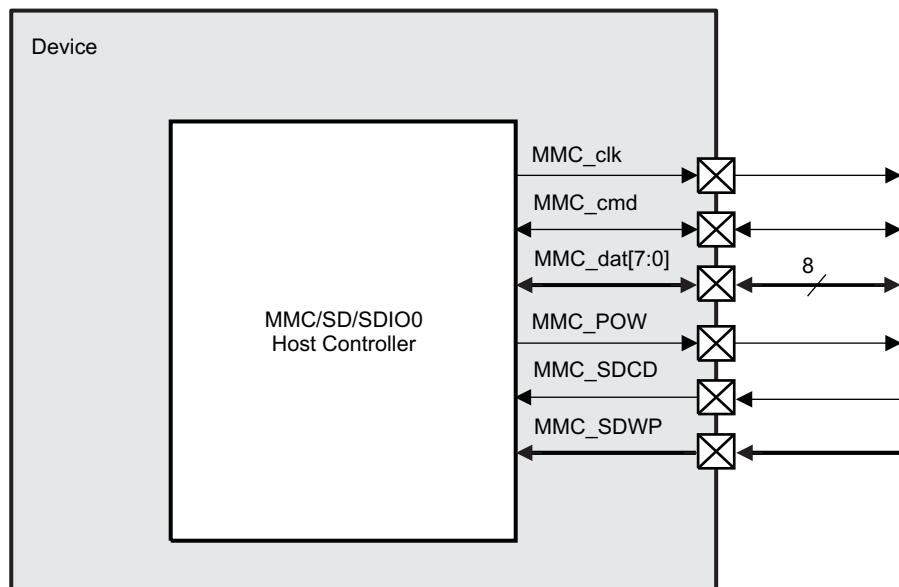


Figure 18-5 shows the MMC/SD/SDIO0 host controller connected to an MMC, SD, or SDIO card and its related external connections. Note that MMC/SD/SDIO0 uses the same signals as MMC/SD/SDIO1 and 2 but adds MMC\_POW.

The following MMC/SD/SDIO controller pins are used

- **MMC\_CMD** This pin is used for two-way communication between the connected card and the MMC/SD/SDIO controller. The MMC/SD/SDIO controller transmits commands to the card and the memory card drives responses to the commands on this pin.
- **MMC\_DAT7-0** Depending on which type of card you are using, you may need to connect 1, 4, or 8 data lines. The number of DAT pins (the data bus width) is set by the Data Transfer Width (DTW) bit in the MMC control register (SD\_HCTL). For more information, see [Section 18.5.1, MULTIMEDIA\\_CARD Registers](#).
- **MMC\_CLK** This pin provides the clock to the memory card from the MMC/SD controller.
- **MMC\_POW** Used for MMC/SD card's cards on/off power supply control. When high, denotes power-on condition.
- **MMC\_SDCD** This input pin serves as the MMC/SD/SDIO carrier detect. This signal is received from a mechanical switch on the slot.
- **MMC\_SDWP** This input pin is used for the SD/SDIO card's write protect. This signal is received from a mechanical protect switch on the slot (system dependant). Applicable only for SD and SDIO cards that have a mechanical sliding tablet on the side of the card.

**Note:** The MMC\_CLK pin functions as an output but must be configured as an I/O to internally loopback the clock to time the inputs.

[Table 18-7](#) provides a summary of these pins.

**Table 18-7. MMC/SD/SDIO Controller Pins and Descriptions**

| Pin                    | Type | 1-Bit Mode   | 4-Bit Mode   | 8-Bit Mode   | Reset Value    |
|------------------------|------|--------------|--------------|--------------|----------------|
| MMC_CLK <sup>(1)</sup> | O    | Clock Line   | Clock Line   | Clock Line   | High impedance |
| MMC_CMD                | I/O  | Command Line | Command Line | Command Line | High impedance |
| MMC_DAT0               | I/O  | Data Line 0  | Data Line 0  | Data Line 0  | 0              |
| MMC_DAT1               | I/O  | (not used)   | Data Line 1  | Data Line 1  | 0              |
| MMC_DAT2               | I/O  | (not used)   | Data Line 2  | Data Line 2  | 0              |
| MMC_DAT3               | I/O  | (not used)   | Data Line 3  | Data Line 3  | 0              |
| MMC_DAT4               | I/O  | (not used)   | (not used)   | Data Line 4  | 0              |
| MMC_DAT5               | I/O  | (not used)   | (not used)   | Data Line 5  | 0              |
| MMC_DAT6               | I/O  | (not used)   | (not used)   | Data Line 6  | 0              |
| MMC_DAT7               | I/O  | (not used)   | (not used)   | Data Line 7  | 0              |

<sup>(1)</sup> The MMC\_CLK pin functions as an output but must be configured as an I/O to internally loopback the clock to time the inputs.

### 18.3.1.2 Protocol and Data Format

The bus protocol between the MMC/SD/SDIO host controller and the card is message-based. Each message is represented by one of the following parts:

**Command:** A command starts an operation. The command is transferred serially from the \MMC/SD/SDIO host controller to the card on the mmc\_cmd line.

**Response:** A response is an answer to a command. The response is sent from the card to the MMC/SD/SDIO host controller. It is transferred serially on the mmc\_cmd line.

**Data:** Data are transferred from the MMC/SD/SDIO host controller to the card or from a card to the MMC/SD/SDIO host controller using the DATA lines.

**Busy:** The mmc\_dat0 signal is maintained low by the card as far as it is programming the data received.

**CRC status:** CRC result is sent by the card through the mmc\_dat0 line when executing a write transfer. In the case of transmission error, occurring on any of the active data lines, the card sends a negative CRC status on mmc\_dat0. In the case of successful transmission, over all active data lines, the card sends a positive CRC status on mmc\_dat0 and starts the data programming procedure.

### 18.3.1.2.1 Protocol

There are two types of data transfer:

- Sequential operation
- Block-oriented operation

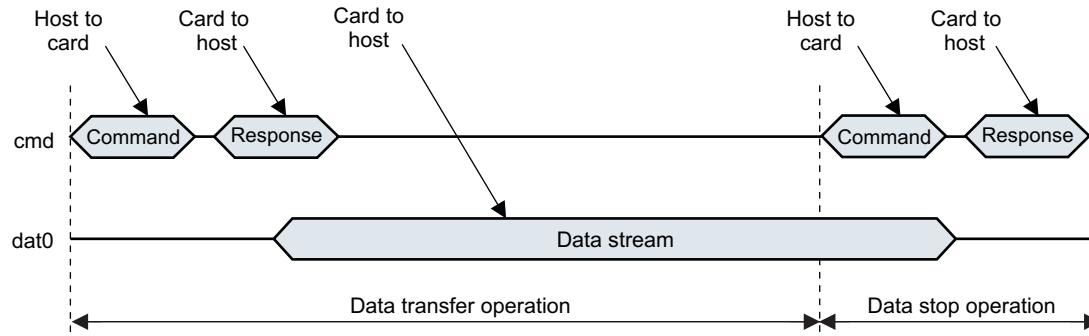
There are specific commands for each type of operation (sequential or block-oriented). See the *Multimedia Card System Specification*, the *SD Memory Card Specifications*, and the *SDIO Card Specification, Part E1* for details about commands and programming sequences supported by the MMC, SD, and SDIO cards.

**CAUTION**

Stream commands are supported only by MMC cards.

Figure 18-6 and Figure 18-7 show how sequential operations are defined. Sequential operation is only for 1-bit transfer and initiates a continuous data stream. The transfer terminates when a stop command follows on the mmc\_cmd line.

**Figure 18-6. Sequential Read Operation (MMC Cards Only)**



**Figure 18-7. Sequential Write Operation (MMC Cards Only)**

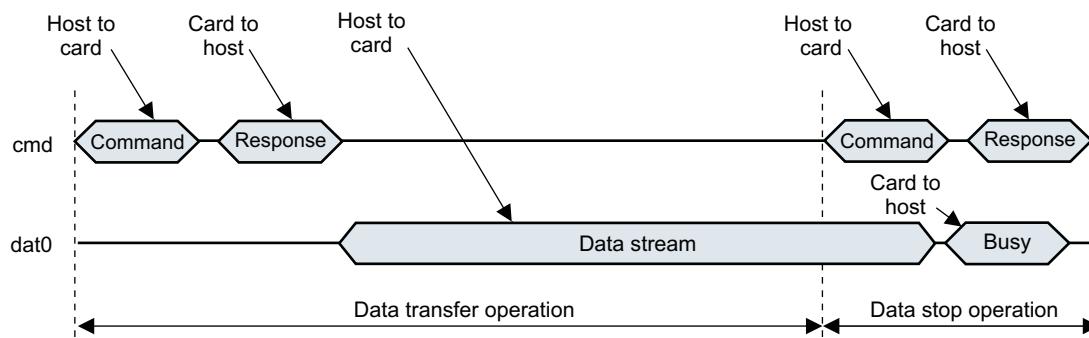
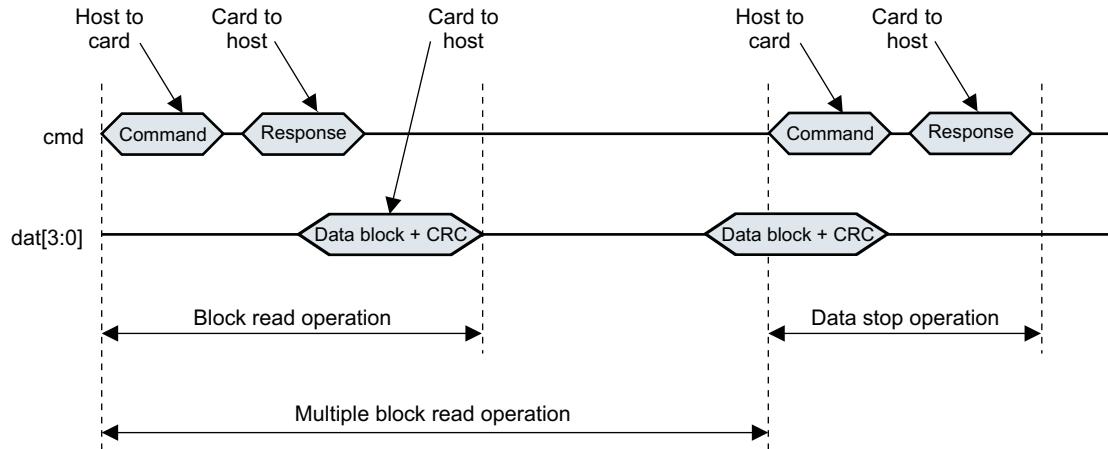
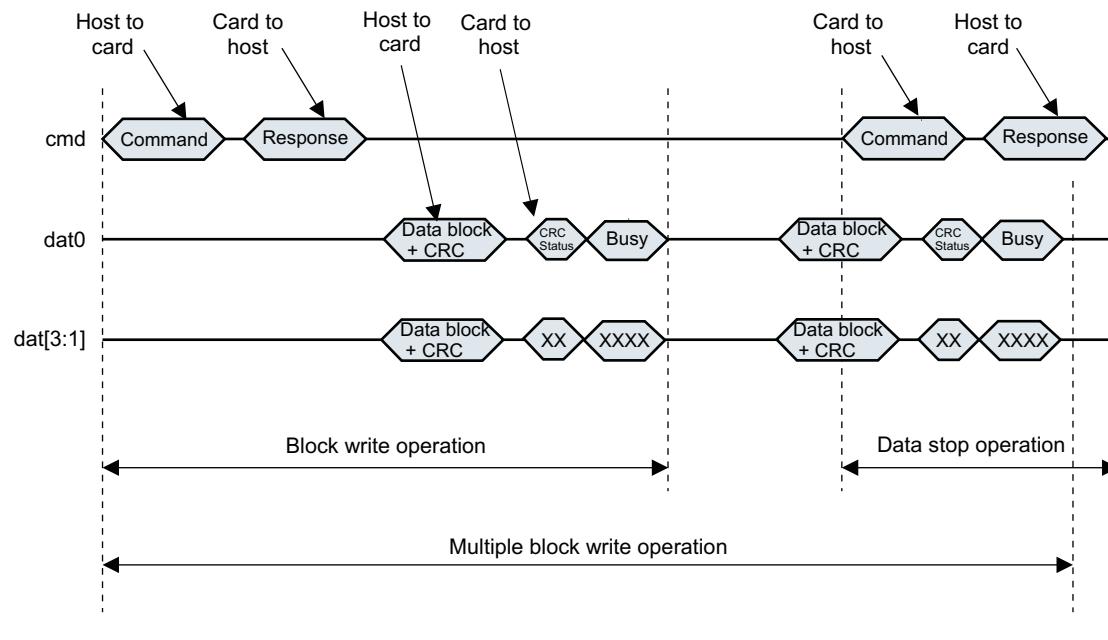


Figure 18-8 and Figure 18-9 show how multiple block-oriented operations are defined. A multiple block-oriented operation sends a data block plus CRC bits. The transfer terminates when a stop command follows on the mmc\_cmd line. These operations are available for all kinds of cards.

**Figure 18-8. Multiple Block Read Operation (MMC Cards Only)**



**Figure 18-9. Multiple Block Write Operation (MMC Cards Only)**



**NOTE:**

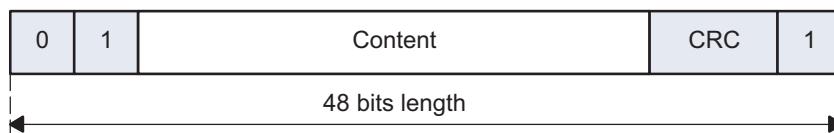
1. The card busy signal is not always generated by the card; the previous examples show a particular case.
2. It is the software's responsibility to do a software reset after a data timeout to ensure that mmc\_clk is stopped. The software reset is done by setting bit 26 in the SD\_SYSCTL register to 1.
3. For multiblock transfer, and especially for MMC cards, you can abort a transfer without using a stop command. Use a CMD23 before a data transfer to define the number of blocks that will be transferred, then the transfer stops automatically after the last block (provided the MMC card supports this feature).

### 18.3.1.2.2 Data Format

#### Coding Scheme for Command Token

Command packets always start with 0 and end with 1. The second bit is a transmitter bit1 for a host command. The content is the command index (coded by 6 bits) and an argument (for example, an address), coded by 32 bits. The content is protected by 7-bit CRC checksum (see [Figure 18-10](#)).

**Figure 18-10. Command Token Format**



#### Coding Scheme for Response Token

Response packets always start with 0 and end with a 1. The second bit is a transmitter bit0 for a card response. The content is different for each type of response (R1, R2, R3, R4, R5, and R6) and the content is protected by 7-bit CRC checksum. Depending on the type of commands sent to the card, the SD\_CMD register must be configured differently to avoid false CRC or index errors to be flagged on command response (see [Table 18-8](#)). For more details about response types, see the *Multimedia Card System Specification*, the *SD Memory Card Specification*, or the *SDIO Card Specification*.

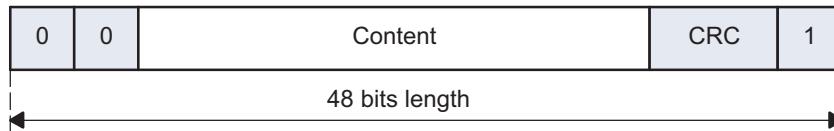
**Table 18-8. Response Type Summary<sup>(1)</sup>**

| Response Type<br>SD_CMD[17:16]<br>RSP_TYPE | Index Check Enable<br>SD_CMD[20]<br>CICE | CRC Check Enable<br>SD_CMD[19]<br>CCCE | Name of Response Type        |
|--|--|--|------------------------------|
| 00   | 0  | 0                                      | No Response                  |
| 01   | 0  | 1                                      | R2                           |
| 10   | 0  | 0                                      | R3 (R4 for SD cards)         |
| 10   | 1  | 1                                      | R1, R6, R5 (R7 for SD cards) |
| 11   | 1  | 1                                      | R1b, R5b                     |

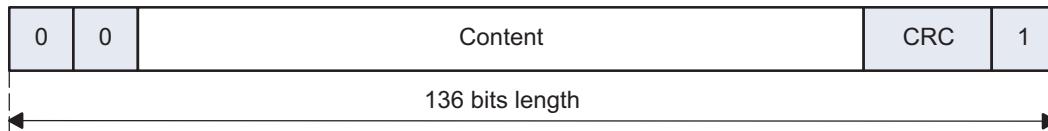
<sup>(1)</sup> The MMC/SD/SDIO host controller assumes that both clocks may be switched off, whatever the value set in the SD\_SYSConfig[9:8] CLOCKACTIVITY bit.

[Figure 18-11](#) and [Figure 18-12](#) depict the 48-bit and 136-bit response packets.

**Figure 18-11. 48-Bit Response Packet (R1, R3, R4, R5, R6)**



**Figure 18-12. 136-Bit Response Packet (R2)**



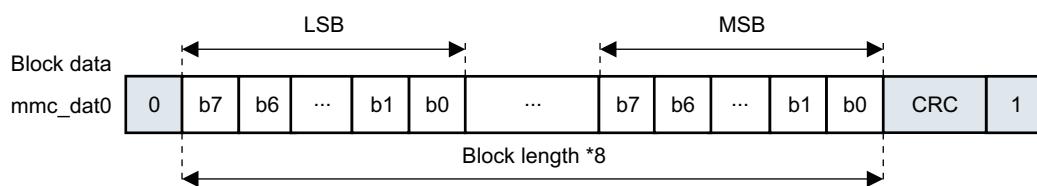
### Coding Scheme for Data Token

Data tokens always start with 0 and end with 1 (see [Figure 18-13](#), [Figure 18-14](#), [Figure 18-15](#), and [Figure 18-16](#)).

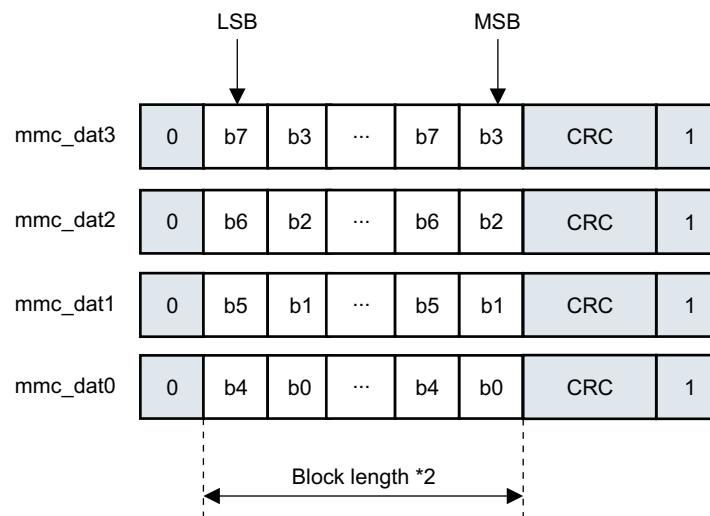
**Figure 18-13. Data Packet for Sequential Transfer (1-Bit)**



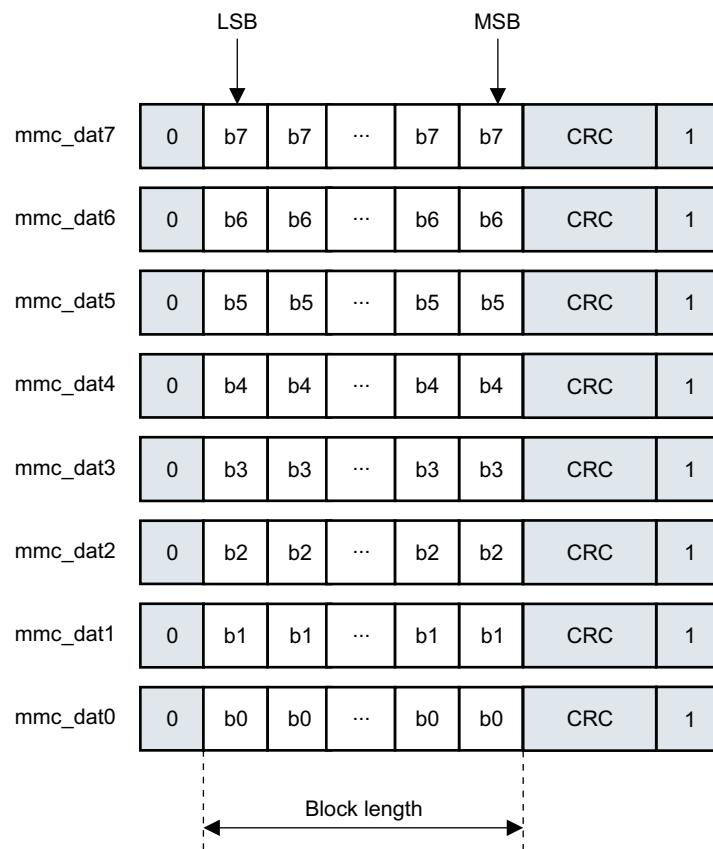
**Figure 18-14. Data Packet for Block Transfer (1-Bit)**



**Figure 18-15. Data Packet for Block Transfer (4-Bit)**



**Figure 18-16. Data Packet for Block Transfer (8-Bit)**



### **18.3.2 Resets**

### **18.3.2.1 Hardware Reset**

The module is reinitialized by the hardware.

The SD\_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a hardware reset.

This hardware reset signal has a global reset action on the module. All configuration registers and all state machines are reset in all clock domains.

This hardware reset signal has a global reset action on the module. All configuration registers and all state-machines are reset in all clock domains.

### **18.3.2.2 Software Reset**

The module is reinitialized by software through the SD\_SYSConfig[1] SOFTRESET bit. This bit has the same action on the module logic as the hardware signal except for:

- Debounce logic
  - SD\_PSTATE, SD\_CAPA, and SD\_CUR\_CAPA registers (see corresponding register descriptions)

The SOFTRESET bit is active high. The bit is automatically reinitialized to 0 by the hardware. The SD\_SYSCTL[24] SRA bit has the same action as the SOFTRESET bit on the design.

The SD\_SYSSTATUS[0] RESETDONE bit can be monitored by the software to check if the module is ready-to-use after a software reset.

Moreover, two partial software reset bits are provided:

- SD\_SYSCTL[26] SRD bit
- SD\_SYSCTL[25] SRC bit

These two reset bits are useful to reinitialize data or command processes respectively in case of line conflict. When set to 1, a reset process is automatically released when the reset completes:

- The SD\_SYSCTL[26] SRD bit resets all finite state-machines and status management that handle data transfers on both the interface and functional side.
- The SD\_SYSCTL[25] SRC bit resets all finite state-machines and status management that handle command transfers on both the interface and functional side.

---

**NOTE:** If **any** of the clock inputs are not present for the MMC/SD/SDIO peripheral, the software reset will not complete.

---

### 18.3.3 Power Management

The MMC/SD/SDIO host controller can enter into different modes and save power:

- Normal mode
- Idle mode

The two modes are mutually exclusive (the module can be in normal mode or in idle mode). The MMC/SD/SDIO host controller is compliant with the PRCM module handshake protocol. When the MMC/SD/SDIO power domain is off, the only way to wake up the power domain and different MMC/SD/SDIO clocks is to monitor the mmc\_dat1 input pin state via a different GPIO line for each MMC/SD/SDIO interface.

#### 18.3.3.1 Normal Mode

The autogating of interface and functional clocks occurs when the following conditions are met:

- The SD\_SYSCONFIG[0] AUTOIDLE bit is set to 1.
- There is no transaction on the MMC interface.

The autogating of interface and functional clocks stops when the following conditions are met:

- A register access occurs through the L3 (or L4) interconnect.
- A wake-up event occurs (an interrupt from a SDIO card).
- A transaction on the MMC/SD/SDIO interface starts.

Then the MMC/SD/SDIO host controller enters in low-power state even if SD\_SYSCONFIG[0] AUTOIDLE is cleared to 0. The functional clock is internally switched off and only interconnect read and write accesses are allowed.

#### 18.3.3.2 Idle Mode

The clocks provided to MMC/SD/SDIO are switched off upon a PRCM module request. They are switched back upon module request. The MMC/SD/SDIO host controller complies with the PRCM module handshaking protocol:

- Idle request from the system power manager
- Idle acknowledgment from the MMC/SD/SDIO host controller
- Wake-up request from the MMC/SD/SDIO host controller

The idle acknowledgment varies according to the SD\_SYSCONFIG[4:3] SIDLEMODE bit field:

- 0: Force-idle mode. The MMC/SD/SDIO host controller acknowledges the system power manager request unconditionally.
- 1h: No-idle mode. The MMC/SD/SDIO host controller ignores the system power manager request and behaves normally as if the request was not asserted.
- 2h: Smart-idle mode. The MMC/SD/SDIO host controller acknowledges the system power manager

- request according to its internal state.
- 3h: Reserved.

During the smart-idle mode period, the MMC/SD/SDIO host controller acknowledges that the OCP and Functional clocks may be switched off whatever the value set in the SD\_SYS CONFIG[9:8] CLOCKACTIVITY field.

#### 18.3.3.3 Transition from Normal Mode to Smart-Idle Mode

Smart-idle mode is enabled when the SD\_SYS CONFIG[4:3] SIDLEMODE bit field is set to 2h or 3h. The MMC/SD/SDIO host controller goes into idle mode when the PRCM issues an idle request, according to its internal activity. The MMC/SD/SDIO host controller acknowledges the idle request from the PRCM after ensuring the following:

- The current multi/single-block transfer is completed.
- Any interrupt or DMA request is asserted.
- There is no card interrupt on the SD\_dat1 signal.

As long as the MMC/SD/SDIO controller does not acknowledge the idle request, if an event occurs, the MMC/SD/SDIO host controller can still generate an interrupt or a DMA request. In this case, the module ignores the idle request from the PRCM.

As soon as the MMC/SD/SDIO controller acknowledges the idle request from the PRCM:

- If Smart-Idle mode the module does not assert any new interrupt or DMA request
- If Smart-Idle wake-up-capable mode the module may generate wake-up events related to interrupt or DMA request.

#### 18.3.3.4 Transition from Smart-Idle Mode to Normal Mode

The MMC/SD/SDIO host controller detects the end of the idle period when the PRCM deasserts the idle request. For the wake-up event, there is a corresponding interrupt status in the SD\_STAT register. The MMC/SD/SDIO host controller operates the conversion between wake-up and interrupt (or DMA request) upon exit from smart-idle mode if the associated enable bit is set in the SD\_ISE register.

Interrupts and wake-up events have independent enable/disable controls, accessible through the SD\_HCTL and SD\_ISE registers. The overall consistency must be ensured by software.

The interrupt status register SD\_STAT is updated with the event that caused the wake-up in the CIRQ bit when the SD\_IE[8] CIRQ\_ENABLE associated bit is enabled. Then, the wake-up event at the origin of the transition from smart-idle mode to normal mode is converted into its corresponding interrupt or DMA request. (The SD\_STAT register is updated and the status of the interrupt signal changes.)

When the idle request from the PRCM is deasserted, the module switches back to normal mode. The module is fully operational.

#### 18.3.3.5 Force-Idle Mode

Force-idle mode is enabled when the SD\_SYS CONFIG[4:3] SIDLEMODE bit field is cleared to 0. Force-idle mode is an idle mode where the MMC/SD/SDIO host controller responds unconditionally to the idle request from the PRCM. Moreover, in this mode, the MMC/SD/SDIO host controller unconditionally deasserts interrupts and DMA request lines are asserted.

The transition from normal mode to force-idle mode does not affect the bits of the SD\_STAT register. In force-idle mode, the interrupt and DMA request lines are deasserted. Interface Clock (OCP) and functional clock (CLKADPI) can be switched off.

**CAUTION**

In Force-idle mode, an idle request from the PRCM during a command or a data transfer can lead to an unexpected and unpredictable result. When the module is idle, any access to the module generates an error as long as the OCP clock is alive.

The module exits the force-idle mode when the PRCM deasserts the idle request. Then the module switches back to normal mode. The module is fully operational. Interrupt and DMA request lines are optionally asserted one clock cycle later.

#### 18.3.3.6 Local Power Management

[Table 18-9](#) describes power-management features available for the MMC/SD/SDIO modules.

**Table 18-9. Local Power Management Features**

| Feature                | Registers                        | Description   |
|------------------------|----------------------------------|---|
| Clock Auto Gating      | SD_SYSCONFIG AUTOIDLE bit        | This bit allows a local power optimization inside module, by gating the OCP clock upon the interface activity or gating the CLKADPI clock upon the internal activity. |
| Slave Idle Modes       | SD_SYSCONFIG SIDLEMODE bit       | Force-idle, No-idle, and Smart-idle modes   |
| Clock Activity         | SD_SYSCONFIG CLOCKACTIVITY bit   | Please see <a href="#">Table 18-10</a> for configuration details.   |
| Master Standby Modes   | SD_SYSCONFIG STANDBYBYP MODE bit | Force-idle, No-idle, and Smart-idle modes   |
| Global Wake-Up Enable  | SD_SYSCONFIG ENAWAKEUP bit       | This bit enables the wake-up feature at module level.   |
| Wake-Up Sources Enable | SD_HCTL register                 | This register holds one active high enable bit per event source able to generate wake-up signal.  |

**Table 18-10. Clock Activity Settings**

| CLOCKACTIVITY Values | Clock State When Module is in IDLE State |         | Features Available when Module is in IDLE State | Wake-Up Events |
|----------------------|--|---------|---|----------------|
|                      | OCP Clock                                | CLKADPI |   |                |
| 00                   | OFF                                      | OFF     | None  | Card Interrupt |
| 10                   | OFF                                      | ON      | None  |                |
| 01                   | ON                                       | OFF     | None  |                |
| 11                   | ON                                       | ON      | All   |                |

**CAUTION**

The PRCM module has no hardware means of reading CLOCKACTIVITY settings. Thus, software must ensure consistent programming between the CLOCKACTIVITY and MMC clock PRCM control bits.

### 18.3.4 Interrupt Requests

Several internal module events can generate an interrupt. Each interrupt has a status bit, an interrupt enable bit, and a signal status enable:

- The status of each type of interrupt is automatically updated in the SD\_STAT register; it indicates which service is required.
- The interrupt status enable bits of the SD\_IE register enable/disable the automatic update of the SD\_STAT register on an event-by-event basis.
- The interrupt signal enable bits of the SD\_ISE register enable/disable the transmission of an interrupt request on the interrupt line MMC\_IRQ (from the MMC/SD/SDIO host controller to the MPU subsystem interrupt controller) on an event-by-event basis.

If an interrupt status is disabled in the SD\_IE register, then the corresponding interrupt request is not transmitted, and the value of the corresponding interrupt signal enable in the SD\_ISE register is ignored.

When an interrupt event occurs, the corresponding status bit is automatically set to 1 (the MMC/SD/SDIO host controller updates the status bit) in the SD\_STAT register. If later a mask is applied on the interrupt in the SD\_ISE register, the interrupt request is deactivated.

When the interrupt source has not been serviced, if the interrupt status is cleared in the SD\_STAT register and the corresponding mask is removed from the SD\_ISE register, the interrupt status is not asserted again in the SD\_STAT register and the MMC/SD/SDIOi host controller does not transmit an interrupt request.

#### CAUTION

If the buffer write ready interrupt (BWR) or the buffer read ready only interrupt (BRR) are not serviced and are cleared in the SD\_STAT register, and the corresponding mask is removed, then the MMC/SD/SDIOi host controller will wait for the service of the interrupt without updating the status SD\_STAT or transmitting an interrupt request.

[Table 18-11](#) lists the event flags, and their mask, that can cause module interrupts.

**Table 18-11. Events**

| Event Flag        | Event Mask                | Map To  | Description  |
|-------------------|---------------------------|---------|--|
| SD_STAT[29] BADA  | SD_IE[29]<br>BADA_ENABLE  | MMC_IRQ | Bad Access to Data space. This bit is set automatically to indicate a bad access to buffer when not allowed. This bit is set during a read access to the data register (SD_DATA) while buffer reads are not allowed (SD_PSTATE[11] BRE=0). This bit is set during a write access to the data register (SD_DATA) while buffer writes are not allowed (SD_STATE[10] BWE=0)                         |
| SD_STAT[28] CERR  | SD_IE[28]<br>CERR_ENABLE  | MMC_IRQ | Card Error. This bit is set automatically when there is at least one error in a response of type R1, R1b, R6, R5 or R5b. Only bits referenced as type E(error) in status field in the response can set a card status error. An error bit in the response is flagged only if corresponding bit in card status response errors SD_CSRE is set. There is not card detection for auto CMD12 command. |
| SD_STAT[25] ADMAE | SD_IE[25]<br>ADMAE_ENABLE | MMC_IRQ | ADMA error. This bit is set when the host controller detects errors during ADMA based data transfer. The stat of the ADMA at an error occurrence is saved in the ADMA Error Status Register. In addition, the host controller generates this interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state.  |
| SD_STAT[24] ACE   | SD_IE[24]<br>ACE_ENABLE   | MMC_IRQ | Auto CMD12 error. This bit is set automatically when one of the bits in Auto CMD12 Error status register has changed from 0 to 1   |
| SD_STAT[22] DEB   | SD_IE[22]<br>DEB_ENABLE   | MMC_IRQ | Data End Bit error. This bit is set automatically when detecting a 0 at the end bit position of read data on DAT line or at the end position of the CRC status in write mode.  |

**Table 18-11. Events (continued)**

| <b>Event Flag</b> | <b>Event Mask</b>        | <b>Map To</b> | <b>Description</b>  |
|-------------------|--------------------------|---------------|---|
| SD_STAT[21] DCRC  | SD_IE[21]<br>DCRC_ENABLE | MMC_IRQ       | Data CRC error. This bit is set automatically when there is a CRC16 error in the data phase response following a block read command or if there is a 3-bit CRC status different of a position "010" token during a block write command.   |
| SD_STAT[20] DTO   | SD_IE[20]<br>DTO_ENABLE  | MMC_IRQ       | Data Timeout error. This bit is set automatically according to the following conditions: A) busy timeout for R1b, R5b response. B) busy timeout after write CRC status. C) write CRC status timeout, or D) read data timeout.   |
| SD_STAT[19] CIE   | SD_IE[19]<br>CIE_ENABLE  | MMC_IRQ       | Command Index Error. This bit is set automatically when response index differs from corresponding command index previously emitted. The check is enabled through SD_CMD[20] CICE bit.   |
| SD_STAT[18] CEB   | SD_IE[18]<br>CEB_ENABLE  | MMC_IRQ       | Command End Bit error. This bit is set automatically when detecting a 0 at the end bit position of a command response.  |
| SD_STAT[17] CCRC  | SD_IE[17]<br>CCRC_ENABLE | MMC_IRQ       | Command CRC error. This bit is set automatically when there is a CRC7 error in the command response. CRC check is enabled through the SD_CMD[19] CCCE bit.  |
| SD_STAT[16] CTO   | SD_IE[16]<br>CTO_ENABLE  | MMC_IRQ       | Command Timeout error. This bit is set automatically when no response is received within 64 clock cycles from the end bit of the command. For commands the reply within 5 clock cycles, the timeout is still detected at 64 clock cycles.   |
| SD_STAT[15] ERRI  | SD_IE[15]<br>ERRI_ENABLE | MMC_IRQ       | Error Interrupt. If any of the bits in the Error Interrupt Status register (SD_STAT[24:15]) are set, the this bit is set to 1.  |
| SD_STAT[10] BSR   | SD_IE[10]<br>BSR_ENABLE  | MMC_IRQ       | Boot Status Received interrupt. This bit is set automatically when SD_CON[18] BOOT_CF0 is set to 1 or 2h and boot status is received on the dat0 line. This interrupt is only used for MMC cards.   |
| SD_STAT[8] CIRQ   | SD_IE[8]<br>CIRQ_ENABLE  | MMC_IRQ       | Card Interrupt. This bit is only used for SD, SDIO, and CE-ATA cards. In 1-bit mode, interrupt source is asynchronous (can be a source of asynchronous wake-up). In 4-bit mode, interrupt source is sampled during the interrupt cycle. In CE-ATA mode, interrupt source is detected when the card drive CMD line to zero during one cycle after data transmission end.   |
| SD_STAT[5] BRR    | SD_IE[5]<br>BRR_ENABLE   | MMC_IRQ       | Buffer Read ready. This bit is set automatically during a read operation to the card when one block specified by SD_BLK[10:0] BLEN is completely written in the buffer. It indicates that the memory card has filled out the buffer and the local host needs to empty the buffer by reading it.   |
| SD_STAT[4] BWR    | SD_IE[4]<br>BWR_ENABLE   | MMC_IRQ       | Buffer Write ready. This bit is automatically set during a write operation to the card when the host can write a complete block as specified by SD_BLK[10:0] BLEN. It indicates that the memory card has emptied one block from the buffer and the local host is able to write one block of data into the buffer.   |
| SD_STAT[3] DMA    | SD_IE[3]<br>DMA_ENABLE   | MMC_IRQ       | DMA interrupt. This status is set when an interrupt is required in the ADMA instruction and after the data transfer is complete.  |
| SD_STAT[2] BGE    | SD_IE[2]<br>BGE_ENABLE   | MMC_IRQ       | Block Gap event. When a stop at block gap is requested (SD_HCTL[16] SBGR), this bit is automatically set when transaction is stopped at the block gap during a read or write operation.   |
| SD_STAT[1] TC     | SD_IE[1]<br>TC_ENABLE    | MMC_IRQ       | Transfer completed. This bit is always set when a read/write transfer is completed or between two blocks when the transfer is stopped due to a stop at block gap requested (SD_HCTL[16] SBGR). In read mode this bit is automatically set on completion of a read transfer (SD_PSTATE[9] RTA). In write mode, this bit is automatically set on completion of the DAT line use (SD_PSTATE[2] DLA).   |
| SD_STAT[0] CC     | SD_IE[0]<br>CC_ENABLE    | MMC_IRQ       | Command complete. This bit is set when a 1-to-0 transition occurs in the register command inhibit (SD_PSTATE[0] CMDI). If the command is a type for which no response is expected, then the command complete interrupt is generated at the end of the command. A command timeout error (SD_STAT[16] CTO) has higher priority than command complete (SD_STAT[0] CC). If a response is expected but none is received, the a Command Timeout error is detected and signaled instead of the Command Complete interrupt. |

#### 18.3.4.1 Interrupt-Driven Operation

An interrupt enable bit must be set in the SD\_IE register to enable the module internal source of interrupt.

When an interrupt event occurs, the single interrupt line is asserted and the LH must:

- Read the SD\_STAT register to identify which event occurred.
- Write 1 into the corresponding bit of the SD\_STAT register to clear the interrupt status and release the interrupt line (if a read is done after this write, this would return 0).

---

**NOTE:** In the SD\_STAT register, Card Interrupt (CIRQ) and Error Interrupt (ERRI) bits cannot be cleared.

The SD\_STAT[8] CIRQ status bit must be masked by disabling the SD\_IE[8] CIRQ\_ENABLE bit (cleared to 0), then the interrupt routine must clear SDIO interrupt source in SDIO card common control register (CCCR).

The SD\_STAT[15] ERRRI bit is automatically cleared when all status bits in SD\_STAT[31:16] are cleared.

---

#### 18.3.4.2 Polling

When the interrupt capability of an event is disabled in the SD\_ISE register, the interrupt line is not asserted:

- Software can poll the status bit in the SD\_STAT register to detect when the corresponding event occurs.
- Writing 1 into the corresponding bit of the SD\_STAT register clears the interrupt status and does not affect the interrupt line state.

---

**NOTE:** Please see the note in [Section 18.3.4.1](#) concerning CIRQ and ERRRI bits clearing.

---

### 18.3.5 DMA Modes

The device supports DMA slave mode only. In this case, the controller is slave on DMA transaction managed by two separated requests (SDMAWREQN and SDMARREQN)

#### 18.3.5.1 DMA Slave Mode Operations

The MMC/SD/SDIO controller can be interfaced with a DMA controller. At system level, the advantage is to discharge the local host (LH) of the data transfers. The module does not support wide DMA access (above 1024 bytes) for SD cards as specified in the *SD Card Specification* and *SD Host Controller Standard Specification*.

The DMA request is issued if the following conditions are met:

- The SD\_CMD[0] DE bit is set to 1 to trigger the initial DMA request (the write must be done when running the data transfer command).
- A command was emitted on the SD\_cmd line.
- There is enough space in the buffer of the MMC/SD/SDIO controller to write an entire block (BLEN writes).

### 18.3.5.1.1 DMA Receive Mode

In a DMA block read operation (single or multiple), the request signal SDMARREQN is asserted to its active level when a complete block is written in the buffer. The block size transfer is specified in the SD\_BLK[10:0] BLEN field.

The SDMARREQN signal is deasserted to its inactive level when the sDMA has read one single word from the buffer. Only one request is sent per block; the DMA controller can make a 1-shot read access or several DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

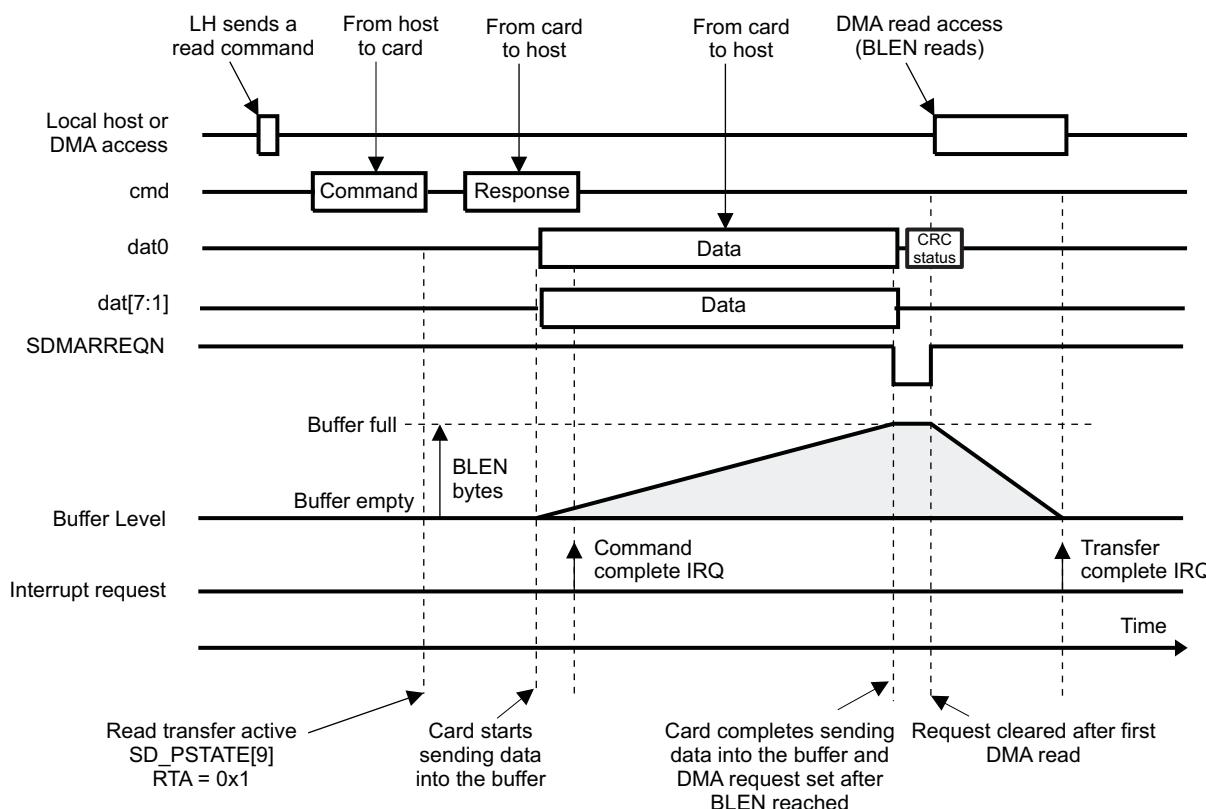
New DMA requests are internally masked if the sDMA has not read exactly BLEN bytes and a new complete block is not ready. As DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1.

The receive buffer never overflows. In multiple block transfers for block size above 512 bytes, when the buffer gets full, the MMC\_CLK clock signal (provided to the card) is momentarily stopped until the sDMA or the MPU performs a read access, which reads a complete block in the buffer.

[Figure 18-17](#) provides a summary:

- DMA transfer size = BLEN buffer size in one shot or by burst
- One DMA request per block

**Figure 18-17. DMA Receive Mode**



### 18.3.5.1.2 DMA Transmit Mode

In a DMA block write operation (single or multiple), the request signal SDMAWREQN is asserted to its active level when a complete block is to be written to the buffer. The block size transfer is specified in the SD\_BLK[10:0] BLEN field.

The SDMAWREQN signal is deasserted to its inactive level when the sDMA has written one single word to the buffer.

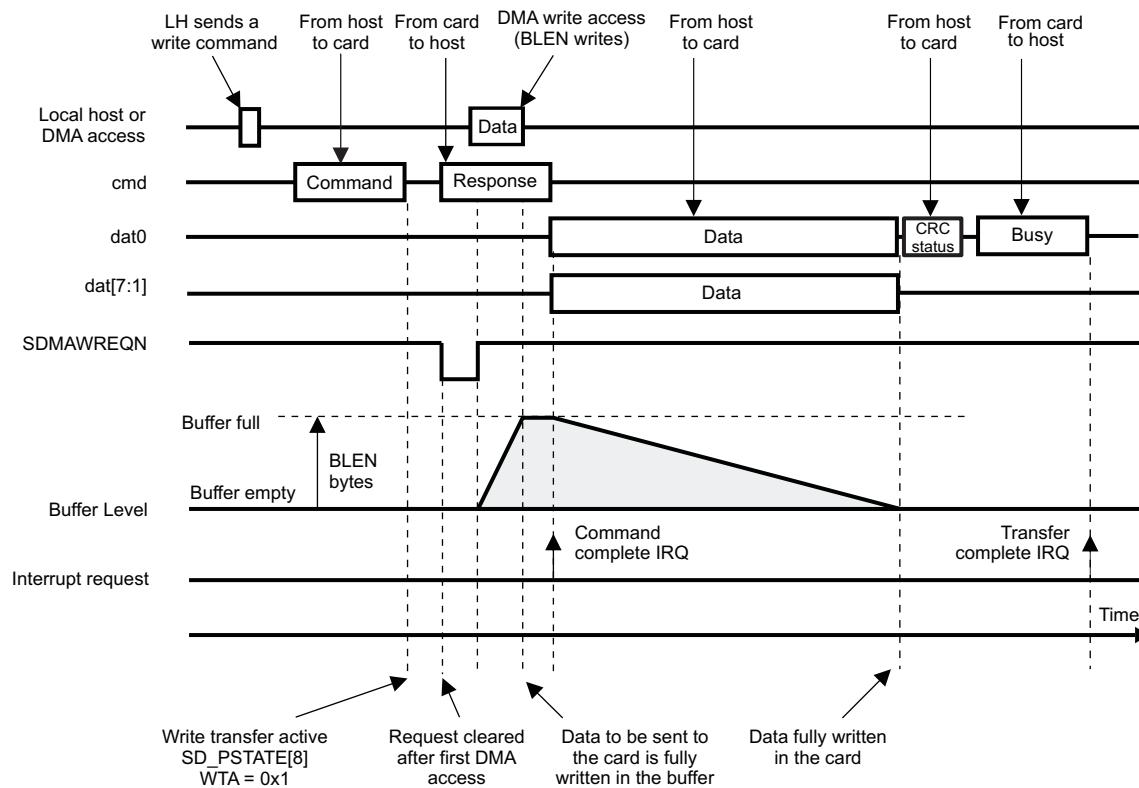
Only one request is sent per block; the DMA controller can make a 1-shot write access or multiple write DMA bursts, in which case the DMA controller must manage the number of burst accesses, according to block size BLEN field.

New DMA requests are internally masked if the sDMA has not written exactly BLEN bytes (as DMA accesses are in 32-bit, then the number of sDMA read is Integer(BLEN/4)+1) and if there is not enough memory space to write a complete block in the buffer.

[Figure 18-18](#) provides a summary:

- DMA transfer size = BLEN buffer size in one shot or by burst
- One DMA request per block

**Figure 18-18. DMA Transmit Mode**



### 18.3.6 Mode Selection

The MMC/SD/SDIO host controller can be used in two modes, MMC and SD/SDIO modes. It has been designed to be the most transparent with the type of card. The type of the card connected is differentiated by the software initialization procedure.

Software identifies the type of card connected during software initialization. For each given card type, there are corresponding commands. Some commands are not supported by all cards. See the *Multimedia Card System Specification*, the *SD Memory Card Specifications*, and the *SDIO Card Specification, Part E1* for more details.

The purpose of the module is to transfer commands and data, to whatever card is connected, respecting the protocol of the connected card. Writes and reads to the card must respect the appropriate protocol of that card.

### 18.3.7 Buffer Management

#### 18.3.7.1 Data Buffer

The MMC/SD/SDIO host controller uses a data buffer. This buffer transfers data from one data bus (Interconnect) to another data bus (SD, SDIO, or MMC card bus) and vice versa.

The buffer is the heart of the interface and ensures the transfer between the two interfaces (L4 and the card). To enhance performance, the data buffer is completed by a prefetch register and a post-write buffer that are not accessible by the host controller.

The read access time of the prefetch register is faster than the one of the data buffer. The prefetch register allows data to be read from the data buffer at an increased speed by preloading data into the prefetch register.

The entry point of the data buffer, the prefetch buffer, and the post-write buffer is the 32-bit register SD\_DATA. A write access to the SD\_DATA register followed by a read access from the SD\_DATA register corresponds to a write access to the post-write buffer followed by a read access to the prefetch buffer. As a consequence, it is normal that the data of the write access to the SD\_DATA register and the data of the read access to the SD\_DATA register are different.

The number of 32-bit accesses to the SD\_DATA register that are needed to read (or write) a data block with a size of SD\_BLK[10:0] BLEN, and equals the rounded up result of BLEN divided by 4. The maximum block size supported by the host controller is hard-coded in the register SD\_CAPA[17:16] MBL field and cannot be changed.

A read access to the SD\_DATA register is allowed only when the buffer read enable status is set to 1 (SD\_PSTATE[11] BRE); otherwise, a bad access (SD\_STAT[29] BADA) is signaled.

A write access to the SD\_DATA register is allowed only when the buffer write enable status is set to 1 (SD\_PSTATE[10] BWE); otherwise, a bad access (SD\_STAT[29] BADA) is signaled and the data is not written.

The data buffer has two modes of operation to store and read of the first and second portions of the data buffer:

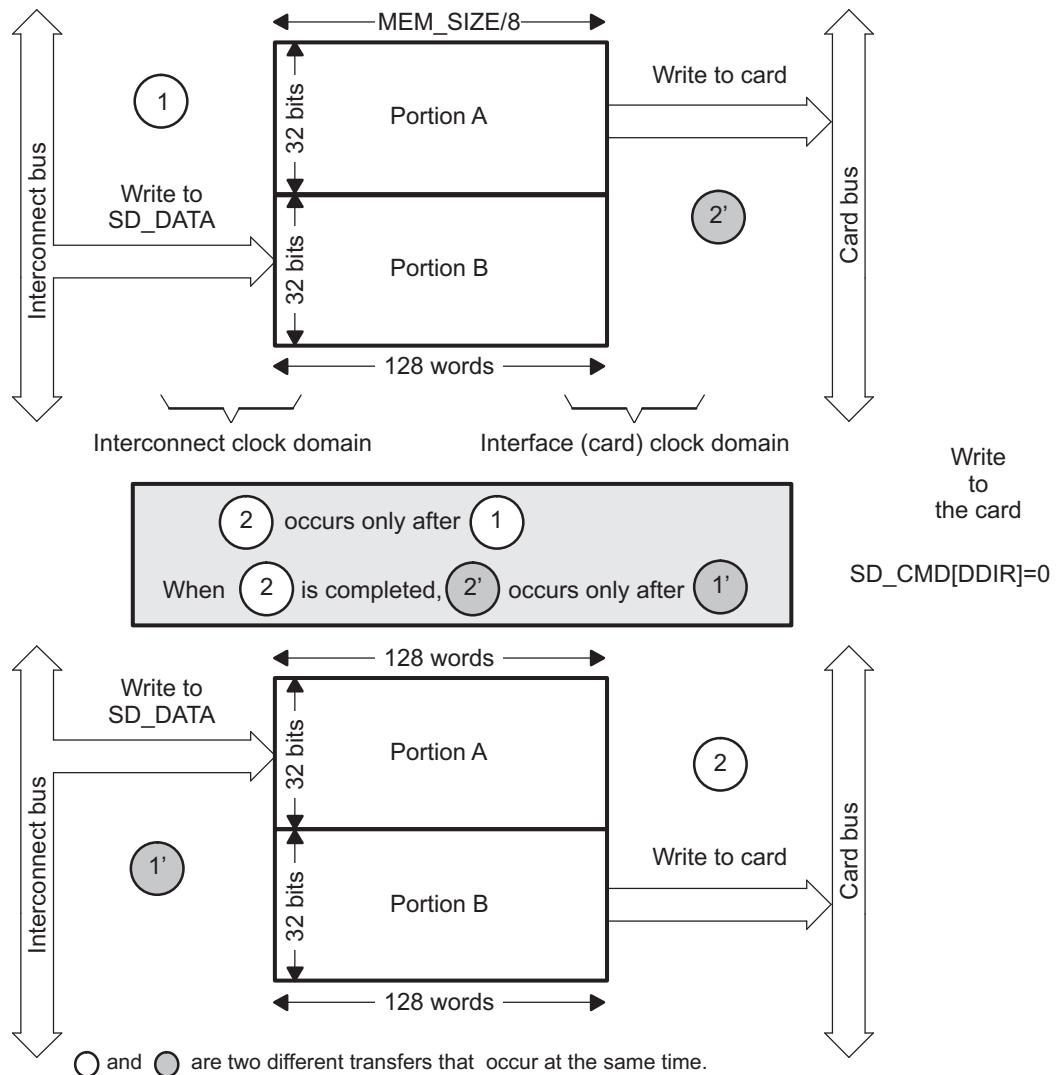
- When the size of the data block to transfer is less than or equal to MEM\_SIZE/2 (in double buffering), two data transfers can occur from one data bus to the other data bus and vice versa at the same time. The MMC/SD/SDIO controller uses the two portions of the data buffer in a ping-pong manner so that storing and reading of the first and second portions of the data buffer are automatically interchanged from time to time so that data may be read from one portion (for instance, through a DMA read access on the interconnect bus) while data (for instance, from the card) is being stored into the other portion and vice versa. When BLEN is less than or equal to 200h (that is, less or equal to 512Bytes), each of the two portions of the buffer that can be used have a size of BLEN (that is, 32-bits x BLEN div by 4). Not more than this total size of 2 times 32-bits x BLEN div by 4 can be used.
- When the size of the data block to transfer is larger than MEM\_SIZE/2, only one data transfer can occur from one data bus to the other data bus at a time. The MMC/SD/SDIO host controller uses the entire data buffer as a single portion. In this mode, a bad access (SD\_STAT[29] BADA) is signaled when two data transfers occur from one data bus to the other data bus and vice versa at the same time.

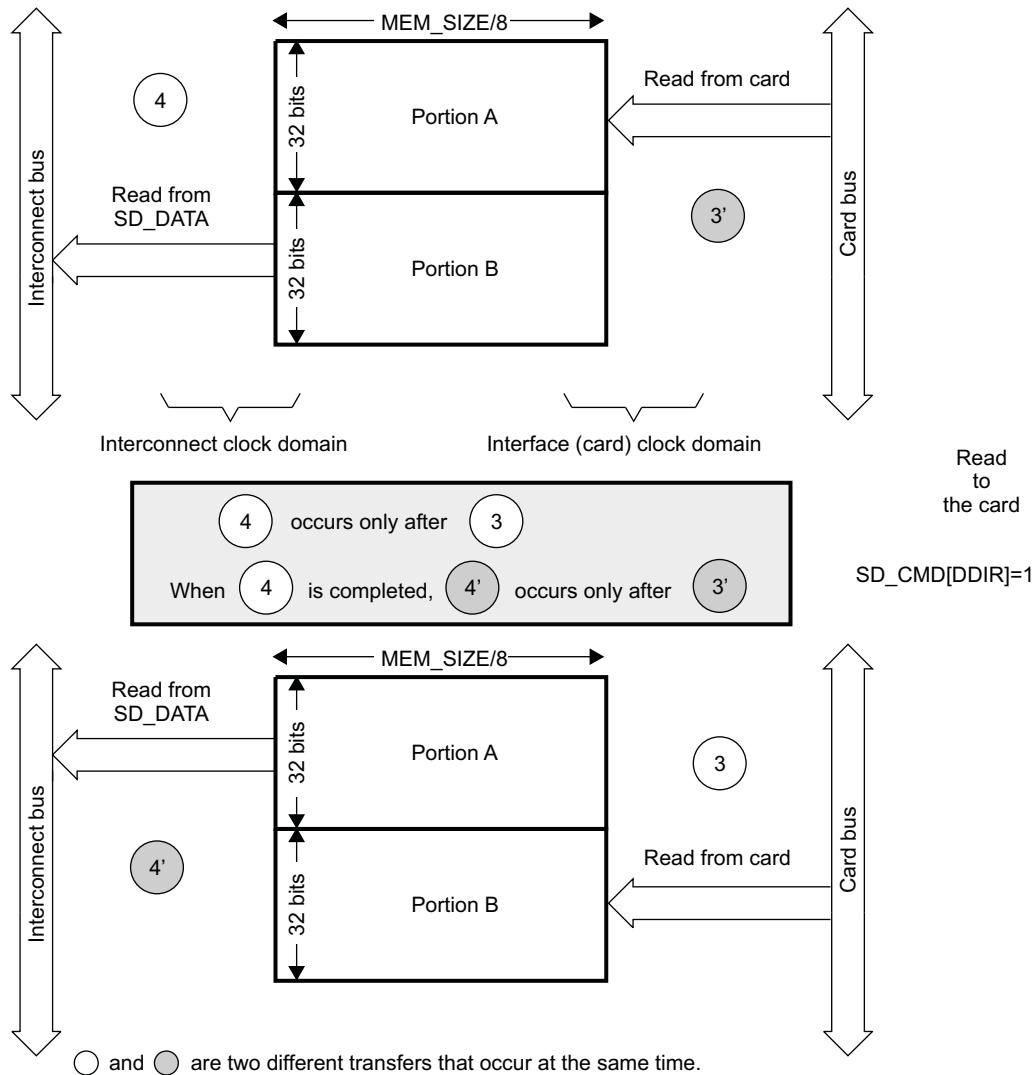
**CAUTION**

The SD\_CMD[4] DDIR bit must be configured before a transfer to indicate the direction of the transfer.

[Figure 18-19](#) shows the buffer management for writing and [Figure 18-20](#) shows the buffer management for reading.

**Figure 18-19. Buffer Management for a Write**



**Figure 18-20. Buffer Management for a Read**


#### 18.3.7.1.1 Memory Size, Block Length, and Buffer Management Relationship

The maximum block length and buffer management that can be targeted by system depend on memory depth setting.

**Table 18-12. Memory Size, BLEN, and Buffer Relationship**

| Memory Size([5:2] MEMSIZE in bytes)       | 512       | 1024               | 2048                | 4096         |
|---|-----------|--------------------|---------------------|--------------|
| Maximum block length supported            | 512       | 1024               | 2048                | 2048         |
| Double-buffering for maximum block length | N/A       | BLEN <= 512        | BLEN <= 1024        | BLEN <= 2048 |
| Single-buffering for block length         | BLEN<=512 | 512 < BLEN <= 1024 | 1024 < BLEN <= 2048 | N/A          |

### 18.3.7.1.2 Data Buffer Status

The data buffer status is defined in the following interrupt status register and status register:

- Interrupt status registers (see ):
  - SD\_STAT[29] BADA Bad access to data space
  - SD\_STAT[5] BRR Buffer read ready
  - SD\_STAT[4] BWR Buffer write ready
- Status registers (see ):
  - SD\_PSTATE[11] BRE Buffer read enable
  - SD\_PSTATE[10] BWE Buffer write enable

### 18.3.8 Transfer Process

The process of a transfer is dependent on the type of command. It can be with or without a response, with or without data.

#### 18.3.8.1 Different Types of Commands

Different types of commands are specific to MMC, SD, or SDIO cards. See the *Multimedia Card System Specification*, the *SD Memory Card Specifications*, the *SDIO Card Specification, Part E1*, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification* for more details.

#### 18.3.8.2 Different Types of Responses

Different types of responses are specific to MMC, SD, or SDIO cards. See the *Multimedia Card System Specification*, the *SD Memory Card Specifications*, the *SDIO Card Specification, Part E1*, or the *SD Card Specification, Part A2, SD Host Controller Standard Specification* for more details.

[Table 18-13](#) shows how the MMC, SD, and SDIO responses are stored in the SD\_RSPxx registers.

**Table 18-13. MMC, SD, SDIO Responses in the SD\_RSPxx Registers**

| Kind of Response                                   | Response Field             | Response Register  |
|--|----------------------------|--|
| R1, R1b (normal response), R3, R4, R5, R5b, R6, R7 | RESP[39:8] <sup>(1)</sup>  | SD_RSP10[31:0]   |
| R1b (Auto CMD12 response)                          | RESP[39:8] <sup>(1)</sup>  | SD_RSP76[31:0]   |
| R2   | RESP[127:0] <sup>(1)</sup> | SD_RSP76[31:0]<br>SD_RSP54[31:0]<br>SD_RSP32[31:0]<br>SD_RSP10[31:0] |

<sup>(1)</sup> RESP refers to the command response format described in the specifications mentioned above.

When the host controller modifies part of the SD\_RSPxx registers, it preserves the unmodified bits.

The host controller stores the Auto CMD12 response in the SD\_RSP76[31:0] register because the Host Controller may have a multiple block data DAT line transfer executing concurrently with a command. This allows the host controller to avoid overwriting the Auto CMD12 response with the command response stored in SD\_RSP10 register and vice versa.

### **18.3.9 Transfer or Command Status and Error Reporting**

Flags in the MMC/SD/SDIO host controller show status of communication with the card:

- A timeout (of a command, a data, or a response)
- A CRC

Error conditions generate interrupts. See [Table 18-14](#) and register description for more details.

**Table 18-14. CC and TC Values Upon Error Detected**

| Error hold in the SD_STAT Register | CC   | TC | Comments   |
|------------------------------------|------|----|--|
| 29                                 | BADA |    | No dependency with CC or TC.<br>BADA is related to the register accesses. Its assertion is not dependent of the ongoing transfer.  |
| 28                                 | CERR | 1  | CC is set upon CERR.   |
| 22                                 | DEB  | 1  | TC is set upon DEB.  |
| 21                                 | DCRC | 1  | TC is set upon DCRC.   |
| 20                                 | DTO  |    | DTO and TC are mutually exclusive.<br>DCRC and DEB cannot occur with DTO.  |
| 19                                 | CIE  | 1  | CC is set upon CIE.  |
| 18                                 | CEB  | 1  | CC is set upon CEB.  |
| 17                                 | CCRC | 1  | CC can be set upon CCRC - See CTO comment  |
| 16                                 | CTO  |    | CTO and CC are mutually exclusive.<br>CIE, CEB and CERR cannot occur with CTO.<br>CTO can occur at the same time as CCRC it indicates a command abort due to a contention on CMD line. In this case no CC appears. |

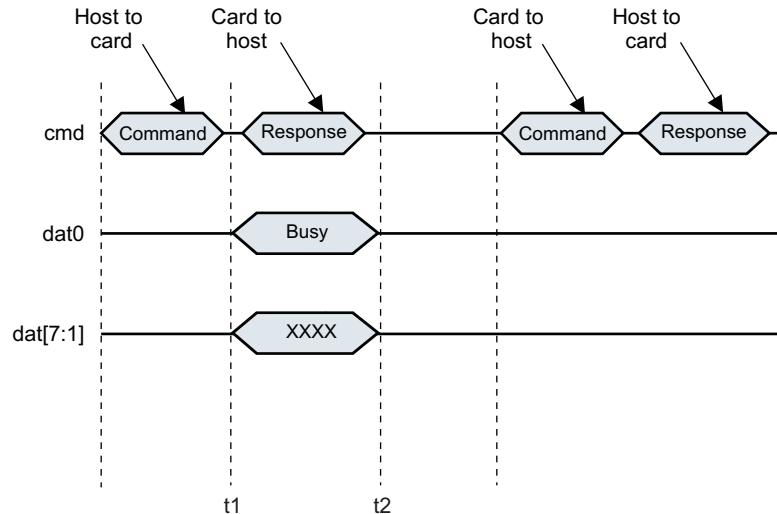
SD\_STAT[21] DCRC event can be asserted in the following conditions:

- Busy timeout for R1b, R5b response type
- Busy timeout after write CRC status
- Write CRC status timeout
- Read data timeout
- Boot acknowledge timeout

### 18.3.9.1 Busy Timeout for R1b, R5b Response Type

Figure 18-21 shows DCRD event condition asserted when there is a busy timeout for R1b or R5b responses.

**Figure 18-21. Busy Timeout for R1b, R5b Responses**



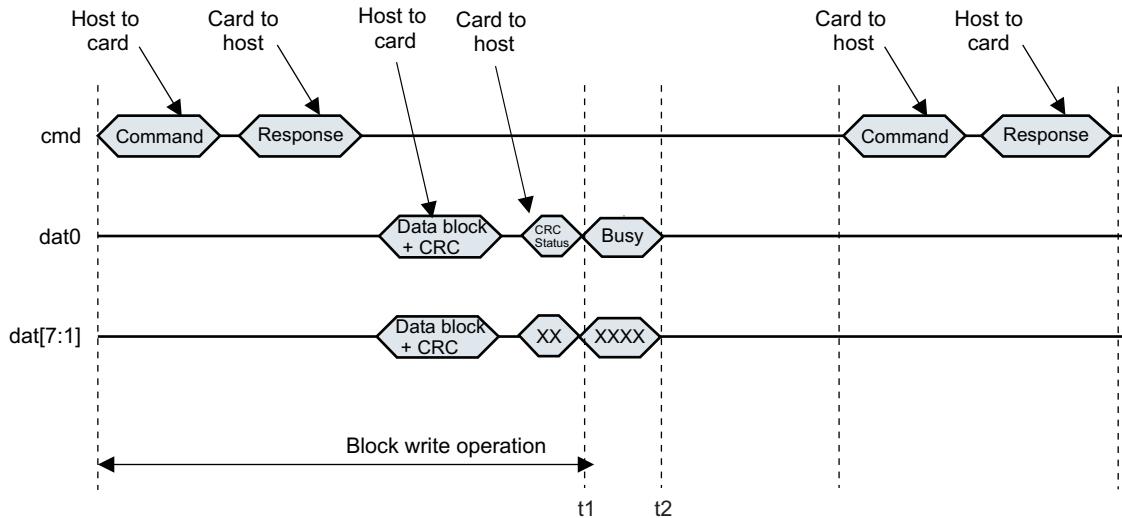
t1 - Data timeout counter is loaded and starts after R1b, R5b response type.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

### 18.3.9.2 Busy Timeout After Write CRC Status

Figure 18-22 shows DCRC event condition asserted when there is busy timeout after write CRC status.

**Figure 18-22. Busy Timeout After Write CRC Status**



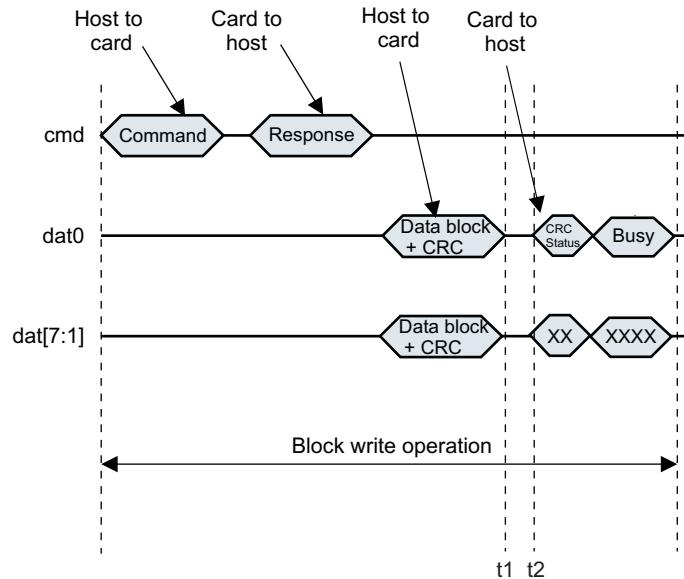
t1 - Data timeout counter is loaded and starts after CRC status.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

### 18.3.9.3 Write CRC Status Timeout

Figure 18-23 shows DCRC event condition asserted when there is write CRC status timeout.

**Figure 18-23. Write CRC Status Timeout**



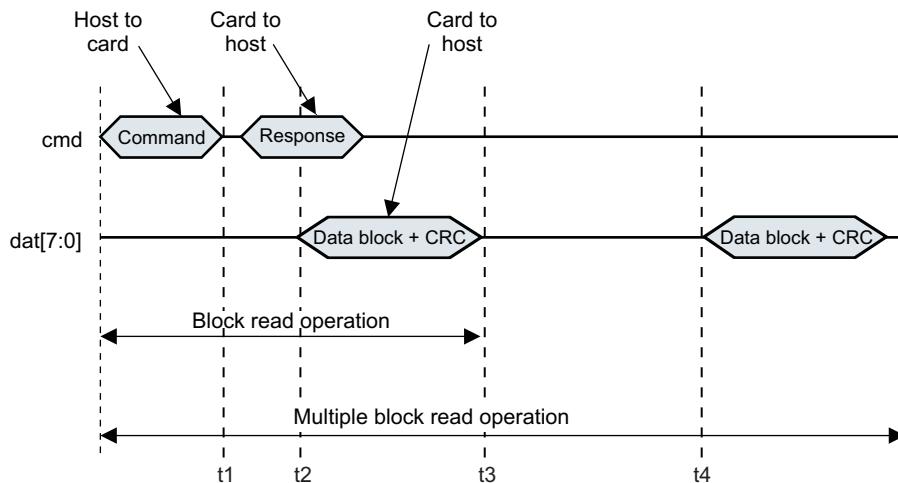
t1 - Data timeout counter is loaded and starts after Data block + CRC.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

### 18.3.9.4 Read Data Timeout

Figure 18-24 shows DCRC event condition asserted when there is read data timeout.

**Figure 18-24. Read Data Timeout**



t1 - Data timeout counter is loaded and starts after Command transmission.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

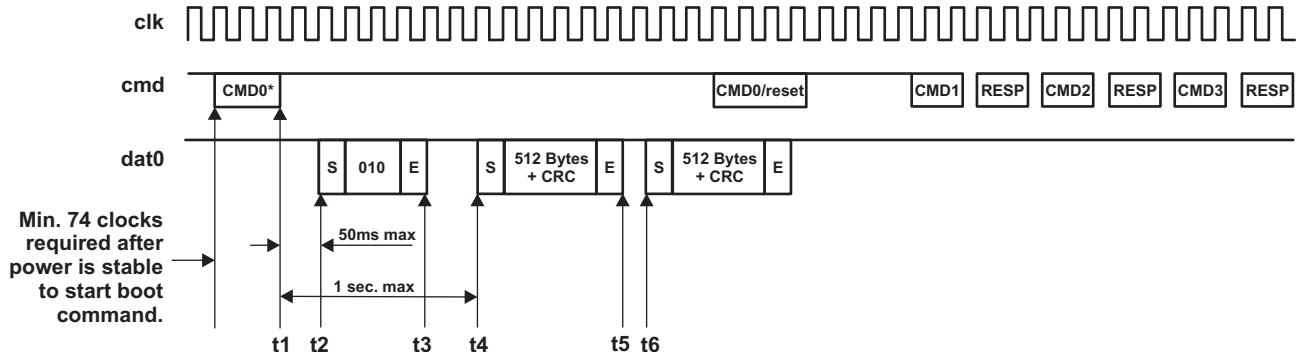
t3 - Data timeout counter is loaded and starts after Data block + CRC transmission.

t4 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

### 18.3.9.5 Boot Acknowledge Timeout

Figure 18-25 shows DCRC event condition asserted when there is boot acknowledge timeout and CMD0 is used.

**Figure 18-25. Boot Acknowledge Timeout When Using CMD0**



\* Refer to MMC Specification for correct argument.

t1 - Data timeout counter is loaded and starts after CMD0.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

t3 - Data timeout counter is loaded and starts.

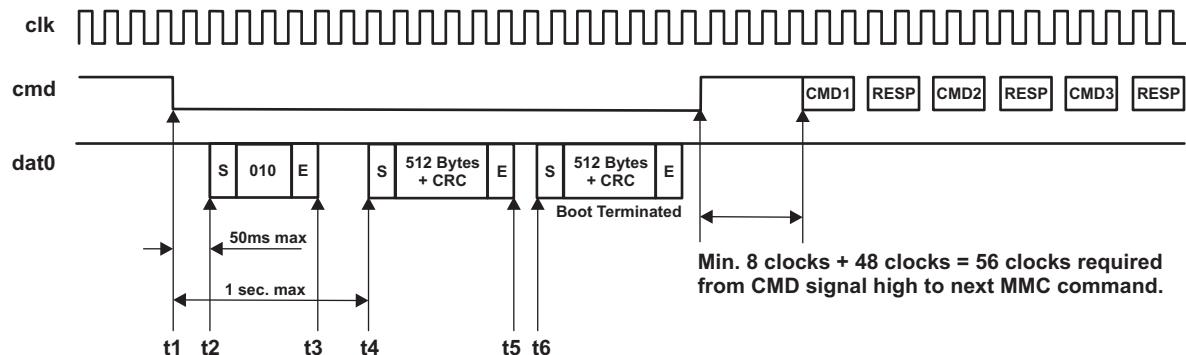
t4 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

t5 - Data timeout counter is loaded and starts after Data + CRC transmission.

t6 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

Figure 18-26 shows DCRC event condition asserted when there is boot acknowledge timeout and CMD line is held low.

**Figure 18-26. Boot Acknowledge Timeout When CMD Held Low**



t1 - Data timeout counter is loaded and starts after cmd line is tied to 0.

t2 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

t3 - Data timeout counter is loaded and starts.

t4 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

t5 - Data timeout counter is loaded and starts after Data + CRC transmission.

t6 - Data timeout counter stops and if it is 0, SD\_STAT[21] DCRC is generated.

### 18.3.10 Auto Command 12 Timings

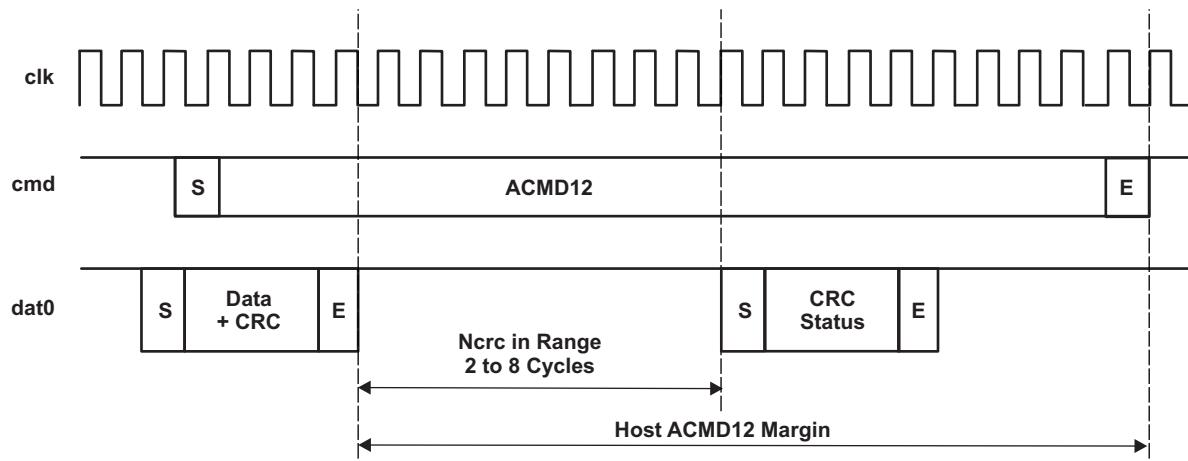
With the UHS definition of SD cards with higher frequency for MMC clocks up to 208, SD standard imposes a specific timing for Auto CMD12 "end bit" arrival.

#### 18.3.10.1 Auto Command 12 Timings During Write Transfer

A margin named Ncrc in range of 2 to 8 cycles has been defined for SDR50 and SDR104 card components for write data transfers, as auto command 12 'end bit' shall arrive after the CRC status "end bit".

Figure 18-27 shows auto CMD12 timings during write transfer.

**Figure 18-27. Auto CMD12 Timing During Write Transfer**



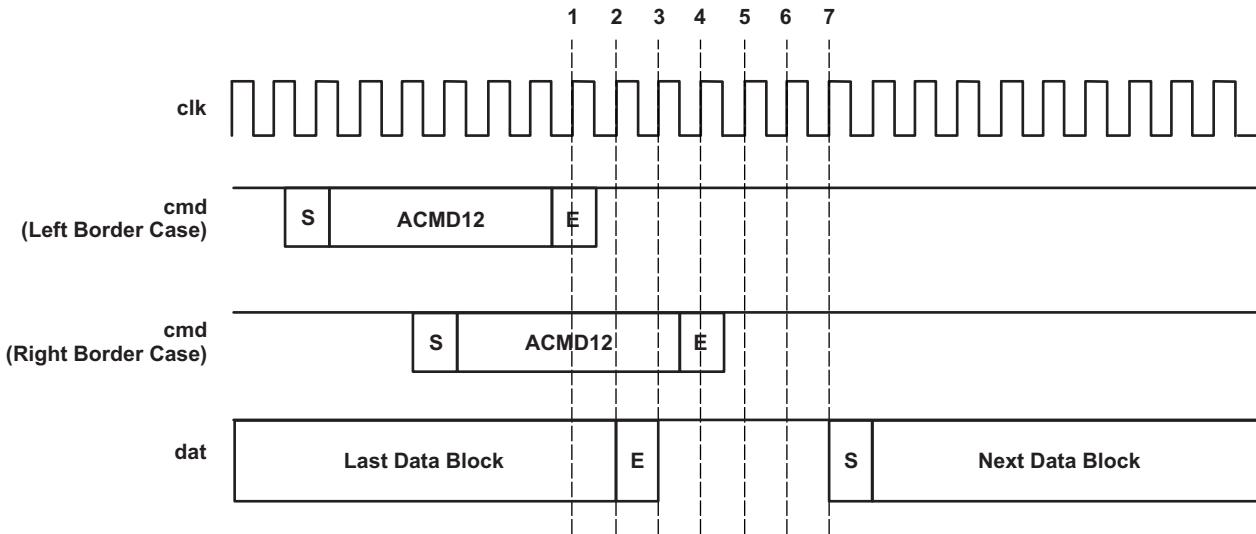
The Host controller has a margin of 18 clock cycles to make sure that auto CMD12 'end bit' arrives after the CRC status. This margin does not depend on MMC bus configuration, DDR or standard transfer, 1,4 or 8 bus width.

### 18.3.10.2 Auto Command 12 Timings During Read Transfer

With UHS very high speed cards gap timing between 2 successive cards has been extended to 4 cycles instead of 2. By the way it gives more flexibility for Host Auto CMD12 arrival in order to receive the last complete and reliable block. SD controller only follows the 'Left Border Case' defined by SD UHS specification.

Figure 18-28 shows ACMD12 timings during read transfer.

**Figure 18-28. Auto Command 12 Timings During Read Transfer**



The Auto CMD12 arrival sent by the Host controller is not sensitive to the MMC bus configuration whether it is DDR or standard transfer and whether it is a 1,4 or 8 bit bus width transfer.

### 18.3.11 Transfer Stop

Whenever a transfer is initiated, the transmission may be willed to stop whereas it is still not finished. Several cases can be faced depending on the transfer type:

- Multiple blocks oriented transfers (for which transfer length is known)
- Continuous stream transfers (which have an infinite length)

---

**NOTE:** Since the MMC/SD/SDIO controller manages transfers based on a block granularity, the buffer will accept a block only if there is enough space to completely store it. Consequently, if a block is pending in the buffer, no command will be sent to the card because the card clock will be shut off by the controller.

---

The MMC/SD/SDIO controller includes two features which make a transfer stop more convenient and easier to manage:

- Auto CMD12 (for MMC and SD only).

This feature is enabled by setting the SD\_CMD[2] ACEN bit to 1 (this setting is relevant for a MMC/SD transfer with a known number of blocks to transfer). When the Auto CMD12 feature is enabled, the MMC/SD/SDIO controller will automatically issue a CMD12 command when the expected number of blocks has been exchanged.

- Stop at block gap

This feature is enabled by setting the SD\_HCTL[16] SBGR bit to 1. When enabled, this capability holds the transfer on until the end of a block boundary. If a stop transmission is needed, software can use this pause to send a CMD12 to the card.

Table 18-15 shows the common ways to stop a transfer, indicating command to send and features to enable.

**Table 18-15. MMC/SD/SDIO Controller Transfer Stop Command Summary**

|                                      |   | WRITE Transfer  |  | READ Transfer   |   |
|--------------------------------------|---|---|--|---|---|
|                                      |   | MMC/SD  | SDIO   | MMC/SD  | SDIO  |
| Single block                         |   | Transfer ends automatically<br>Wait TC                      | Transfer ends automatically<br>Wait TC                     | Transfer ends automatically<br>Wait TC                      | Transfer ends automatically<br>Wait TC  |
| Multi blocks<br>(finite or infinite) | Before the programmed block boundary                      | Send CMD12<br>Wait TC                                       | Send CMD52<br>Wait TC                                      | Send CMD12<br>Wait TC                                       | Send CMD52<br>Wait TC   |
|                                      | Stop at the end of the transfer<br>(finite transfer only) | Auto CMD12 active<br>Transfer ends automatically<br>Wait TC | Set SD_HCTL[16]<br>SBGR bit to 1.<br>Send CMD52<br>Wait TC | Auto CMD12 active<br>Transfer ends automatically<br>Wait TC | If READ_WAIT supported<br>Stop at block gap<br>Wait TC<br><br>If READ_WAIT not supported<br>Send CMD52<br>Wait TC |

---

**NOTE:** The MMC/SD/SDIO controller will send the stop command to the card on a block boundary, regardless the moment the command was written to the controller registers.

---

### 18.3.12 Output Signals Generation

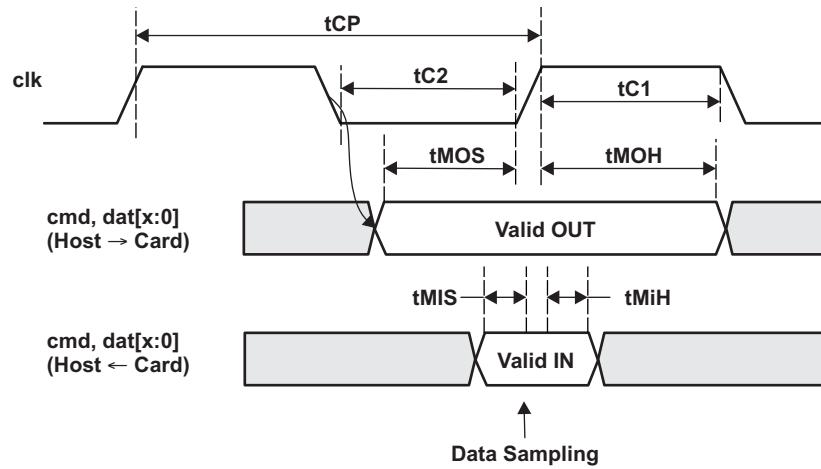
The MMC/SD/SDIO output signals can be driven on either falling edge or rising edge depending on the SD\_HCTL[2] HSPE bit. This feature allows to reach better timing performance, and thus to increase data transfer frequency.

#### 18.3.12.1 Generation on Falling Edge of MMC Clock

The controller is by default in this mode to maximize hold timings. In this case, SD\_HCTL[2] HSPE bit is cleared to 0.

[Figure 18-29](#) shows the output signals of the module when generating from the falling edge of the MMC clock.

**Figure 18-29. Output Driven on Falling Edge**



#### 18.3.12.2 Generation on Rising Edge of MMC Clock

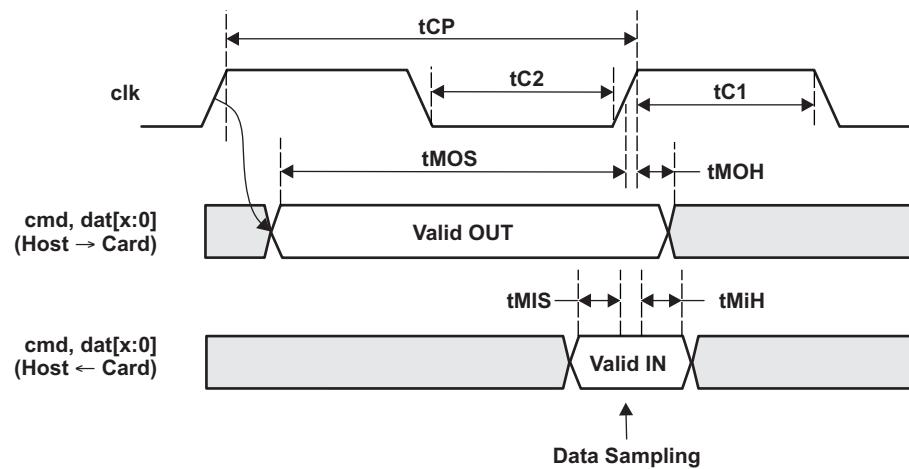
This mode increases setup timings and allows reaching higher bus frequency. This feature is activated by setting SD\_HCTL[2] HSPE bit to 1. The controller shall be set in this mode to support SDR transfers.

---

**NOTE:** Do not use this feature in Dual Data Rate mode (when SD\_CON[19] DDR is set to 1).

---

[Figure 18-30](#) shows the output signals of the module when generating from the rising edge of the MMC clock.

**Figure 18-30. Output Driven on Rising Edge**


### 18.3.13 Card Boot Mode Management

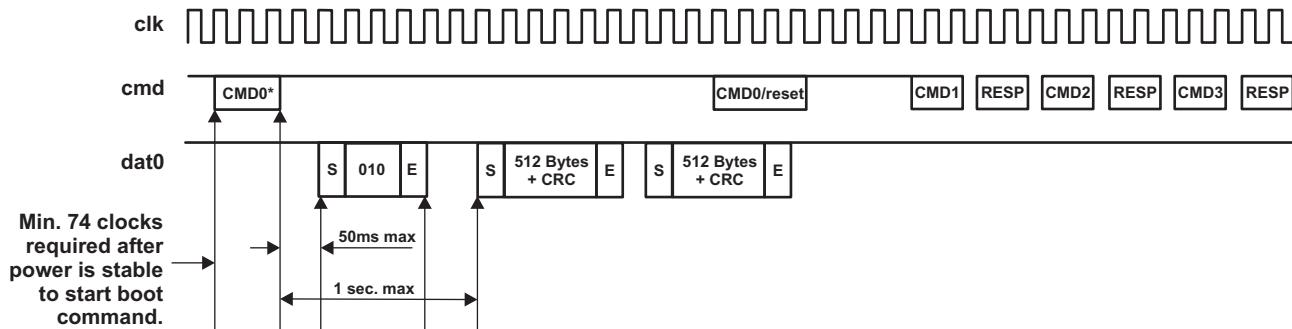
Boot Operation Mode allows the MMC/SD/SDIO host controller to read boot data from the connected slave (MMC device) by keeping CMD line low after power-on (or sending CMD0 with specific argument) before issuing CMD1. The data can be read from either boot area or user area, depending on register setting. Power-on boot defines a way for the boot-code to be accessed by the MMC/SD/SDIO host controller without an upper-level software driver, speeding the time it takes for a controller to access the boot code.

The two possible ways to issue a boot command (either issuing a CMD0 or driving the CMD line to 0 during the whole boot phase) are described in the following sections.

#### 18.3.13.1 Boot Mode Using CMD0

Figure 18-31 shows the timing diagram of a boot sequence using CMD0.

**Figure 18-31. Boot Mode With CMD0**



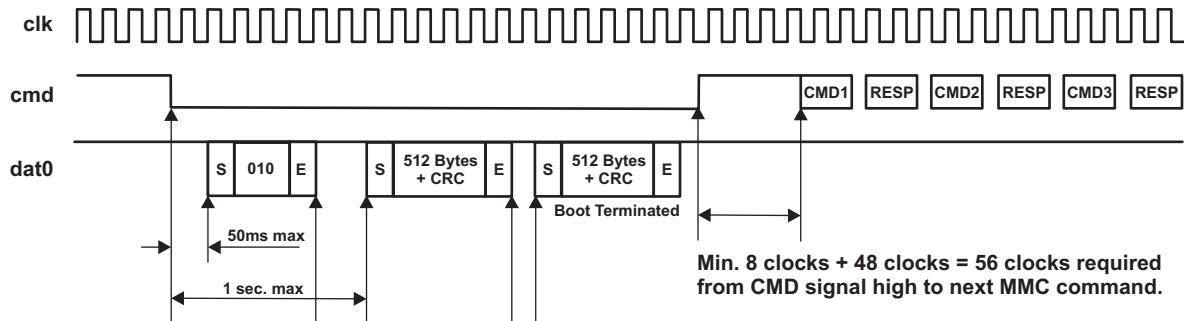
\* Refer to MMC Specification for correct argument.

- Configure:
  - MMCHS\_CON[BOOT\_CF0] to 0
  - MMCHS\_CON[BOOT\_ACK] (if an acknowledge will be received) to 0x1
  - MMCHS\_BLK with the correct block length and number of block
  - MMCHS\_SYSCTL[DTO] for timeout  
If transfer is done in DDR mode also set MMCHS\_CON[DDR] to 1.
- Write register MMCHS\_ARG with correct argument (see MMC Specification).
- Write in MMCHS\_CMD register to start CMD0 transfer with these bit fields set:
  - INDX set to 0x00
  - DP set to '1'
  - DDIR set to '1'
  - MSBS set to '1'
  - BCE set to '1'
- If boot status is not received within the timing defined, the MMCHS\_STAT[DTO] will be generated. Otherwise the MMCHS\_STAT[BSR] is arisen.
- After the transfer is complete, the controller will generate the MMCHS\_STAT[TC], and then the system can emit another CMD0 (MMCHS\_CON[BOOT\_ACK] previously cleared to 0x0) to exit the card from boot state.
- If the system wants to abort the boot sequence it must issue a CMD0 with MMCHS\_CMD[CMD\_TYPE] set to 0x3 (MMCHS\_CON[BOOT\_ACK] previously cleared to 0x0) during the transfer to abort transfer and enable card to exit from boot state.

### 18.3.13.2 Boot Mode With CMD Held Low

Figure 18-32 shows the timing diagram of a boot sequence with CMD line tied to 0.

**Figure 18-32. Boot Mode With CMD Line Tied to 0**



- Configure:
  - MMCHS\_CON[BOOT\_CF0] and MMCHS\_CON[BOOT\_ACK] (if an acknowledge will be received) to 0x1
  - MMCHS\_BLK with correct block length and number of block
  - MMCHS\_SYSCTL[DTO] for timeout
  - If transfer is done in DDR mode also set MMCHS\_CON[DDR] to 1.
- Write in MMCHS\_CMD register to start boot sequence with:
  - DP set to '1'
  - DDIR set to '1'
  - MSBS set to '1'
  - BCE set to '1'
 This leads the controller to force CMD line to '0'.
- If the boot status is not received within the timing defined, the MMCHS\_STAT[DTO] will be generated. Otherwise the MMCHS\_STAT[BSR] is arisen.
- After the transfer is complete, the controller will generate the MMCHS\_STAT[TC], and then the system must clear MMCHS\_CON[BOOT\_CF0] to 0x0 to release the CMD line and enable the card to exit from boot state.
- If the system wants to abort the boot sequence it must clear MMCHS\_CON[BOOT\_CF0] to 0x0 during transfer to enable the card to exit from boot state.

### 18.3.14 CE-ATA Command Completion Disable Management

The MMC/SD/SDIO controller supports CE-ATA features, in particular the detection of command completion token. When a command that requires a command completion signal (SD\_CON[12] CEATA and SD\_CMD[2] ACEN set to 1) is launched, the host system is no longer allowed to emit a new command in parallel of data transfer unless it is a command completion disable token.

The settings to emit a command completion disable token follow:

- SD\_CON[12] CEATA is set to 1.
- SD\_CON[2] HR set to 1.
- Clear the SD\_ARG register.
- Write into SD\_CMD register with value 0000 0000h.

When a command completion disable token was emitted (that is, SD\_STAT[0] CC received), the host system is again allowed to emit another type of command (for example a transfer abort command CMD12 to abort transfer).

A critical case can be met when command completion signal disable (CCSD) is emitted during the last data block transfer, the sequence on command line could be sent very close to command completion signal (CCS) token sent by the card.

Three cases can be met:

- CCS is receive just before CCSD is emitted:  
An interrupt CIRQ is generated with CCS detection, CCSD is transmitted to card then an interrupt CC is generated when CCSD ends. In this case, card consider the CCSD sequence.
- CCS is not generated or generated during the CCSD transfer:  
The CCS bit cannot be detected (conflict is not possible as they drive the same level on command line, then no CIRQ interrupt is generated; besides CC interrupt is generated when CCSD ends).
- CCS is generated without CCSD token required:  
Only the interrupt CIRQ is generated when CCS is detected.

### 18.3.15 Test Registers

Test registers are available to be compliant with SD Host controller specification. This feature is useful to generate interrupts manually for driver debugging. The Force Event register (SD\_FE) is used to control the Error Interrupt Status and Auto CMD12 Error Status. The System Test register (SD\_SYSTEST) is used to control the signals that connect to I/O pins when the module is configured in system test (SD\_CON[4] MODE = 1) mode for boundary connectivity verification.

### 18.3.16 MMC/SD/SDIO Hardware Status Features

Table 18-16 summarizes the MMC/SD/SDIO hardware status features.

**Table 18-16. MMC/SD/SDIO Hardware Status Features**

| Feature                      | Type   | Register/Bit Field/Observability Control | Description   |
|------------------------------|--------|--|---|
| Interrupt flags              |        | See <a href="#">Section 18.3.4</a> .     |   |
| CMD line signal level        | Status | [24] CLEV                                | Indicates the level of the cmd line   |
| DAT lines signal level       | Status | [23:20] DLEV                             | Indicates the level of the data lines   |
| Buffer read enable           | Status | [11] BRE                                 | Readable data exists in the buffer.   |
| Buffer write enable          | Status | [10] BWE                                 | Indicates whether there is enough space in the buffer to write BLEN bytes of data |
| Read transfer active         | Status | [9] RTA                                  | This status is used for detecting completion of a read transfer.                  |
| Write transfer active        | Status | [8] WTA                                  | This status indicates a write transfer active.                                    |
| Data line active             | Status | [2] DLA                                  | Indicates whether the data lines are active                                       |
| Command Inhibit (data lines) | Status | [1] DATI                                 | Indicates whether issuing of command using data lines is allowed                  |
| Command inhibit (CMD line)   | Status | [0] CMDI                                 | Indicates whether issuing of command using CMD line is allowed                    |

## 18.4 Low-Level Programming Models

### 18.4.1 Surrounding Modules Global Initialization

This section identifies the requirements of initializing the surrounding modules when the module has to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the MMC/SD/SDIO modules.

**Table 18-17. Global Init for Surrounding Modules**

| Surrounding Modules     | Comments  |
|-------------------------|---|
| PRCM                    | Module interface and functional clocks must be enabled. For more information, see <a href="#">Chapter 8, Power, Reset, and Clock Management</a> . |
| Control module          | Module-specific pad muxing and configuration must be set in the control module. See <a href="#">Chapter 9, Control Module</a> .                   |
| (optional) MPU INTC     | MPU INTC configuration must be done to enable the interrupts from the SD module. See <a href="#">Chapter 6, Interrupts</a> .                      |
| (optional) EDMA         | DMA configuration must be done to enable the module DMA channel requests. See <a href="#">Chapter 11, EDMA</a> .                                  |
| (optional) Interconnect | For more information about the interconnect configuration, see <a href="#">Chapter 10, Interconnects</a> .  |

---

**NOTE:** The MPU interrupt controller and the EDMA configurations are necessary, if the interrupt and DMA based communication modes are used.

---

### 18.4.2 MMC/SD/SDIO Controller Initialization Flow

The next sections outline the four steps to initialize the MMC/SD/SDIO controller:

- Initialize Clocks
- Software reset of the controller
- Set module's hardware capabilities
- Set module's Idle and Wake-Up modes

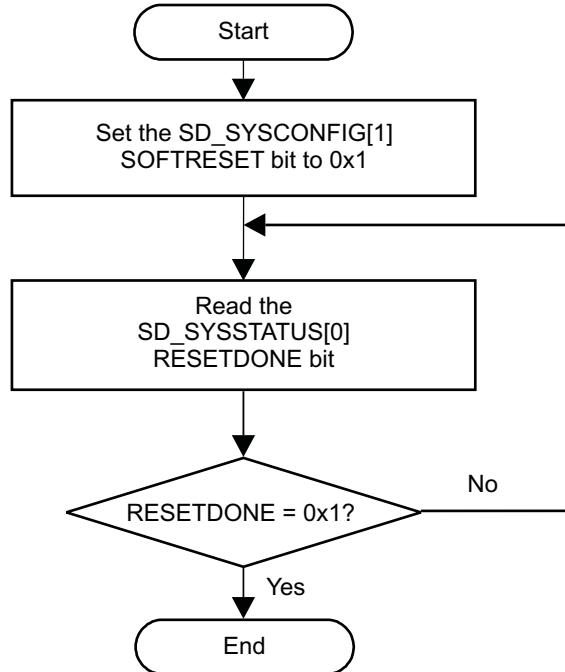
#### 18.4.2.1 Enable OCP and CLKADPI Clocks

Prior to any SD register access one must enable the SD OCP clock and CLKADPI clock in PRCM module registers. For more information, see [Chapter 8, Power, Reset, and Clock Management](#).

#### 18.4.2.2 SD Soft Reset Flow

Figure 18-33 shows the soft reset process of MMC/SD/SDIO controller.

**Figure 18-33. MMC/SD/SDIO Controller Software Reset Flow**



#### 18.4.2.3 Set SD Default Capabilities

Software must read capabilities (in boot ROM for instance) and is allowed to set (write) SD\_CAPA[26:24] and SD\_CUR\_CAPA[23:0] registers before the MMC/SD/SDIO host driver is started.

#### 18.4.2.4 Wake-Up Configuration

Table 18-18 details SD controller wake-up configuration.

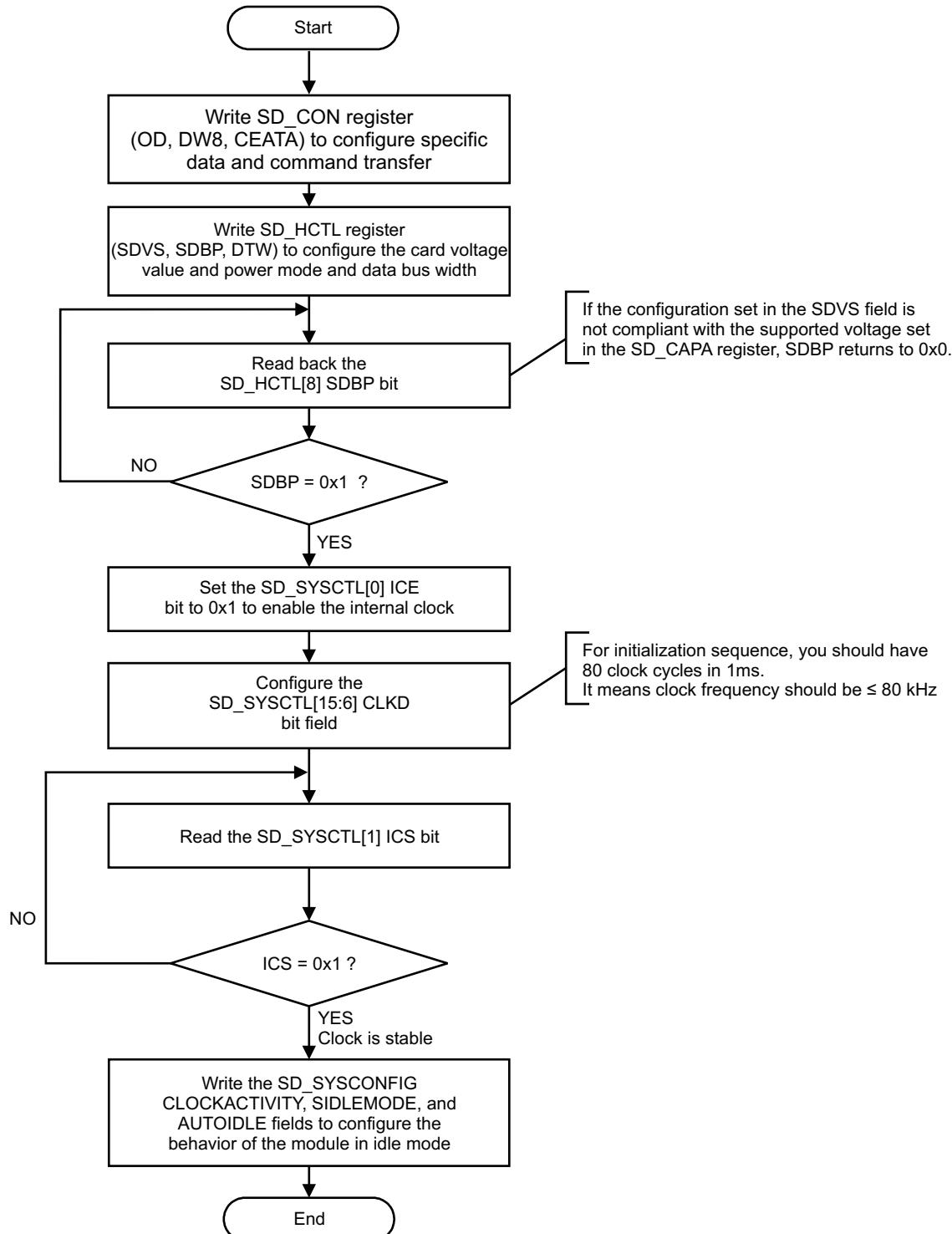
**Table 18-18. MMC/SD/SDIO Controller Wake-Up Configuration**

| Step   | Access Type | Register/Bit Field/Programming Model |
|--|-------------|--------------------------------------|
| Configure wake-up bit (if necessary).                      | W           | SD_SYSCONFIG[2] ENAWAKEUP            |
| Enable wake-up events on SD card interrupt (if necessary). | W           | SD_HCTL[24] IWE                      |
| SDIO Card onlyEnable card interrupt (if necessary).        | W           | SD_IE[8] CIRQENABLE                  |

#### 18.4.2.5 MMC Host and Bus Configuration

Figure 18-34 details the MMC bus configuration process.

**Figure 18-34. MMC/SD/SDIO Controller Bus Configuration Flow**



### 18.4.3 Operational Modes Configuration

#### 18.4.3.1 Basic Operations for MMC/SD/SDIO Host Controller

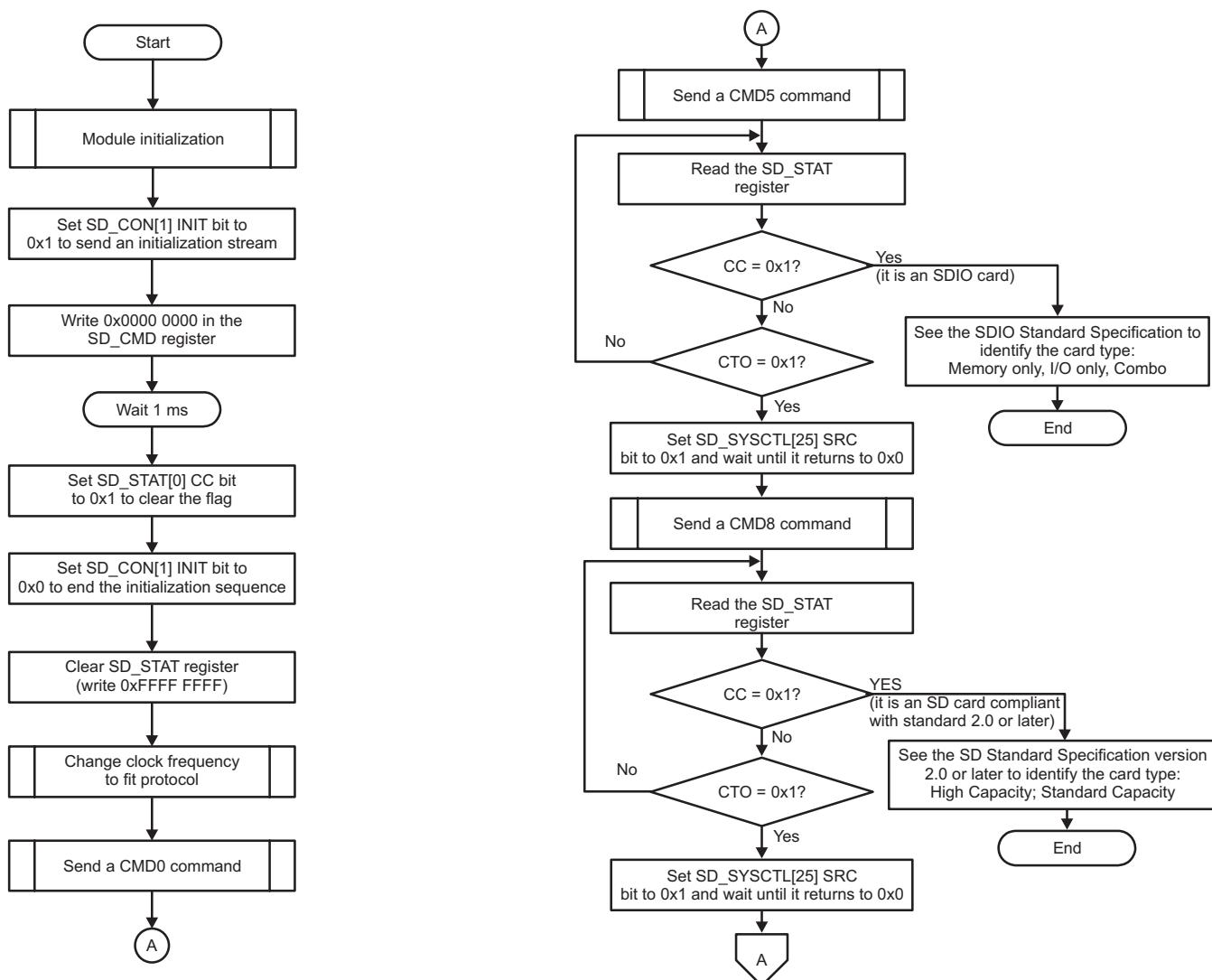
The MMC/SD/SDIO controller performs data transfers: data to card (referred to as write transfers) and data from card (referred to as read transfers).

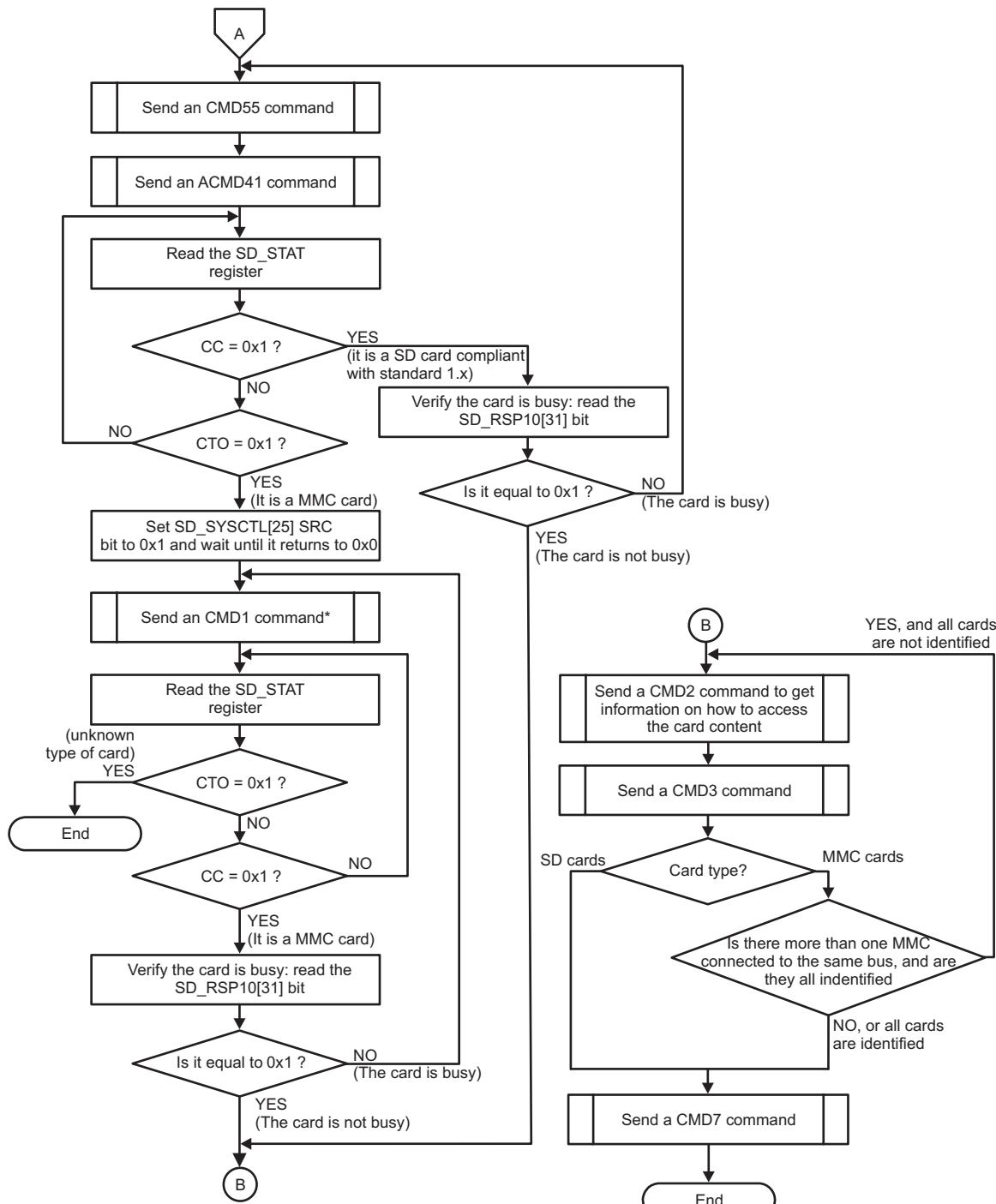
The host controller requires transfers to run on a block-by-block basis, rather than on a DMA burst size basis. A single DMA request (or block request interrupt) is signaled for each block. Pipelining is supported as long as the block size is less than one half of the memory buffer size.

#### 18.4.3.2 Card Detection, Identification, and Selection

Figure 18-35 and Figure 18-36 show the card identification and selection process.

**Figure 18-35. MMC/SD/SDIO Controller Card Identification and Selection - Part 1**



**Figure 18-36. MMC/SD/SDIO Controller Card Identification and Selection - Part 2**


## 18.5 Multimedia Card Registers

### CAUTION

The MMC/SD/SDIO registers are limited to 32-bit data accesses. 16-bit and 8-bit are not allowed and can corrupt register content.

### 18.5.1 MULTIMEDIA\_CARD Registers

[Table 18-19](#) lists the memory-mapped registers for the MULTIMEDIA\_CARD. All register offset addresses not listed in [Table 18-19](#) should be considered as reserved locations and the register contents should not be modified.

**Table 18-19. MULTIMEDIA\_CARD REGISTERS**

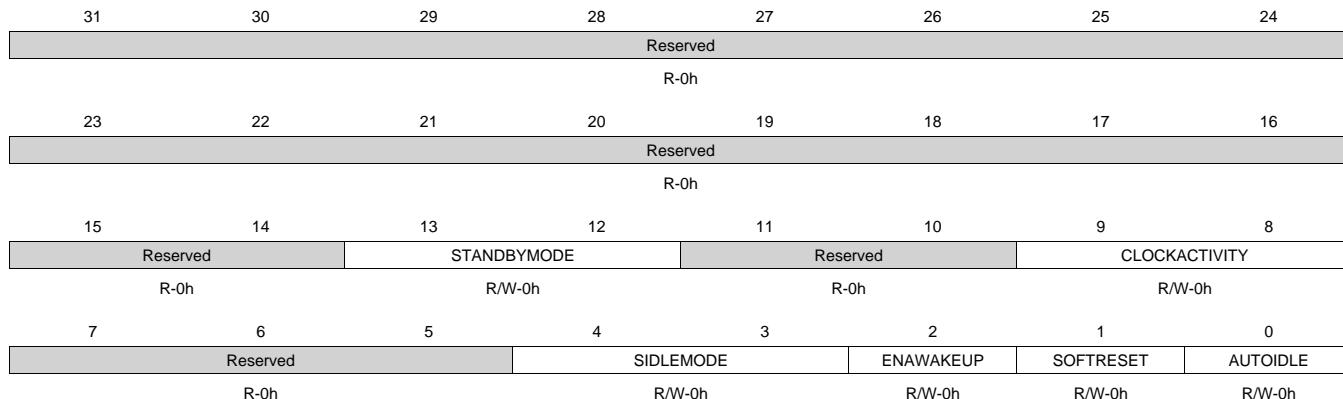
| Offset | Acronym      | Register Name | Section                           |
|--------|--------------|---------------|-----------------------------------|
| 110h   | SD_SYSCONFIG |               | <a href="#">Section 18.5.1.1</a>  |
| 114h   | SD_SYSSTATUS |               | <a href="#">Section 18.5.1.2</a>  |
| 124h   | SD_CSRE      |               | <a href="#">Section 18.5.1.3</a>  |
| 128h   | SD_SYSTEST   |               | <a href="#">Section 18.5.1.4</a>  |
| 12Ch   | SD_CON       |               | <a href="#">Section 18.5.1.5</a>  |
| 130h   | SD_PWCNT     |               | <a href="#">Section 18.5.1.6</a>  |
| 200h   | SD_SDMASA    |               | <a href="#">Section 18.5.1.7</a>  |
| 204h   | SD_BLK       |               | <a href="#">Section 18.5.1.8</a>  |
| 208h   | SD_ARG       |               | <a href="#">Section 18.5.1.9</a>  |
| 20Ch   | SD_CMD       |               | <a href="#">Section 18.5.1.10</a> |
| 210h   | SD_RSP10     |               | <a href="#">Section 18.5.1.11</a> |
| 214h   | SD_RSP32     |               | <a href="#">Section 18.5.1.12</a> |
| 218h   | SD_RSP54     |               | <a href="#">Section 18.5.1.13</a> |
| 21Ch   | SD_RSP76     |               | <a href="#">Section 18.5.1.14</a> |
| 220h   | SD_DATA      |               | <a href="#">Section 18.5.1.15</a> |
| 224h   | SD_PSTATE    |               | <a href="#">Section 18.5.1.16</a> |
| 228h   | SD_HCTL      |               | <a href="#">Section 18.5.1.17</a> |
| 22Ch   | SD_SYSCTL    |               | <a href="#">Section 18.5.1.18</a> |
| 230h   | SD_STAT      |               | <a href="#">Section 18.5.1.19</a> |
| 234h   | SD_IE        |               | <a href="#">Section 18.5.1.20</a> |
| 238h   | SD_ISE       |               | <a href="#">Section 18.5.1.21</a> |
| 23Ch   | SD_AC12      |               | <a href="#">Section 18.5.1.22</a> |
| 240h   | SD_CAPA      |               | <a href="#">Section 18.5.1.23</a> |
| 248h   | SD_CUR_CAPA  |               | <a href="#">Section 18.5.1.24</a> |
| 250h   | SD_FE        |               | <a href="#">Section 18.5.1.25</a> |
| 254h   | SD_ADMAES    |               | <a href="#">Section 18.5.1.26</a> |
| 258h   | SD ADMASAL   |               | <a href="#">Section 18.5.1.27</a> |
| 25Ch   | SD ADMASAH   |               | <a href="#">Section 18.5.1.28</a> |
| 2FCh   | SD_REV       |               | <a href="#">Section 18.5.1.29</a> |

### 18.5.1.1 SD\_SYSConfig Register (offset = 110h) [reset = 0h]

SD\_SYSConfig is shown in [Figure 18-37](#) and described in [Table 18-20](#).

This register allows controlling various parameters of the OCP interface.

**Figure 18-37. SD\_SYSConfig Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-20. SD\_SYSConfig Register Field Descriptions**

| Bit   | Field         | Type | Reset | Description  |
|-------|---------------|------|-------|--|
| 31-14 | Reserved      | R    | 0h    |  |
| 13-12 | STANDBYMODE   | R/W  | 0h    | Master interface power Management, standby/wait control.<br>The bit field is only useful when generic parameter MADMA_EN (Master ADMA enable) is set as active, otherwise it is a read only register read a 0.<br>0x0 = Force-standby. Mstandby is forced unconditionally.<br>0x1 = No-standby. Mstandby is never asserted.<br>0x2 = Smart-standby modelocal initiator standby status depends on local conditions, i.e. the module's functional requirement from the initiator. IP module shall not generate (initiator-related) wake-up events.<br>0x3 = Smart-Standby wake-up-capable modelocal initiator standby status depends on local conditions, i.e. the module's functional requirement from the initiator. IP module can generate (master-related) wake-up events when in standby state. Mode is only relevant if the appropriate IP module "mwake-up" output is implemented. Functional clock is maintained. Interface clock may be switched off. |
| 11-10 | Reserved      | R    | 0h    |  |
| 9-8   | CLOCKACTIVITY | R/W  | 0h    | Clocks activity during wake up mode period.<br>Bit 8 is the Interface clock.<br>Bit 9 is the Functional clock.<br>0x0 = Interface and Functional clock may be switched off.<br>0x1 = Interface clock is maintained. Functional clock may be switched-off.<br>0x2 = Functional clock is maintained. Interface clock may be switched-off.<br>0x3 = Interface and Functional clocks are maintained.   |
| 7-5   | Reserved      | R    | 0h    |  |

**Table 18-20. SD\_SYSConfig Register Field Descriptions (continued)**

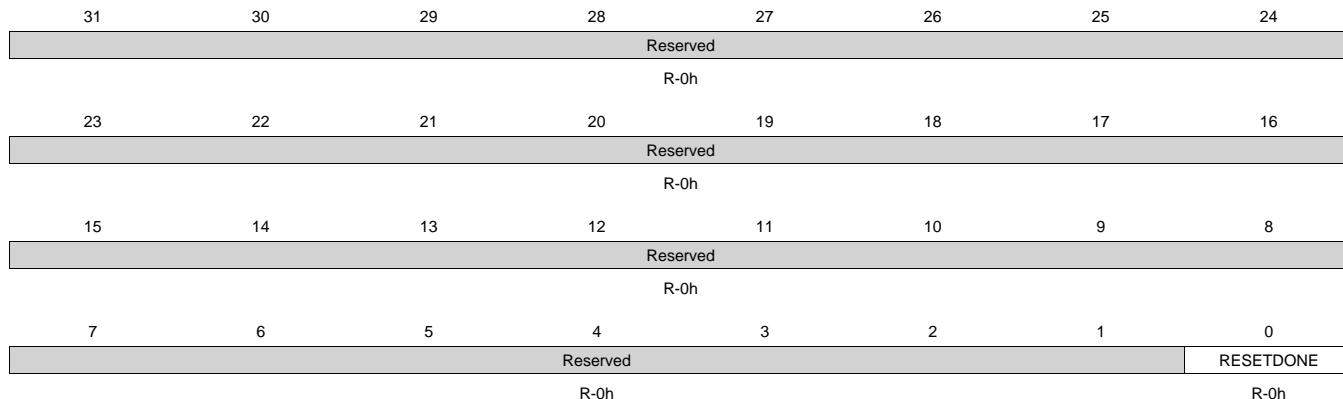
| Bit | Field     | Type | Reset | Description   |
|-----|-----------|------|-------|---|
| 4-3 | SIDLEMODE | R/W  | 0h    | <p>Power management</p> <p>0x0 = If an idle request is detected, the MMC/SD/SDIO host controller acknowledges it unconditionally and goes in Inactive mode. Interrupt and DMA requests are unconditionally deasserted.</p> <p>0x1 = If an idle request is detected, the request is ignored and the module keeps on behaving normally.</p> <p>0x2 = If an idle request is detected, the module will switch to wake up mode based on its internal activity, and the wake up capability can be used if the wake up capability is enabled (bit SD_SYSConfig[2] ENAWAKEUP bit is set to 1).</p> <p>0x3 = Reserved.</p> |
| 2   | ENAWAKEUP | R/W  | 0h    | <p>Wake-up feature control</p> <p>0x0 = Wake-up capability is disabled.</p> <p>0x1 = Wake-up capability is enabled.</p>   |
| 1   | SOFTRESET | R/W  | 0h    | <p>Software reset.</p> <p>The bit is automatically reset by the hardware.</p> <p>During reset, it always returns 0.</p> <p>0x0(W) = No effect</p> <p>0x0(R) = Normal mode</p> <p>0x1(W) = Trigger a module reset.</p> <p>0x1(R) = The module is reset.</p>  |
| 0   | AUTOIDLE  | R/W  | 0h    | <p>Internal Clock gating strategy</p> <p>0x0(R) = Clocks are free-running.</p> <p>0x1(W) = Automatic clock gating strategy is applied, based on the interconnect and MMC interface activity.</p>  |

### 18.5.1.2 SD\_SYSSTATUS Register (offset = 114h) [reset = 0h]

SD\_SYSSTATUS is shown in [Figure 18-38](#) and described in [Table 18-21](#).

This register provides status information about the module excluding the interrupt status information.

**Figure 18-38. SD\_SYSSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-21. SD\_SYSSTATUS Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 31-1 | Reserved  | R    | 0h    |   |
| 0    | RESETDONE | R    | 0h    | Internal Reset Monitoring.<br>Note the debounce clock , the interface clock and the functional clock shall be provided to the MMC/SD/SDIO host controller to allow the internal reset monitoring.<br>0x0 = Internal module reset is on-going<br>0x1 = Reset completed |

### 18.5.1.3 SD\_CSRE Register (offset = 124h) [reset = 0h]

SD\_CSRE is shown in [Figure 18-39](#) and described in [Table 18-22](#).

This register enables the host controller to detect card status errors of response type R1, R1b for all cards and of R5, R5b and R6 response for cards types SD or SDIO. When a bit SD\_CSRE[i] is set to 1, if the corresponding bit at the same position in the response SD\_RSP10[i] is set to 1, the host controller indicates a card error (SD\_STAT[28] CERR bit) interrupt status to avoid the host driver reading the response register (SD\_RSP10). No automatic card error detection for autoCMD12 is implemented; the host system has to check autoCMD12 response register (SD\_RSP76) for possible card errors.

**Figure 18-39. SD\_CSRE Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CSRE   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-22. SD\_CSRE Register Field Descriptions**

| Bit  | Field | Type | Reset | Description                |
|------|-------|------|-------|----------------------------|
| 31-0 | CSRE  | R/W  | 0h    | Card status response error |

#### 18.5.1.4 SD\_SYTEST Register (offset = 128h) [reset = 0h]

SD\_SYTEST is shown in [Figure 18-40](#) and described in [Table 18-23](#).

This register is used to control the signals that connect to I/O pins when the module is configured in system test (SYTEST) mode for boundary connectivity verification. In SYTEST mode, a write into SD\_CMD register will not start a transfer. The buffer behaves as a stack accessible only by the local host (push and pop operations). In this mode, the Transfer Block Size (SD\_BLK[10:0] BLEN bits) and the Blocks count for current transfer (SD\_BLK[31:16] NBLK bits) are needed to generate a Buffer write ready interrupt (SD\_STAT[4] BWR bit) or a Buffer read ready interrupt (SD\_STAT[5] BRR bit) and DMA requests if enabled.

**Figure 18-40. SD\_SYTEST Register**

|          |        |        |        |        |        |        |        |
|----------|--------|--------|--------|--------|--------|--------|--------|
| 31       | 30     | 29     | 28     | 27     | 26     | 25     | 24     |
| Reserved |        |        |        |        |        |        |        |
| R-0h     |        |        |        |        |        |        |        |
| 23       | 22     | 21     | 20     | 19     | 18     | 17     | 16     |
| Reserved |        |        |        |        |        |        |        |
| R-0h     |        |        |        |        |        |        |        |
| 15       | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| SDCD     | SDWP   | WAKD   | SSB    | D7D    | D6D    | D5D    | D4D    |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| D3D      | D2D    | D1D    | D0D    | DDIR   | CDAT   | CDIR   | MCKD   |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-23. SD\_SYTEST Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-17 | Reserved | R    | 0h    |   |
| 16    | OBI      | R/W  | 0h    | Out-of-band interrupt (OBI) data value.<br>0x0 = The out-of-band interrupt pin is driven low.<br>0x1 = The out-of-band interrupt pin is driven high.  |
| 15    | SDCD     | R/W  | 0h    | Card detect input signal (SDCD) data value<br>0x0 = The card detect pin is driven low.<br>0x1 = The card detect pin is driven high.   |
| 14    | SDWP     | R/W  | 0h    | Write protect input signal (SDWP) data value<br>0x0 = The write protect pin SDWP is driven low.<br>0x1 = The write protect pin SDWP is driven high.   |
| 13    | WAKD     | R/W  | 0h    | Wake request output signal data value.<br>0x0(W) = The pin SWAKEUP is driven low.<br>0x0(R) = No action. Returns 0.<br>0x1(W) = The pin SWAKEUP is driven high.<br>0x1(R) = No action. Returns 1.   |
| 12    | SSB      | R/W  | 0h    | Set status bit.<br>This bit must be cleared prior attempting to clear a status bit of the interrupt status register (SD_STAT).<br>0x0(W) = Clear this SSB bit field. Writing 0 does not clear already set status bits.<br>0x0(R) = No action. Returns 0.<br>0x1(W) = Force to 1 all status bits of the interrupt status register (SD_STAT) only if the corresponding bit field in the Interrupt signal enable register (SD_ISE) is set.<br>0x1(R) = No action. Returns 1. |

Table 18-23. SD\_SYSTEST Register Field Descriptions (continued)

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 11  | D7D   | R/W  | 0h    | <p>DAT7 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT7 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT7 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 10  | D6D   | R/W  | 0h    | <p>DAT6 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT6 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT6 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 9   | D5D   | R/W  | 0h    | <p>DAT5 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT5 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT5 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 8   | D4D   | R/W  | 0h    | <p>DAT4 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT4 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT4 line (high) If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |

**Table 18-23. SD\_SYSTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 7   | D3D   | R/W  | 0h    | <p>DAT3 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT3 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT3 line (high). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 6   | D2D   | R/W  | 0h    | <p>DAT2 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT2 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT2 line (high). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 5   | D1D   | R/W  | 0h    | <p>DAT1 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT1 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT1 line (high). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 4   | D0D   | R/W  | 0h    | <p>DAT0 input/output signal data value.</p> <p>0x0(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven low. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x0(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (low). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 0.</p> <p>0x1(W) = If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), the DAT0 line is driven high. If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), no effect.</p> <p>0x1(R) = If SD_SYSTEST[3] DDIR bit = 1 (input mode direction), returns the value on the DAT0 line (high). If SD_SYSTEST[3] DDIR bit = 0 (output mode direction), returns 1.</p> |
| 3   | DDIR  | R/W  | 0h    | <p>Control of the DAT[7:0] pins direction.</p> <p>0x0(W) = The DAT lines are outputs (host to card).</p> <p>0x0(R) = No action. Returns 0.</p> <p>0x1(W) = The DAT lines are inputs (card to host).</p> <p>0x1(R) = No action. Returns 1.</p>   |

**Table 18-23. SD\_SYSTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 2   | CDAT  | R/W  | 0h    | CMD input/output signal data value<br>0x0(W) = If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven low. If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.<br>0x0(R) = If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (low). If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 0 .<br>0x1(W) = If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), the CMD line is driven high. If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), no effect.<br>0x1(R) = If SD_SYSTEST[1] CDIR bit = 1 (input mode direction), returns the value on the CMD line (high) If SD_SYSTEST[1] CDIR bit = 0 (output mode direction), returns 1 . |
| 1   | CDIR  | R/W  | 0h    | Control of the CMD pin direction<br>0x0(W) = The CMD line is an output (host to card).<br>0x0(R) = No action. Returns 0.<br>0x1(W) = The CMD line is an input (card to host) .<br>0x1(R) = No action. Returns 1.  |
| 0   | MCKD  | R/W  | 0h    | MMC clock output signal data value<br>0x0(W) = The output clock is driven low.<br>0x0(R) = No action. Returns 0.<br>0x1(W) = The output clock is driven high.<br>0x1(R) = No action. Returns 1.   |

### 18.5.1.5 SD\_CON Register (offset = 12Ch) [reset = 0h]

SD\_CON is shown in [Figure 18-41](#) and described in [Table 18-24](#).

This register is used: To select the functional mode for any card. To send an initialization sequence to any card. To send an initialization sequence to any card. To enable the detection on the mmc\_dat[1] signal of a card interrupt for SDIO cards only. It also configures the parameters related to the card detect and write protect input signals

**Figure 18-41. SD\_CON Register**

| 31       | 30 | 29       | 28      | 27     | 26       | 25       | 24         |
|----------|----|----------|---------|--------|----------|----------|------------|
| Reserved |    |          |         |        |          |          |            |
| R-0h     |    |          |         |        |          |          |            |
| 23       | 22 | 21       | 20      | 19     | 18       | 17       | 16         |
| Reserved |    | SDMA_LnE | DMA_MnS | DDR    | BOOT_CF0 | BOOT_ACK | CLKEXTFREE |
| R-0h     |    | R/W-0h   | R/W-0h  | R/W-0h | R/W-0h   | R/W-0h   | R/W-0h     |
| 15       | 14 | 13       | 12      | 11     | 10       | 9        | 8          |
| PADEN    |    | Reserved | CEATA   | CTPL   |          | DVAL     | WPP        |
| R/W-0h   |    | R-0h     | R/W-0h  | R/W-0h |          | R/W-0h   | R/W-0h     |
| 7        | 6  | 5        | 4       | 3      | 2        | 1        | 0          |
| CDP      |    | MIT      | DW8     | MODE   | STR      | HR       | INIT       |
| R/W-0h   |    | R/W-0h   | R/W-0h  | R/W-0h | R/W-0h   | R/W-0h   | R/W-0h     |
|          |    |          |         |        |          |          |            |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-24. SD\_CON Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-22 | Reserved | R    | 0h    |   |
| 21    | SDMA_LnE | R/W  | 0h    | Slave DMA Level/Edge Request.<br>The waveform of the DMA request can be configured either edge sensitive with early de-assertion on first access to SD_DATA register or late de-assertion, request remains active until last allowed data written into SD_DATA.<br>0x0 = Slave DMA edge sensitive.<br>0x1 = Slave DMA level sensitive.  |
| 20    | DMA_MnS  | R/W  | 0h    | DMA Master or Slave selection.<br>When this bit is set and the controller is configured to use the DMA, Ocp master interface is used to get datas from system using ADMA2 procedure (direct access to the memory).<br>This option is only available if generic parameter MADMA_EN is asserted to 1.<br>0x0 = The controller is slave on data transfers with system.<br>0x1 = Not available on this device.  |
| 19    | DDR      | R/W  | 0h    | Dual Data Rate mode.<br>When this register is set, the controller uses both clock edge to emit or receive data.<br>Odd bytes are transmitted on falling edges and even bytes are transmitted on rise edges.<br>It only applies on Data bytes and CRC, Start, end bits and CRC status are kept full cycle.<br>This bit field is only meaningful and active for even clock divider ratio of SD_SYSCTL[CLKD], it is insensitive to SD_HCTL[HSPE] setting.<br>Note: DDR mode is not supported on AM335x.<br>Always set this bit to 0.<br>0x0 = Standard modeData are transmitted on a single edge.<br>0x1 = Data Bytes and CRC are transmitted on both edges. |

**Table 18-24. SD\_CON Register Field Descriptions (continued)**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 18    | BOOT_CF0   | R/W  | 0h    | <p>Boot Status Supported.<br/> This register is set when the CMD line needs to be forced to 0 for a boot sequence.<br/> CMD line is driven to 0 after writing in SD_CMD.<br/> The line is released when this bit field is de-asserted and aborts data transfer in case of a pending transaction.</p> <p>0x0(W) = CMD line forced to 0 is enabled.<br/> 0x0(R) = CMD line not forced.<br/> 0x1(W) = CMD line forced to 0 is enabled and will be active after writing into SD_CMD register.<br/> 0x1(R) = CMD line is released when it was previously forced to 0 by a boot sequence.</p>   |
| 17    | BOOT_ACK   | R/W  | 0h    | <p>Boot acknowledge received.<br/> When this bit is set the controller should receive a boot status on DAT0 line after next command issued.<br/> If no status is received a data timeout will be generated.<br/> 0x0 = No acknowledge to be received.<br/> 0x1 = A boot status will be received on DAT0 line after issuing a command.</p>   |
| 16    | CLKEXTFREE | R/W  | 0h    | <p>External clock free running.<br/> This register is used to maintain card clock out of transfer transaction to enable slave module (for example to generate a synchronous interrupt on mmc_dat[1]).<br/> The Clock will be maintained only if SD_SYSCTL[2] CEN bit is set.<br/> 0x0 = External card clock is cut off outside active transaction period.<br/> 0x1 = External card clock is maintained even out of active transaction period only if SD_SYSCTL[2] CEN bit is set.</p>   |
| 15    | PADEN      | R/W  | 0h    | <p>Control power for MMC lines.<br/> This register is only useful when MMC PADs contain power saving mechanism to minimize its leakage power.<br/> It works as a GPIO that directly controls the ACTIVE pin of PADs. Excepted for mmc_dat[1], the signal is also combined outside the module with the dedicated power control SD_CON[11] CTPL bit.<br/> 0x0 = ADPIDE module pin is not forced, it is automatically generated by the MMC fsm.<br/> 0x1 = ADPIDE module pin is forced to active state</p>   |
| 14-13 | Reserved   | R    | 0h    |   |
| 12    | CEATA      | R/W  | 0h    | <p>CE-ATA control mode (MMC cards compliant with CE-ATA).<br/> This bit selects the active level of the out-of-band interrupt coming from MMC cards.<br/> The usage of the Out-of-Band signal (OBI) is not supported.<br/> 0x0 = Standard MMC/SD/SDIO mode.<br/> 0x1 = CE-ATA mode. Next commands are considered as CE-ATA commands.</p>  |
| 11    | CTPL       | R/W  | 0h    | <p>Control Power for mmc_dat[1] line (SD cards).<br/> By default, this bit is cleared to 0 and the host controller automatically disables all the input buffers outside of a transaction to minimize the leakage current.<br/> SDIO cards.<br/> When this bit is set to 1, the host controller automatically disables all the input buffers except the buffer of mmc_dat[1] outside of a transaction in order to detect asynchronous card interrupt on mmc_dat[1] line and minimize the leakage current of the buffers.<br/> 0x0 = Disable all the input buffers outside of a transaction.<br/> 0x1 = Disable all the input buffers except the buffer of mmc_dat[1] outside of a transaction.</p> |

**Table 18-24. SD\_CON Register Field Descriptions (continued)**

| Bit  | Field | Type | Reset | Description  |
|------|-------|------|-------|--|
| 10-9 | DVAL  | R/W  | 0h    | Debounce filter value (all cards).<br>This register is used to define a debounce period to filter the card detect input signal (SDCD).<br>The usage of the card detect input signal (SDCD) is optional and depends on the system integration and the type of the connector housing that accommodates the card.<br>0x0 = 33 us debounce period<br>0x1 = 231 us debounce period<br>0x2 = 1 ms debounce period<br>0x3 = 8.4 ms debounce period  |
| 8    | WPP   | R/W  | 0h    | Write protect polarity (SD and SDIO cards only).<br>This bit selects the active level of the write protect input signal (SDWP).<br>The usage of the write protect input signal (SDWP) is optional and depends on the system integration and the type of the connector housing that accommodates the card.<br>0x0 = Active high level<br>0x1 = Active low level   |
| 7    | CDP   | R/W  | 0h    | Card detect polarity (all cards).<br>This bit selects the active level of the write protect input signal (SDWP).<br>The usage of the write protect input signal (SDWP) is optional and depends on the system integration and the type of the connector housing that accommodates the card.<br>0x0 = Active high level<br>0x1 = Active low level  |
| 6    | MIT   | R/W  | 0h    | MMC interrupt command (MMC cards only).<br>This bit must be set to 1, when the next write access to the command register (SD_CMD) is for writing a MMC interrupt command (CMD40) requiring the command timeout detection to be disabled for the command response.<br>0x0 = Command timeout enabled.<br>0x1 = Command timeout disabled.   |
| 5    | DW8   | R/W  | 0h    | 8-bit mode MMC select (MMC cards only).<br>For SD/SDIO cards, this bit must be cleared to 0.<br>For MMC card, this bit must be set following a valid SWITCH command (CMD6) with the correct value and extend CSD index written in the argument.<br>Prior to this command, the MMC card configuration register (CSD and EXT_CSD) must be verified for compliancy with MMC standard specification.<br>0x0 = 1-bit or 4-bit data width<br>0x1 = 8-bit data width  |
| 4    | MODE  | R/W  | 0h    | Mode select (all cards).<br>This bit selects the functional mode.<br>0x0 = Functional mode. Transfers to the MMC/SD/SDIO cards follow the card protocol. The MMC clock is enabled. MMC/SD transfers are operated under the control of the SD_CMD register.<br>0x1 = SYSTEST mode. SYSTEST mode. The signal pins are configured as general-purpose input/output and the 1024-byte buffer is configured as a stack memory accessible only by the local host or system DMA. The pins retain their default type (input, output or in-out). SYSTEST mode is operated under the control of the SYSTEST register. |

**Table 18-24. SD\_CON Register Field Descriptions (continued)**

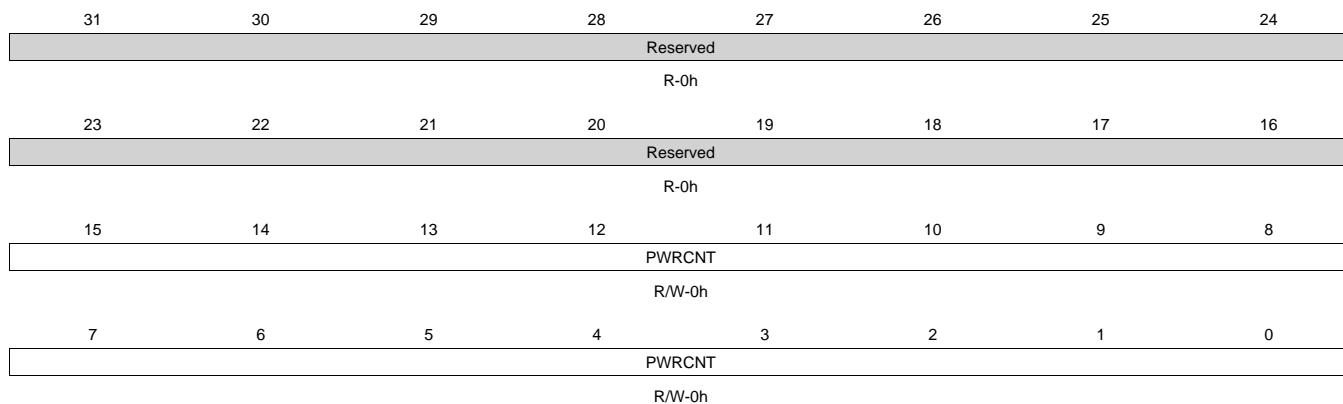
| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 3   | STR   | R/W  | 0h    | <p>Stream command (MMC cards only).<br/> This bit must be set to 1 only for the stream data transfers (read or write) of the adtc commands.<br/> Stream read is a class 1 command (CMD11READ_DAT_UNTIL_STOP).<br/> Stream write is a class 3 command (CMD20WRITE_DAT_UNTIL_STOP).<br/> 0x0 = Block oriented data transfer<br/> 0x1 = Stream oriented data transfer</p>  |
| 2   | HR    | R/W  | 0h    | <p>Broadcast host response (MMC cards only).<br/> This register is used to force the host to generate a 48-bit response for bc command type.<br/> It can be used to terminate the interrupt mode by generating a CMD40 response by the core.<br/> In order to have the host response to be generated in open drain mode, the register SD_CON[OD] must be set to 1.<br/> When SD_CON[12] CEATA bit is set to 1 and SD_ARG cleared to 0, when writing 0000 0000h into SD_CMD register, the host controller performs a 'command completion signal disable' token (i.e., mmc_cmd line held to 0 during 47 cycles followed by a 1).<br/> 0x0 = The host does not generate a 48-bit response instead of a command.<br/> 0x1 = The host generates a 48-bit response instead of a command or a command completion signal disable token.</p>                           |
| 1   | INIT  | R/W  | 0h    | <p>Send initialization stream (all cards).<br/> When this bit is set to 1, and the card is idle, an initialization sequence is sent to the card.<br/> An initialization sequence consists of setting the mmc_cmd line to 1 during 80 clock cycles.<br/> The initialization sequence is mandatory - but it is not required to do it through this bit - this bit makes it easier.<br/> Clock divider (SD_SYSCTL [15:6] CLKD bits) should be set to ensure that 80 clock periods are greater than 1ms.<br/> Note: In this mode, there is no command sent to the card and no response is expected.<br/> A command complete interrupt will be generated once the initialization sequence is completed.<br/> SD_STAT[0] CC bit can be polled.<br/> 0x0 = The host does not send an initialization sequence<br/> 0x1 = The host sends an initialization sequence</p> |
| 0   | OD    | R/W  | 0h    | <p>Card open drain mode (MMC cards only).<br/> This bit must be set to 1 for MMC card commands 1, 2, 3 and 40, and if the MMC card bus is operating in open-drain mode during the response phase to the command sent.<br/> Typically, during card identification mode when the card is either in idle, ready or ident state.<br/> It is also necessary to set this bit to 1, for a broadcast host response (see Broadcast host response register SD_CON[2] HR bit).<br/> 0x0 = No open drain<br/> 0x1 = Open drain or broadcast host response</p>   |

### 18.5.1.6 SD\_PWCNT Register (offset = 130h) [reset = 0h]

SD\_PWCNT is shown in [Figure 18-42](#) and described in [Table 18-25](#).

This register is used to program a mmc counter to delay command transfers after activating the PAD power, this value depends on PAD characteristics and voltage.

**Figure 18-42. SD\_PWCNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-25. SD\_PWCNT Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-16 | Reserved | R    | 0h    |   |
| 15-0  | PWRcnt   | R/W  | 0h    | <p>Power counter register.<br/>           This register is used to introduce a delay between the PAD ACTIVE pin assertion and the command issued.</p> <p>0x0 = No additional delay added<br/>           0x1 = TCF delay (card clock period)<br/>           0x2 = TCF x 2 delay (card clock period)<br/>           0xffffe = TCF x 65534 delay (card clock period)<br/>           0xffff = TCF x 65535 delay (card clock period)</p> |

### 18.5.1.7 SD\_SDMASA Register (offset = 200h) [reset = 0h]

SD\_SDMASA is shown in [Figure 18-43](#) and described in [Table 18-26](#).

This register is used to program a mmc counter to delay command transfers after activating the PAD power. This value depends on PAD characteristics and voltage.

**Figure 18-43. SD\_SDMASA Register**

| 31           | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| SDMA_SYSADDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-26. SD\_SDMASA Register Field Descriptions**

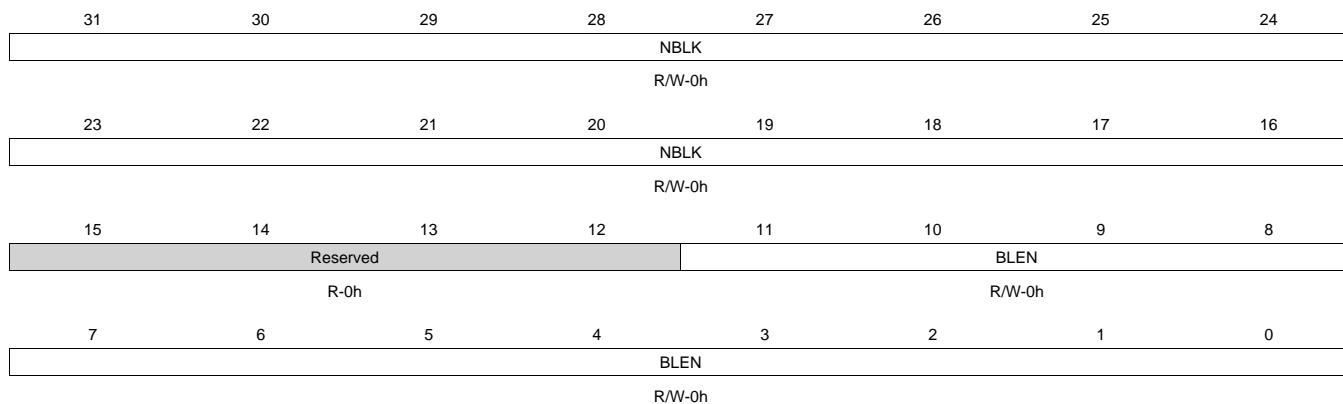
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-0 | SDMA_SYSADDR | R    | 0h    | <p>This register contains the system memory address for a SDMA transfer.</p> <p>When the Host Controller stops a SDMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing (i.e., after a transaction has stopped).</p> <p>Read operations during transfers may return an invalid value.</p> <p>The Host Driver shall initialize this register before starting a SDMA transaction.</p> <p>After SDMA has stopped, the next system address of the next contiguous data position can be read from this register.</p> <p>The SDMA transfer waits at the every boundary specified by the Host SDMA Buffer Boundary in the Block Size register.</p> <p>The Host Controller generates DMA Interrupt to request the Host Driver to update this register.</p> <p>The Host Driver sets the next system address of the next data position to this register.</p> <p>When the most upper byte of this register (003h) is written, the Host Controller restarts the SDMA transfer.</p> <p>When restarting SDMA by the Resume command or by setting Continue Request in the Block Gap Control register, the Host Controller shall start at the next contiguous address stored here in the SDMA System Address register.</p> <p>ADMA does not use this register.</p> |

### 18.5.1.8 SD\_BLK Register (offset = 204h) [reset = 0h]

SD\_BLK is shown in [Figure 18-44](#) and described in [Table 18-27](#).

This register shall be used for any card. SD\_BLK[BLEN] is the block size register. SD\_BLK[NBLK] is the block count register.

**Figure 18-44. SD\_BLOCK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-27. SD\_BLOCK Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | NBLK     | R/W  | 0h    | <p>Blocks count for current transfer.<br/>           This register is enabled when Block count Enable (SD_CMD[1] BCE bit) is set to 1 and is valid only for multiple block transfers.<br/>           Setting the block count to 0 results no data blocks being transferred.<br/>           Note: The host controller decrements the block count after each block transfer and stops when the count reaches zero.<br/>           This register can be accessed only if no transaction is executing (i.e., after a transaction has stopped).<br/>           Read operations during transfers may return an invalid value and write operation will be ignored.<br/>           In suspend context, the number of blocks yet to be transferred can be determined by reading this register.<br/>           When restoring transfer context prior to issuing a Resume command, The local host shall restore the previously saved block count.</p> <p>0x0 = Stop count<br/>           0x1 = 1 block<br/>           0x2 = 2 blocks<br/>           0xffff = 65535 blocks</p> |
| 15-12 | Reserved | R    | 0h    |  |
| 11-0  | BLEN     | R/W  | 0h    | <p>Transfer block size.<br/>           This register specifies the block size for block data transfers.<br/>           Read operations during transfers may return an invalid value, and write operations are ignored.<br/>           When a CMD12 command is issued to stop the transfer, a read of the BLEN field after transfer completion (SD_STAT[1] TC bit set to 1) will not return the true byte number of data length while the stop occurs but the value written in this register before transfer is launched.</p> <p>0x0 = No data transfer<br/>           0x1 = 1 byte block length<br/>           0x2 = 2 bytes block length<br/>           0x3 = 3 bytes block length<br/>           0x1ff = 511 bytes block length<br/>           0x200 = 512 bytes block length<br/>           0x7ff = 2047 bytes block length<br/>           0x800 = 2048 bytes block length</p>  |

### 18.5.1.9 SD\_ARG Register (offset = 208h) [reset = 0h]

SD\_ARG is shown in [Figure 18-45](#) and described in [Table 18-28](#).

This register contains command argument specified as bit 39-8 of Command-Format. These registers must be initialized prior to sending the command itself to the card (write action into the register SD\_CMD register). Only exception is for a command index specifying stuff bits in arguments, making a write unnecessary.

**Figure 18-45. SD\_ARG Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ARG    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-28. SD\_ARG Register Field Descriptions**

| Bit  | Field | Type | Reset | Description                    |
|------|-------|------|-------|--------------------------------|
| 31-0 | ARG   | R/W  | 0h    | Command argument bits [31:0] . |

### 18.5.1.10 SD\_CMD Register (offset = 20Ch) [reset = 0h]

SD\_CMD is shown in [Figure 18-46](#) and described in [Table 18-29](#).

SD\_CMD[31:16] = the command register. SD\_CMD[15:0] = the transfer mode. This register configures the data and command transfers. A write into the most significant byte send the command. A write into SD\_CMD[15:0] registers during data transfer has no effect. This register can be used for any card. In SYSTEST mode, a write into SD\_CMD register will not start a transfer.

**Figure 18-46. SD\_CMD Register**

| 31       | 30     | 29     | 28       | 27       | 26       | 25     | 24     |
|----------|--------|--------|----------|----------|----------|--------|--------|
| Reserved |        | INDX   |          |          |          |        |        |
| R-0h     |        | R/W-0h |          |          |          |        |        |
| 23       | 22     | 21     | 20       | 19       | 18       | 17     | 16     |
| CMD_TYPE | DP     | CICE   | CCCE     | Reserved | RSP_TYPE |        |        |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h   | R-0h     | R/W-0h   |        |        |
| 15       | 14     | 13     | 12       | 11       | 10       | 9      | 8      |
| Reserved |        | R-0h   |          |          |          |        |        |
| 7        | 6      | 5      | 4        | 3        | 2        | 1      | 0      |
| Reserved | MSBS   | DDIR   | Reserved | ACEN     | BCE      | DE     |        |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h   | R/W-0h   | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-29. SD\_CMD Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description |
|-------|----------|------|-------|-------------|
| 31-30 | Reserved | R    | 0h    |             |

**Table 18-29. SD\_CMD Register Field Descriptions (continued)**

| Bit   | Field | Type | Reset | Description   |
|-------|-------|------|-------|---|
| 29-24 | INDX  | R/W  | 0h    | <p>Command index binary encoded value from 0 to 63 specifying the command number send to card.</p> <p>0x0 = CMD0 or ACMD0<br/>           0x1 = CMD1 or ACMD1<br/>           0x2 = CMD2 or ACMD2<br/>           0x3 = CMD3 or ACMD3<br/>           0x4 = CMD4 or ACMD4<br/>           0x5 = CMD5 or ACMD5<br/>           0x6 = CMD6 or ACMD6<br/>           0x7 = CMD7 or ACMD7<br/>           0x8 = CMD8 or ACMD8<br/>           0x9 = CMD9 or ACMD9<br/>           0xa = CMD10 or ACMD10<br/>           0xb = CMD11 or ACMD11<br/>           0xc = CMD12 or ACMD12<br/>           0xd = CMD13 or ACMD13<br/>           0xe = CMD14 or ACMD14<br/>           0xf = CMD15 or ACMD15<br/>           0x10 = CMD16 or ACMD16<br/>           0x11 = CMD17 or ACMD17<br/>           0x12 = CMD18 or ACMD18<br/>           0x13 = CMD19 or ACMD19<br/>           0x14 = CMD20 or ACMD20<br/>           0x15 = CMD21 or ACMD21<br/>           0x16 = CMD22 or ACMD22<br/>           0x17 = CMD23 or ACMD23<br/>           0x18 = CMD24 or ACMD24<br/>           0x19 = CMD25 or ACMD25<br/>           0x1a = CMD26 or ACMD26<br/>           0x1b = CMD27 or ACMD27<br/>           0x1c = CMD28 or ACMD28<br/>           0x1d = CMD29 or ACMD29<br/>           0x1e = CMD30 or ACMD30<br/>           0x1f = CMD31 or ACMD31<br/>           0x20 = CMD32 or ACMD32<br/>           0x21 = CMD33 or ACMD33<br/>           0x22 = CMD34 or ACMD34<br/>           0x23 = CMD35 or ACMD35<br/>           0x24 = CMD36 or ACMD36<br/>           0x25 = CMD37 or ACMD37<br/>           0x26 = CMD38 or ACMD38<br/>           0x27 = CMD39 or ACMD39<br/>           0x28 = CMD40 or ACMD40<br/>           0x29 = CMD41 or ACMD41<br/>           0x2a = CMD42 or ACMD42<br/>           0x2b = CMD43 or ACMD43<br/>           0x2c = CMD44 or ACMD44<br/>           0x2d = CMD45 or ACMD45<br/>           0x2e = CMD46 or ACMD46<br/>           0x2f = CMD47 or ACMD47<br/>           0x30 = CMD48 or ACMD48<br/>           0x31 = CMD49 or ACMD49<br/>           0x32 = CMD50 or ACMD5<br/>           0x33 = CMD51 or ACMD5<br/>           0x34 = CMD52 or ACMD5</p> |

**Table 18-29. SD\_CMD Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
|       |          |      |       | 0x35 = CMD53 or ACMD5<br>0x36 = CMD54 or ACMD5<br>0x37 = CMD55 or ACMD5<br>0x38 = CMD56 or ACMD5<br>0x39 = CMD57 or ACMD5<br>0x3a = CMD58 or ACMD5<br>0x3b = CMD59 or ACMD5<br>0x3c = CMD60 or ACMD60<br>0x3d = CMD61 or ACMD61<br>0x3e = CMD62 or ACMD62<br>0x3f = CMD63 or ACMD63  |
| 23-22 | CMD_TYPE | R/W  | 0h    | Command type.<br>This register specifies three types of special commands: Suspend, Resume and Abort.<br>These bits shall be cleared to 0b00 for all other commands.<br>0x0 = Others commands<br>0x1 = Upon CMD52 "Bus Suspend" operation<br>0x2 = Upon CMD52 "Function Select" operation<br>0x3 = Upon CMD12 or CMD52 "I/O Abort" command  |
| 21    | DP       | R/W  | 0h    | Data present select.<br>This register indicates that data is present and mmc_dat line shall be used.<br>It must be cleared to 0 in the following conditions: Command using only mmc_cmd line.<br>Command with no data transfer but using busy signal on mmc_dat0.<br>Resume command.<br>0x0 = Command with no data transfer<br>0x1 = Command with data transfer  |
| 20    | CICE     | R/W  | 0h    | Command Index check enable.<br>This bit must be set to 1 to enable index check on command response to compare the index field in the response against the index of the command.<br>If the index is not the same in the response as in the command, it is reported as a command index error (SD_STAT[19] CIE bit set to1)<br>NoteThe CICE bit cannot be configured for an Auto CMD12, then index check is automatically checked when this command is issued.<br>0x0 = Index check disable<br>0x1 = Index check enable |
| 19    | CCCE     | R/W  | 0h    | Command CRC check enable.<br>This bit must be set to 1 to enable CRC7 check on command response to protect the response against transmission errors on the bus.<br>If an error is detected, it is reported as a command CRC error (SD_STAT[17] CCRC bit set to 1).<br>NoteThe CCCE bit cannot be configured for an Auto CMD12, and then CRC check is automatically checked when this command is issued.<br>0x0 = CRC7 check disable<br>0x1 = CRC7 check enable   |
| 18    | Reserved | R    | 0h    |  |
| 17-16 | RSP_TYPE | R/W  | 0h    | Response type.<br>This bits defines the response type of the command.<br>0x0 = No response<br>0x1 = Response Length 136 bits<br>0x2 = Response Length 48 bits<br>0x3 = Response Length 48 bits with busy after response  |
| 15-6  | Reserved | R    | 0h    |  |

**Table 18-29. SD\_CMD Register Field Descriptions (continued)**

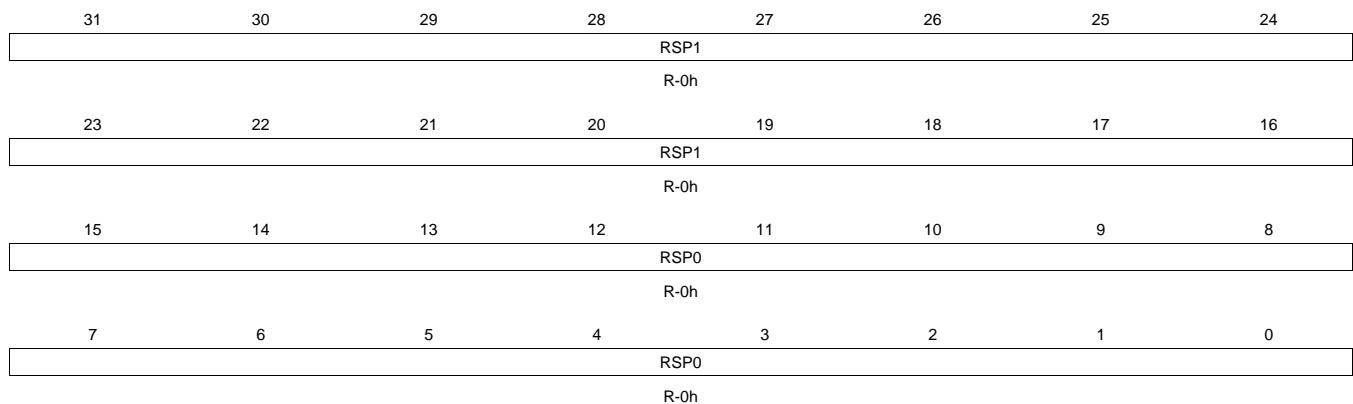
| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 5   | MSBS     | R/W  | 0h    | Multi/Single block select.<br>This bit must be set to 1 for data transfer in case of multi block command.<br>For any others command this bit shall be cleared to 0.<br>0x0 = Single block. If this bit is 0, it is not necessary to set the register SD_BLK[31:16] NBLK bits.<br>0x1 = Multi block. When Block Count is disabled (SD_CMD[1] BCE bit is cleared to 0) in Multiple block transfers (SD_CMD[5] MSBS bit is set to 1), the module can perform infinite transfer.   |
| 4   | DDIR     | R/W  | 0h    | Data transfer Direction.<br>Select This bit defines either data transfer will be a read or a write.<br>0x0 = Data Write (host to card)<br>0x1 = Data Read (card to host)   |
| 3   | Reserved | R    | 0h    |  |
| 2   | ACEN     | R/W  | 0h    | Auto CMD12 Enable (SD cards only).<br>When this bit is set to 1, the host controller issues a CMD12 automatically after the transfer completion of the last block.<br>The Host Driver shall not set this bit to issue commands that do not require CMD12 to stop data transfer.<br>In particular, secure commands do not require CMD12.<br>For CE-ATA commands (SD_CON[12] CEATA bit set to 1), auto CMD12 is useless<br>therefore when this bit is set the mechanism to detect command completion signal, named CCS, interrupt is activated.<br>0x0 = Auto CMD12 disable<br>0x1 = Auto CMD12 enable or CCS detection enabled. |
| 1   | BCE      | R/W  | 0h    | Block Count Enable (Multiple block transfers only).<br>This bit is used to enable the block count register (SD_BLK [31:16] NBLK bits).<br>When Block Count is disabled (SD_CMD[1] BCE bit is cleared to 0) in Multiple block transfers (SD_CMD[5] MSBS bits is set to 1), the module can perform infinite transfer.<br>0x0 = Block count disabled for infinite transfer.<br>0x1 = Block count enabled for multiple block transfer with known number of blocks  |
| 0   | DE       | R/W  | 0h    | DMA Enable.<br>This bit is used to enable DMA mode for host data access.<br>0x0 = DMA mode disable<br>0x1 = DMA mode enable  |

### 18.5.1.11 SD\_RSP10 Register (offset = 210h) [reset = 0h]

SD\_RSP10 is shown in [Figure 18-47](#) and described in [Table 18-30](#).

This 32-bit register holds bits positions [31:0] of command response type R1, R1b, R2, R3, R4, R5, R5b, or R6.

**Figure 18-47. SD\_RSP10 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-30. SD\_RSP10 Register Field Descriptions**

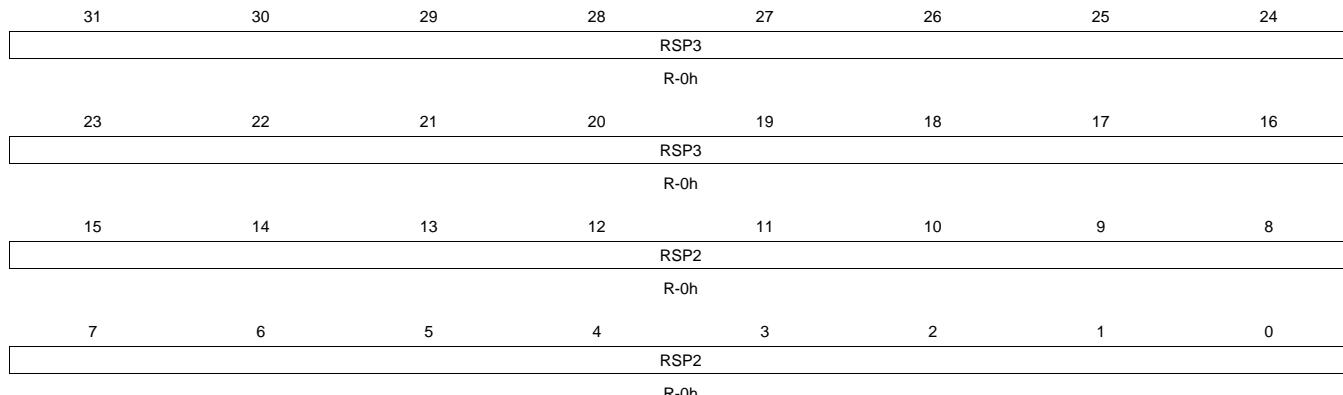
| Bit   | Field | Type | Reset | Description              |
|-------|-------|------|-------|--------------------------|
| 31-16 | RSP1  | R    | 0h    | Command Response [31:16] |
| 15-0  | RSP0  | R    | 0h    | Command Response [15:0]  |

### 18.5.1.12 SD\_RSP32 Register (offset = 214h) [reset = 0h]

SD\_RSP32 is shown in [Figure 18-48](#) and described in [Table 18-31](#).

This 32-bit register holds bits positions [63:32] of command response type R2.

**Figure 18-48. SD\_RSP32 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-31. SD\_RSP32 Register Field Descriptions**

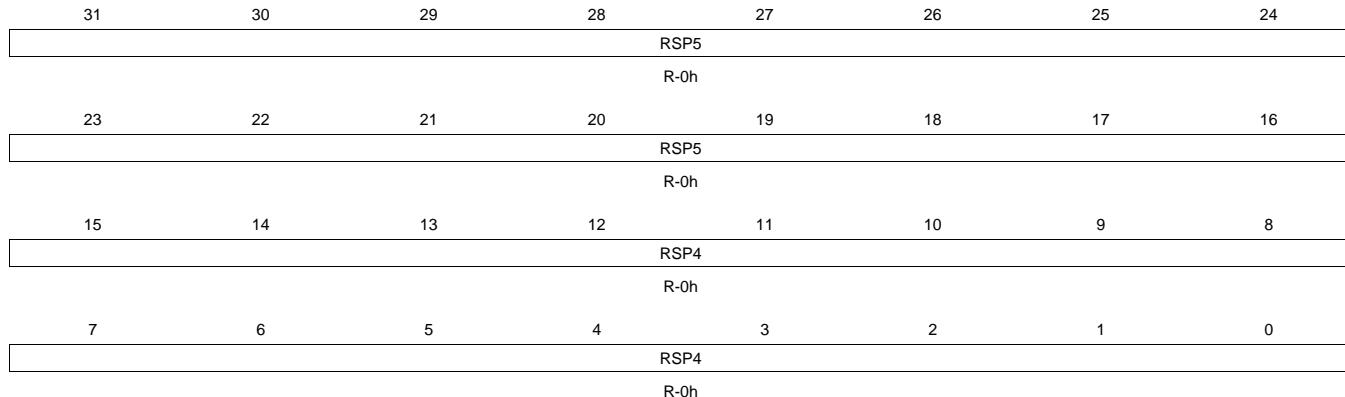
| Bit   | Field | Type | Reset | Description              |
|-------|-------|------|-------|--------------------------|
| 31-16 | RSP3  | R    | 0h    | Command Response [63:48] |
| 15-0  | RSP2  | R    | 0h    | Command Response [47:32] |

### 18.5.1.13 SD\_RSP54 Register (offset = 218h) [reset = 0h]

SD\_RSP54 is shown in [Figure 18-49](#) and described in [Table 18-32](#).

This 32-bit register holds bits positions [95:64] of command response type R2.

**Figure 18-49. SD\_RSP54 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-32. SD\_RSP54 Register Field Descriptions**

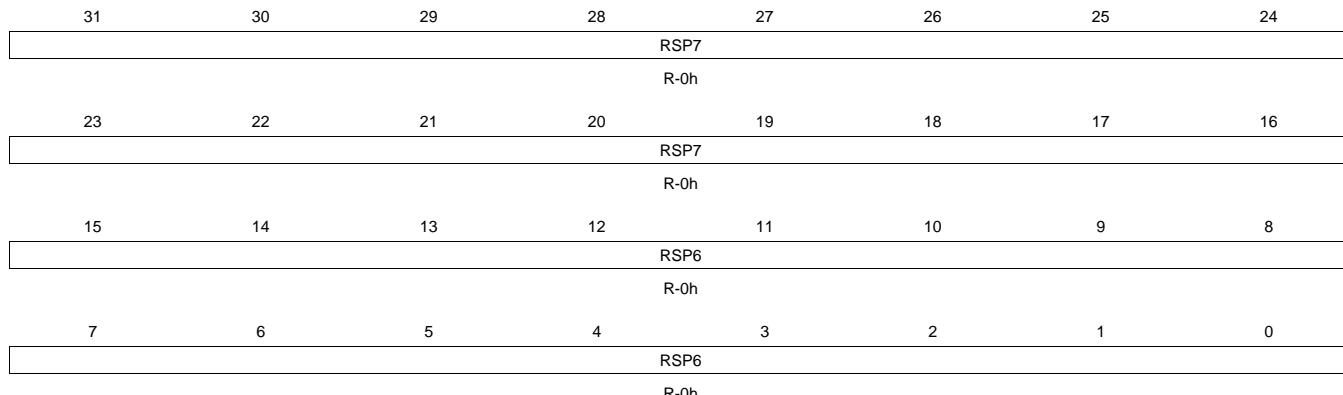
| Bit   | Field | Type | Reset | Description              |
|-------|-------|------|-------|--------------------------|
| 31-16 | RSP5  | R    | 0h    | Command Response [95:80] |
| 15-0  | RSP4  | R    | 0h    | Command Response [79:64] |

#### **18.5.1.14 SD\_RSP76 Register (offset = 21Ch) [reset = 0h]**

SD\_RSP76 is shown in [Figure 18-50](#) and described in [Table 18-33](#).

This 32-bit register holds bits positions [127:96] of command response type R2.

**Figure 18-50. SD\_RSP76 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-33. SD\_RSP76 Register Field Descriptions**

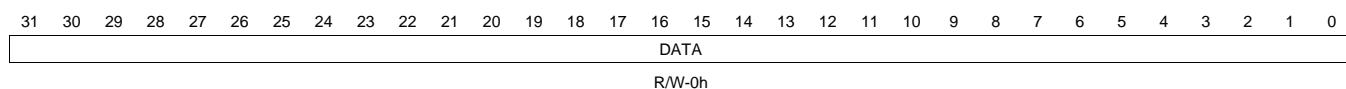
| Bit   | Field | Type | Reset | Description                |
|-------|-------|------|-------|----------------------------|
| 31-16 | RSP7  | R    | 0h    | Command Response [127:112] |
| 15-0  | RSP6  | R    | 0h    | Command Response [111:96]  |

### 18.5.1.15 SD\_DATA Register (offset = 220h) [reset = 0h]

SD\_DATA is shown in [Figure 18-51](#) and described in [Table 18-34](#).

This register is the 32-bit entry point of the buffer for read or write data transfers. The buffer size is 32bitsx256(1024 bytes). Bytes within a word are stored and read in little endian format. This buffer can be used as two 512 byte buffers to transfer data efficiently without reducing the throughput. Sequential and contiguous access is necessary to increment the pointer correctly. Random or skipped access is not allowed. In little endian, if the local host accesses this register byte-wise or 16bit-wise, the least significant byte (bits [7:0]) must always be written/read first. The update of the buffer address is done on the most significant byte write for full 32-bit DATA register or on the most significant byte of the last word of block transfer. Example 1Byte or 16-bit access: Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1100 (2-bytes) OK. Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=0100 (1-byte) OK. Mbyteen[3:0]=0001 (1-byte) => Mbyteen[3:0]=0010 (1-byte) => Mbyteen[3:0]=1000 (1-byte) Bad.

**Figure 18-51. SD\_DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-34. SD\_DATA Register Field Descriptions**

| Bit  | Field | Type | Reset | Description   |
|------|-------|------|-------|---|
| 31-0 | DATA  | R/W  | 0h    | Data register [31:0].<br>In functional mode (SD_CON[4] MODE bit set to the default value 0): A read access to this register is allowed only when the buffer read enable status is set to 1 (SD_PSTATE[11] BRE bit), otherwise a bad access (SD_STAT[29] BADA bit) is signaled.<br>A write access to this register is allowed only when the buffer write enable status is set to 1 (SD_PSTATE[10] BWE bit), otherwise a bad access (SD_STAT[29] BADA bit) is signaled and the data is not written. |

### 18.5.1.16 SD\_PSTATE Register (offset = 224h) [reset = 0h]

SD\_PSTATE is shown in [Figure 18-52](#) and described in [Table 18-35](#).

The Host can get the status of the Host controller from this 32-bit read only register.

**Figure 18-52. SD\_PSTATE Register**

|          |    |    |    |      |      |      |      |
|----------|----|----|----|------|------|------|------|
| 31       | 30 | 29 | 28 | 27   | 26   | 25   | 24   |
| Reserved |    |    |    |      |      |      | CLEV |
| R-0h     |    |    |    |      |      |      | R-0h |
| 23       | 22 | 21 | 20 | 19   | 18   | 17   | 16   |
| DLEV     |    |    |    | WP   | CDPL | CSS  | CINS |
| R-0h     |    |    |    | R-0h | R-0h | R-0h | R-0h |
| 15       | 14 | 13 | 12 | 11   | 10   | 9    | 8    |
| Reserved |    |    |    | BRE  | BWE  | RTA  | WTA  |
| R-0h     |    |    |    | R-0h | R-0h | R-0h | R-0h |
| 7        | 6  | 5  | 4  | 3    | 2    | 1    | 0    |
| Reserved |    |    |    | DLA  | DATI | CMDI |      |
| R-0h     |    |    |    | R-0h | R-0h | R-0h |      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-35. SD\_PSTATE Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-25 | Reserved | R    | 0h    |   |
| 24    | CLEV     | R    | 0h    | mmc_cmd line signal level.<br>This status is used to check the mmc_cmd line level to recover from errors, and for debugging.<br>The value of this register after reset depends on the mmc_cmd line level at that time.<br>0x0 = The mmc_cmd line level is 0.<br>0x1 = The mmc_cmd line level is 1.  |
| 23-20 | DLEV     | R    | 0h    | mmc_dat [3:0] line signal level mmc_dat3 equal to or greater than bit 23.<br>mmc_dat2 equal to or greater than bit 22.<br>mmc_dat1 equal to or greater than bit 21.<br>mmc_dat0 equal to or greater than bit 20.<br>This status is used to check mmc_dat line level to recover from errors, and for debugging.<br>This is especially useful in detecting the busy signal level from mmc_dat0 .<br>The value of these registers after reset depends on the mmc_dat lines level at that time. |
| 19    | WP       | R    | 0h    | Write Protect.<br>MMC/SD/SDIO1 only.<br>SDIO cards only.<br>This bit reflects the write protect input pin (SDWP) level.<br>The value of this register after reset depends one the protect input pin (SDWP) level at that time.<br>0x0 = If SD_CON[8] WPP is cleared to 0 (default), the card is write protected, otherwise the card is not write protected.<br>0x1 = If SD_CON[8] WPP is cleared to 0 (default), the card is not write protected, otherwise the card is write protected.    |

**Table 18-35. SD\_PSTATE Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 18    | CDPL     | R    | 0h    | <p>Card Detect Pin Level.<br/>MMC/SD/SDIO1 only.<br/>SDIO cards only.</p> <p>This bit reflects the inverse value of the card detect input pin (SDCD). Debouncing is not performed on this bit and is valid only when Card State is stable. (SD_PSTATE[17] is set to 1).</p> <p>This bit must be debounced by software. The value of this register after reset depends on the card detect input pin (SDCD) level at that time.</p> <p>0x0 = The value of the card detect input pin (SDCD) is 1.<br/>0x1 = The value of the card detect input pin (SDCD) is 0.</p>  |
| 17    | CSS      | R    | 0h    | <p>Card State Stable.</p> <p>This bit is used for testing. It is set to 1 only when Card Detect Pin Level is stable (SD_PSTATE[18] CPDL).</p> <p>Debouncing is performed on the card detect input pin (SDCD) to detect card stability.</p> <p>This bit is not affected by software reset.</p> <p>0x0 = Reset or Debouncing.<br/>0x1 = Reset or Debouncing.</p>  |
| 16    | CINS     | R    | 0h    | <p>Card inserted.</p> <p>This bit is the debounced value of the card detect input pin (SDCD). An inactive to active transition of the card detect input pin (SDCD) will generate a card insertion interrupt (SD_STAT[CINS]). A active to inactive transition of the card detect input pin (SDCD) will generate a card removal interrupt (SD_STAT[REM]).</p> <p>This bit is not affected by a software reset.</p> <p>0x0 = If SD_CON[CDP] is cleared to 0 (default), no card is detected. The card may have been removed from the card slot. If SD_CON[CDP] is set to 1, the card has been inserted.</p> <p>0x1 = If SD_CON[CDP] is cleared to 0 (default), the card has been inserted from the card slot. If SD_CON[CDP] is set to 1, no card is detected. The card may have been removed from the card slot.</p> |
| 15-12 | Reserved | R    | 0h    |   |
| 11    | BRE      | R    | 0h    | <p>Buffer read enable.</p> <p>This bit is used for non-DMA read transfers. It indicates that a complete block specified by SD_BLK [10:0] BLEN bits has been written in the buffer and is ready to be read.</p> <p>It is cleared to 0 when the entire block is read from the buffer. It is set to 1 when a block data is ready in the buffer and generates the Buffer read ready status of interrupt (SD_STAT[5] BRR bit).</p> <p>0x0 = Read BLEN bytes disable<br/>0x1 = Read BLEN bytes enable. Readable data exists in the buffer.</p>  |
| 10    | BWE      | R    | 0h    | <p>Buffer Write enable.</p> <p>This status is used for non-DMA write transfers. It indicates if space is available for write data.</p> <p>0x0 = There is no room left in the buffer to write BLEN bytes of data.<br/>0x1 = There is enough space in the buffer to write BLEN bytes of data.</p>   |
| 9     | RTA      | R    | 0h    | <p>Read transfer active.</p> <p>This status is used for detecting completion of a read transfer. It is set to 1 after the end bit of read command or by activating a continue request (SD_HCTL[17] CR bit) following a stop at block gap request.</p> <p>This bit is cleared to 0 when all data have been read by the local host after last block or after a stop at block gap request.</p> <p>0x0 = No valid data on the mmc_dat lines.<br/>0x1 = Read data transfer on going.</p>   |

**Table 18-35. SD\_PSTATE Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 8   | WTA      | R    | 0h    | <p>Write transfer active.<br/>         This status indicates a write transfer active.<br/>         It is set to 1 after the end bit of write command or by activating a continue request (SD_HCTL[17] CR bit) following a stop at block gap request.<br/>         This bit is cleared to 0 when CRC status has been received after last block or after a stop at block gap request.<br/>         0x0 = No valid data on the mmc_dat lines.<br/>         0x1 = Write data transfer on going.</p>  |
| 7-3 | Reserved | R    | 0h    |  |
| 2   | DLA      | R    | 0h    | <p>mmc_dat line active.<br/>         This status bit indicates whether one of the mmc_dat lines is in use.<br/>         In the case of read transactions (card to host)This bit is set to 1 after the end bit of read command or by activating continue request SD_HCTL[17] CR bit.<br/>         This bit is cleared to 0 when the host controller received the end bit of the last data block or at the beginning of the read wait mode.<br/>         In the case of write transactions (host to card)This bit is set to 1 after the end bit of write command or by activating continue request SD_HCTL[17] CR bit.<br/>         This bit is cleared to 0 on the end of busy event for the last block.<br/>         The host controller must wait 8 clock cycles with line not busy to really consider not "busy state" or after the busy block as a result of a stop at gap request.<br/>         0x0 = mmc_dat line inactive<br/>         0x1 = mmc_dat line active</p> |
| 1   | DATI     | R    | 0h    | <p>Command inhibit (mmc_dat).<br/>         This status bit is generated if either mmc_dat line is active (SD_PSTATE[2] DLA bit) or Read transfer is active (SD_PSTATE[9] RTA bit) or when a command with busy is issued.<br/>         This bit prevents the local host to issue a command.<br/>         A change of this bit from 1 to 0 generates a transfer complete interrupt (SD_STAT[1] TC bit).<br/>         0x0 = Issuing of command using the mmc_dat lines is allowed<br/>         0x1 = Issuing of command using mmc_dat lines is not allowed</p>  |
| 0   | CMDI     | R    | 0h    | <p>Command inhibit(mmc_cmd).<br/>         This status bit indicates that the mmc_cmd line is in use.<br/>         This bit is cleared to 0 when the most significant byte is written into the command register.<br/>         This bit is not set when Auto CMD12 is transmitted.<br/>         This bit is cleared to 0 in either the following cases: After the end bit of the command response, excepted if there is a command conflict error (SD_STAT[17] CCRC bit or SD_STAT[18] CEB bit set to 1) or a Auto CMD12 is not executed (SD_AC12[0] ACNE bit).<br/>         After the end bit of the command without response (SD_CMD [17:16] RSP_TYPE bits set to "00").<br/>         In case of a command data error is detected (SD_STAT[19] CTO bit set to 10, this register is not automatically cleared.<br/>         0x0 = Issuing of command using mmc_cmd line is allowed<br/>         0x1 = Issuing of command using mmc_cmd line is not allowed</p>               |

### 18.5.1.17 SD\_HCTL Register (offset = 228h) [reset = 0h]

SD\_HCTL is shown in [Figure 18-53](#) and described in [Table 18-36](#).

This register defines the host controls to set power, wake-up and transfer parameters. SD\_HCTL[31:24] = Wake-up control. SD\_HCTL[23:16] = Block gap control. SD\_HCTL[15:8] = Power control. SD\_HCTL[7:0] = Host control. If your device does not support MMC cards, then those bits in this register which are meant for MMC card use should be assumed to be reserved.

**Figure 18-53. SD\_HCTL Register**

| 31     | 30     | 29       | 28 | 27     | 26     | 25     | 24       |
|--------|--------|----------|----|--------|--------|--------|----------|
|        |        | Reserved |    | OBWE   | REM    | INS    | IWE      |
|        |        | R-0h     |    | R/W-0h | R/W-0h | R/W-0h | R/W-0h   |
| 23     | 22     | 21       | 20 | 19     | 18     | 17     | 16       |
|        |        | Reserved |    | IBG    | RWC    | CR     | SBGR     |
|        |        | R-0h     |    | R/W-0h | R/W-0h | R/W-0h | R/W-0h   |
| 15     | 14     | 13       | 12 | 11     | 10     | 9      | 8        |
|        |        | Reserved |    |        | SDVS   |        | SDBP     |
|        |        | R-0h     |    |        | R/W-0h |        | R/W-0h   |
| 7      | 6      | 5        | 4  | 3      | 2      | 1      | 0        |
| CDSS   | CDTL   | Reserved |    | DMAS   | HSPE   | DTW    | Reserved |
| R/W-0h | R/W-0h | R-0h     |    | R/W-0h | R/W-0h | R/W-0h | R-0h     |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-36. SD\_HCTL Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-28 | Reserved | R    | 0h    |   |
| 27    | OBWE     | R/W  | 0h    | Wake-up event enable for 'out-of-band' Interrupt.<br>This bit enables wake-up events for 'out-of-band' assertion.<br>Wake-up is generated if the wake-up feature is enabled (SD_SYSConfig[2] ENAWAKEUP bit).<br>The write to this register is ignored when SD_CON[14] OBIE bit is not set.<br>0x0 = Disable wake-up on 'out-of-band' Interrupt<br>0x1 = Enable wake-up on 'out-of-band' Interrupt |
| 26    | REM      | R/W  | 0h    | Wake-up event enable on SD card removal.<br>This bit enables wake-up events for card removal assertion.<br>Wake-up is generated if the wake-up feature is enabled (SD_SYSConfig[2] ENAWAKEUP bit).<br>0x0 = Disable wake-up on card removal<br>0x1 = Enable wake-up on card removal   |
| 25    | INS      | R/W  | 0h    | Wake-up event enable on SD card insertion This bit enables wake-up events for card insertion assertion.<br>Wake-up is generated if the wake-up feature is enabled (SD_SYSConfig[2] ENAWAKEUP bit).<br>0x0 = Disable wake-up on card insertion<br>0x1 = Enable wake-up on card insertion   |
| 24    | IWE      | R/W  | 0h    | Wake-up event enable on SD card interrupt.<br>This bit enables wake-up events for card interrupt assertion.<br>Wake-up is generated if the wake-up feature is enabled (SD_SYSConfig[2] ENAWAKEUP bit) and enable status bit is set (SD_IE[8] CIRQ_ENABLE bit).<br>0x0 = Disable wake-up on card interrupt<br>0x1 = Enable wake-up on card interrupt   |
| 23-20 | Reserved | R    | 0h    |   |

**Table 18-36. SD\_HCTL Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 19    | IBG      | R/W  | 0h    | Interrupt block at gap.<br>This bit is valid only in 4-bit mode of SDIO card to enable interrupt detection in the interrupt cycle at block gap for a multiple block transfer.<br>For MMC cards and for SD card this bit should be cleared to 0.<br>0x0 = Disable interrupt detection at the block gap in 4-bit mode<br>0x1 = Enable interrupt detection at the block gap in 4-bit mode  |
| 18    | RWC      | R/W  | 0h    | Read wait control.<br>The read wait function is optional only for SDIO cards.<br>If the card supports read wait, this bit must be enabled, then requesting a stop at block gap (SD_HCTL[16] SBGR bit) generates a read wait period after the current end of block.<br>Be careful, if read wait is not supported it may cause a conflict on mmc_dat line.<br>0x0 = Disable read wait control. Suspend/resume cannot be supported.<br>0x1 = Enable read wait control  |
| 17    | CR       | R/W  | 0h    | Continue request.<br>This bit is used to restart a transaction that was stopped by requesting a stop at block gap (SD_HCTL[16] SBGR bit).<br>Set this bit to 1 restarts the transfer.<br>The bit is automatically cleared to 0 by the host controller when transfer has restarted, that is, mmc_dat line is active (SD_PSTATE[2] DLA bit) or transferring data (SD_PSTATE[8] WTA bit).<br>The Stop at block gap request must be disabled (SD_HCTL[16] SBGR bit =0) before setting this bit.<br>0x0 = No affect<br>0x1 = Transfer restart  |
| 16    | SBGR     | R/W  | 0h    | Stop at block gap request.<br>This bit is used to stop executing a transaction at the next block gap.<br>The transfer can restart with a continue request (SD_HCTL[17] CR bit) or during a suspend/resume sequence.<br>In case of read transfer, the card must support read wait control.<br>In case of write transfer, the host driver shall set this bit after all block data written.<br>Until the transfer completion (SD_STAT[1] TC bit set to 1), the host driver shall leave this bit set to 1. If this bit is set, the local host shall not write to the data register (SD_DATA).<br>0x0 = Transfer mode<br>0x1 = Stop at block gap |
| 15-12 | Reserved | R    | 0h    |   |
| 11-9  | SDVS     | R/W  | 0h    | SD bus voltage select (All cards).<br>The host driver should set these bits to select the voltage level for the card according to the voltage supported by the system (SD_CAPA[26] VS18 bit, SD_CAPA[25] VS30 bit, SD_CAPA[24] VS33 bit) before starting a transfer.<br>If MMCHS<br>2: This field must be set to 5h.<br>If MMCHS<br>3: This field must be set to 5h.<br>0x5 = 1.8 V (Typical)<br>0x6 = 3.0 V (Typical)<br>0x7 = 3.3 V (Typical)   |

**Table 18-36. SD\_HCTL Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 8   | SDBP     | R/W  | 0h    | <p>SD bus power.</p> <p>Before setting this bit, the host driver shall select the SD bus voltage (SD_HCTL[11:9] SDVS bits).</p> <p>If the host controller detects the No card state, this bit is automatically cleared to 0.</p> <p>If the module is power off, a write in the command register (SD_CMD) will not start the transfer.</p> <p>A write to this bit has no effect if the selected SD bus voltage is not supported according to capability register (SD_CAPA[VS*]).</p> <p>0x0 = Power off<br/>0x1 = Power on</p>  |
| 7   | CDSS     | R/W  | 0h    | <p>Card Detect Signal Selection.</p> <p>This bit selects source for the card detection.</p> <p>When the source for the card detection is switched, the interrupt should be disabled during the switching period by clearing the Interrupt Status/Signal Enable register in order to mask unexpected interrupt being caused by the glitch.</p> <p>The Interrupt Status/Signal Enable should be disabled during over the period of debouncing.</p> <p>0x0 = SD_CD# is selected (for normal use).<br/>0x1 = The Card Detect Test Level is selected (for test purposes).</p> |
| 6   | CDTL     | R/W  | 0h    | <p>Card Detect Test Level.</p> <p>This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not.</p> <p>0 = No card<br/>1 = Card inserted.</p>  |
| 5   | Reserved | R    | 0h    |  |
| 4-3 | DMAS     | R/W  | 0h    | <p>DMA Select.</p> <p>One of the supported DMA modes can be selected.</p> <p>The host driver shall check support of DMA modes by referencing the Capabilities register.</p> <p>Use of selected DMA is determined by DMA Enable of the Transfer Mode register.</p> <p>This register is only meaningful when MADMA_EN is set to 1. When MADMA_EN is cleared to 0 the bit field is read only and returned value is 0.</p> <p>0x0 = Reserved<br/>0x1 = Reserved<br/>0x2 = 32-bit Address ADMA2 is selected.<br/>0x3 = Reserved</p>   |
| 2   | HSPE     | R/W  | 0h    | <p>High Speed Enable.</p> <p>Before setting this bit, the Host Driver shall check the High Speed Support in the Capabilities register.</p> <p>If this bit is cleared to 0 (default), the Host Controller outputs CMD line and DAT lines at the falling edge of the SD Clock.</p> <p>If this bit is set to 1, the Host Controller outputs CMD line and DAT lines at the rising edge of the SD Clock.</p> <p>This bit shall not be set when dual data rate mode is activated in SD_CON[DDR].</p> <p>0x0 = Normal speed mode<br/>0x1 = High speed mode</p>                  |
| 1   | DTW      | R/W  | 0h    | <p>Data transfer width.</p> <p>This bit must be set following a valid SET_BUS_WIDTH command (ACMD6) with the value written in bit 1 of the argument.</p> <p>Prior to this command, the SD card configuration register (SCR) must be verified for the supported bus width by the SD card.</p> <p>0x0 = 1-bit Data width (mmc_dat0 used)<br/>0x1 = 4-bit Data width (mmc_dat[3:0] used)</p>  |
| 0   | Reserved | R    | 0h    |  |

### 18.5.1.18 SD\_SYSCTL Register (offset = 22Ch) [reset = 0h]

SD\_SYSCTL is shown in [Figure 18-54](#) and described in [Table 18-37](#).

This register defines the system controls to set software resets, clock frequency management and data timeout. SD\_SYSCTL[31:24] = Software resets. SD\_SYSCTL[23:16] = Timeout control. SD\_SYSCTL[15:0] = Clock control.

**Figure 18-54. SD\_SYSCTL Register**

| 31 | 30       | 29       | 28       | 27     | 26     | 25     | 24     |
|----|----------|----------|----------|--------|--------|--------|--------|
|    |          | Reserved |          |        | SRD    | SRC    | SRA    |
|    |          | R-0h     |          |        | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22       | 21       | 20       | 19     | 18     | 17     | 16     |
|    | Reserved |          |          |        | DTO    |        |        |
|    |          | R-0h     |          |        | R/W-0h |        |        |
| 15 | 14       | 13       | 12       | 11     | 10     | 9      | 8      |
|    |          |          | CLKD     |        |        |        |        |
|    |          |          |          | R/W-0h |        |        |        |
| 7  | 6        | 5        | 4        | 3      | 2      | 1      | 0      |
|    | CLKD     |          | Reserved |        | CEN    | ICS    | ICE    |
|    |          |          | R-0h     |        | R/W-0h | R-0h   | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-37. SD\_SYSCTL Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-27 | Reserved | R    | 0h    |  |
| 26    | SRD      | R/W  | 0h    | <p>Software reset for mmc_dat line.<br/>           This bit is set to 1 for reset and released to 0 when completed.<br/>           Due to additional implementation logic, the reset does not immediately start when asserted.<br/>           The proper procedure is: (a) Set to 1 to start reset, (b) Poll for 1 to identify start of reset, and (c) Poll for 0 to identify reset is complete.<br/>           mmc_dat finite state machine in both clock domain are also reset.<br/>           These registers are cleared by the SD_SYSCTL[26] SRD bit:<br/>           SD_DATA.<br/>           SD_PSTATEBRE, BWE, RTA, WTA, DLA and DATI.<br/>           SD_HCTLSBGR and CR.<br/>           SD_STATBRR, BWR, BGE and TC Interconnect and MMC buffer data management is reinitialized.<br/>           Note: If a soft reset is issued when an interrupt is asserted, data may be lost.<br/>           0x0 = Reset completed<br/>           0x1 = Software reset for mmc_dat line</p> |
| 25    | SRC      | R/W  | 0h    | <p>Software reset for mmc_cmd line.<br/>           This bit is set to 1 for reset and released to 0 when completed.<br/>           Due to additional implementation logic, the reset does not immediately start when asserted.<br/>           The proper procedure is: (a) Set to 1 to start reset, (b) Poll for 1 to identify start of reset, and (c) Poll for 0 to identify reset is complete.<br/>           mmc_cmd finite state machine in both clock domain are also reset.<br/>           These registers are cleared by the SD_SYSCTL[25] SRC bit:<br/>           SD_PSTATECMDI.<br/>           SD_STATCC Interconnect and MMC command status management is reinitialized.<br/>           Note: If a soft reset is issued when an interrupt is asserted, data may be lost.<br/>           0x0 = Reset completed<br/>           0x1 = Software reset for mmc_cmd line</p>   |

**Table 18-37. SD\_SYSCTL Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 24    | SRA      | R/W  | 0h    | Software reset for all.<br>This bit is set to 1 for reset , and released to 0 when completed.<br>This reset affects the entire host controller except for the card detection circuit and capabilities registers.<br>0x0 = Reset completed<br>0x1 = Software reset for all the design   |
| 23-20 | Reserved | R    | 0h    |  |
| 19-16 | DTO      | R/W  | 0h    | Data timeout counter value and busy timeout.<br>This value determines the interval by which mmc_dat lines timeouts are detected.<br>The host driver needs to set this bit field based on: The maximum read access time (NAC) (Refer to the SD Specification Part1 Physical Layer).<br>The data read access time values (TAAC and NSAC) in the card specific data register (CSD) of the card.<br>The timeout clock base frequency (SD_CAPA [5:0] TCF bits).<br>If the card does not respond within the specified number of cycles, a data timeout error occurs (SD_STAT[20] DTO bit).<br>The SD_SYSCTL[19,16] DTO bit field is also used to check busy duration, to generate busy timeout for commands with busy response or for busy programming during a write command.<br>Timeout on CRC status is generated if no CRC token is present after a block write.<br>0x0 = TCF x 2^13<br>0x1 = TCF x 2^14<br>0xe = TCF x 2^27<br>0xf = Reserved |
| 15-6  | CLKD     | R/W  | 0h    | Clock frequency select.<br>These bits define the ratio between a reference clock frequency (system dependant) and the output clock frequency on the mmc_clk pin of either the memory card (MMC, SD, or SDIO).<br>0x0 = Clock Ref bypass<br>0x1 = Clock Ref bypass<br>0x2 = Clock Ref / 2<br>0x3 = Clock Ref / 3<br>0x3ff = Clock Ref / 1023  |
| 5-3   | Reserved | R    | 0h    |  |
| 2     | CEN      | R/W  | 0h    | Clock enable.<br>This bit controls if the clock is provided to the card or not.<br>0x0 = The clock is not provided to the card . Clock frequency can be changed .<br>0x1 = The clock is provided to the card and can be automatically gated when SD_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value). The host driver shall wait to set this bit to 1 until the Internal clock is stable (SD_SYSCTL[1] ICS bit).  |
| 1     | ICS      | R    | 0h    | Internal clock stable (status)<br>This bit indicates either the internal clock is stable or not.<br>0x0 = The internal clock is not stable.<br>0x1 = The internal clock is stable after enabling the clock (SD_SYSCTL[0] ICE bit) or after changing the clock ratio (SD_SYSCTL[15:6] CLKD bits).   |
| 0     | ICE      | R/W  | 0h    | Internal clock enable.<br>This register controls the internal clock activity.<br>In very low power state, the internal clock is stopped.<br>NoteThe activity of the debounce clock (used for wake-up events) and the interface clock (used for reads and writes to the module register map) are not affected by this register.<br>0x0 = The internal clock is stopped (very low power state).<br>0x1 = The internal clock oscillates and can be automatically gated when SD_SYSCONFIG[0] AUTOIDLE bit is set to 1 (default value).   |

### 18.5.1.19 SD\_STAT Register (offset = 230h) [reset = 0h]

SD\_STAT is shown in [Figure 18-55](#) and described in [Table 18-38](#).

The interrupt status regroups all the status of the module internal events that can generate an interrupt. SD\_STAT[31:16] = Error Interrupt Status. SD\_STAT[15:0] = Normal Interrupt Status. The error bits are located in the upper 16 bits of the SD\_STAT register. All bits are cleared by writing a 1 to them.

Additionally, bits 15 and 8 serve as special error bits. These cannot be cleared by writing a 1 to them. Bit 15 (ERRI) is automatically cleared when the error causing to ERRI to be set is handled. (that is, when bits 31:16 are cleared, bit 15 will be automatically cleared). Bit 8 (CIRQ) is cleared by writing a 0 to SD\_IE[8] (masking the interrupt) and servicing the interrupt.

**Figure 18-55. SD\_STAT Register**

| 31       | 30     | 29       | 28     | 27       | 26     | 25     | 24     |
|----------|--------|----------|--------|----------|--------|--------|--------|
| Reserved | BADA   | CERR     |        | Reserved | ADMAE  | ACE    |        |
| R-0h     | R/W-0h | R/W-0h   |        | R-0h     | R/W-0h | R/W-0h |        |
| 23       | 22     | 21       | 20     | 19       | 18     | 17     | 16     |
| Reserved | DEB    | DCRC     | DTO    | CIE      | CEB    | CCRC   | CTO    |
| R-0h     | R/W-0h | R/W-0h   | R/W-0h | R/W-0h   | R/W-0h | R/W-0h | R/W-0h |
| 15       | 14     | 13       | 12     | 11       | 10     | 9      | 8      |
| ERRI     |        | Reserved |        | BSR      | OBI    | CIRQ   |        |
| R-0h     |        | R-0h     |        | R/W-0h   | R-0h   | R-0h   |        |
| 7        | 6      | 5        | 4      | 3        | 2      | 1      | 0      |
| CREM     | CINS   | BRR      | BWR    | DMA      | BGE    | TC     | CC     |
| R/W-0h   | R/W-0h | R/W-0h   | R/W-0h | R/W-0h   | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-38. SD\_STAT Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-30 | Reserved | R    | 0h    |  |
| 29    | BADA     | R/W  | 0h    | Bad access to data space.<br>This bit is set automatically to indicate a bad access to buffer when not allowed: During a read access to the data register (SD_DATA) while buffer reads are not allowed (SD_PSTATE[11] BRE bit=0). During a write access to the data register (SD_DATA) while buffer writes are not allowed (SD_PSTATE[10] BWE bit=0).<br>0x0(W) = Status bit unchanged<br>0x0(R) = No interrupt<br>0x1(W) = Status is cleared.<br>0x1(R) = Bad access  |
| 28    | CERR     | R/W  | 0h    | Card error.<br>This bit is set automatically when there is at least one error in a response of type R1, R1b, R6, R5 or R5b.<br>Only bits referenced as type E (error) in status field in the response can set a card status error.<br>An error bit in the response is flagged only if corresponding bit in card status response error SD_CSRE is set.<br>There is no card error detection for autoCMD12 command.<br>The host driver shall read SD_RSP76 register to detect error bits in the command response.<br>0x0(W) = Status bit unchanged<br>0x0(R) = No error<br>0x1(W) = Status is cleared.<br>0x1(R) = Card error |
| 27-26 | Reserved | R    | 0h    |  |

**Table 18-38. SD\_STAT Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 25  | ADMAE    | R/W  | 0h    | <p>ADMA Error.<br/>           This bit is set when the Host Controller detects errors during ADMA based data transfer.<br/>           The state of the ADMA at an error occurrence is saved in the ADMA Error Status Register.<br/>           In addition, the Host Controller generates this interrupt when it detects invalid descriptor data (Valid=0) at the ST_FDS state.<br/>           ADMA Error State in the ADMA Error Status indicates that an error occurs in ST_FDS state.<br/>           The Host Driver may find that Valid bit is not set at the error descriptor.<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No interrupt<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = ADMA error</p> |
| 24  | ACE      | R/W  | 0h    | <p>Auto CMD12 error.<br/>           This bit is set automatically when one of the bits in Auto CMD12 Error status register has changed from 0 to 1.<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = AutoCMD12 error</p>   |
| 23  | Reserved | R    | 0h    |  |
| 22  | DEB      | R/W  | 0h    | <p>Data End Bit error.<br/>           This bit is set automatically when detecting a 0 at the end bit position of read data on mmc_dat line or at the end position of the CRC status in write mode.<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Data end bit error</p>  |
| 21  | DCRC     | R/W  | 0h    | <p>Data CRC Error.<br/>           This bit is set automatically when there is a CRC16 error in the data phase response following a block read command or if there is a 3-bit CRC status different of a position "010" token during a block write command.<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Data CRC error</p>  |
| 20  | DTO      | R/W  | 0h    | <p>Data timeout error.<br/>           This bit is set automatically according to the following conditions:<br/>           Busy timeout for R1b, R5b response type.<br/>           Busy timeout after write CRC status.<br/>           Write CRC status timeout.<br/>           Read data timeout.<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Time out</p>  |
| 19  | CIE      | R/W  | 0h    | <p>Command index error.<br/>           This bit is set automatically when response index differs from corresponding command index previously emitted.<br/>           It depends on the enable bit (SD_CMD[20] CICE).<br/>           0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Command index error</p>  |

**Table 18-38. SD\_STAT Register Field Descriptions (continued)**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 18    | CEB      | R/W  | 0h    | <p>Command end bit error.<br/>           This bit is set automatically when detecting a 0 at the end bit position of a command response.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Command end bit error</p>   |
| 17    | CCRC     | R/W  | 0h    | <p>Command CRC error.<br/>           This bit is set automatically when there is a CRC7 error in the command response depending on the enable bit (SD_CMD[19] CCCE).</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Command CRC error</p>   |
| 16    | CTO      | R/W  | 0h    | <p>Command timeout error.<br/>           This bit is set automatically when no response is received within 64 clock cycles from the end bit of the command.<br/>           For commands that reply within 5 clock cycles - the timeout is still detected at 64 clock cycles.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No error<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Time Out</p>  |
| 15    | ERRI     | R    | 0h    | <p>Error interrupt.<br/>           If any of the bits in the Error Interrupt Status register (SD_STAT [31:16]) are set, then this bit is set to 1.<br/>           Therefore the host driver can efficiently test for an error by checking this bit first.<br/>           Writes to this bit are ignored.</p> <p>0x0 = No interrupt<br/>           0x1 = Error interrupt event(s) occurred</p>   |
| 14-11 | Reserved | R    | 0h    |   |
| 10    | BSR      | R/W  | 0h    | <p>Boot Status Received Interrupt.<br/>           This bit is set automatically when SD_CON[BOOT] is set 1 or 2 and a boot status is received on DAT[0] line.<br/>           This interrupt is only useful for MMC card.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No interrupt<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Boot Status Received Interrupt occurred.</p>  |
| 9     | OBI      | R    | 0h    | <p>Out-of-band interrupt (This interrupt is only useful for MMC card).<br/>           This bit is set automatically when SD_CON[14] OBIE bit is set and an out-of-band interrupt occurs on OBI pin.<br/>           The interrupt detection depends on polarity controlled by SD_CON[13] OBIP bit.<br/>           The out-of-band interrupt signal is a system specific feature for future use, this signal is not required for existing specification implementation.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0 = No out-of-band interrupt<br/>           0x1(W) = Status is cleared.<br/>           0x1 = Interrupt out-of-band occurs</p> |

**Table 18-38. SD\_STAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 8   | CIRQ  | R    | 0h    | <p>Card interrupt.<br/> This bit is only used for SD and SDIO cards.<br/> In<br/> 1-bit mode, interrupt source is asynchronous (can be a source of asynchronous wake-up).<br/> In<br/> 4-bit mode, interrupt source is sampled during the interrupt cycle.<br/> In CE-ATA mode, interrupt source is detected when the card drives mmc_cmd line to zero during one cycle after data transmission end.<br/> All modes above are fully exclusive.<br/> The controller interrupt must be clear by setting SD_IE[8] CIRQ_ENABLE to 0, then the host driver must start the interrupt service with card (clearing card interrupt status) to remove card interrupt source.<br/> Otherwise the Controller interrupt will be reasserted as soon as SD_IE[8] CIRQ_ENABLE is set to 1.<br/> Writes to this bit are ignored.<br/> 0x0 = No card interrupt<br/> 0x1 = Generate card interrupt</p> |
| 7   | CREM  | R/W  | 0h    | <p>Card Removal.<br/> This bit is set automatically when SD_PSTATE[CINS] changes from 1 to 0.<br/> A clear of this bit doesn't affect Card inserted present state (SD_PSTATE[CINS]).<br/> 0x0(W) = Status bit unchanged<br/> 0x0(R) = Card State stable or debouncing<br/> 0x1(W) = Status is cleared<br/> 0x1(R) = Card Removed</p>  |
| 6   | CINS  | R/W  | 0h    | <p>Card Insertion.<br/> This bit is set automatically when SD_PSTATE[CINS] changes from 0 to 1.<br/> A clear of this bit doesn't affect Card inserted present state (SD_PSTATE[CINS]).<br/> 0x0(W) = Status bit unchanged<br/> 0x0(R) = Card State stable or debouncing<br/> 0x1(W) = Status is cleared.<br/> 0x1(R) = Card inserted</p>  |
| 5   | BRR   | R/W  | 0h    | <p>Buffer read ready.<br/> This bit is set automatically during a read operation to the card (see class 2 - block oriented read commands) when one block specified by the SD_BLK [10:0] BLEN bit field is completely written in the buffer.<br/> It indicates that the memory card has filled out the buffer and that the local host needs to empty the buffer by reading it.<br/> Note: If the DMA receive-mode is enabled, this bit is never set instead a DMA receive request to the main DMA controller of the system is generated.<br/> 0x0(W) = Status bit unchanged<br/> 0x0(R) = Not ready to read buffer<br/> 0x1(W) = Status is cleared.<br/> 0x1(R) = Ready to read buffer</p>   |

**Table 18-38. SD\_STAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 4   | BWR   | R/W  | 0h    | <p>Buffer write ready.<br/>           This bit is set automatically during a write operation to the card (see class 4 - block oriented write command) when the host can write a complete block as specified by SD_BLK [10:0] BLEN.</p> <p>It indicates that the memory card has emptied one block from the buffer and that the local host is able to write one block of data into the buffer.</p> <p>Note: If the DMA transmit mode is enabled, this bit is never set instead, a DMA transmit request to the main DMA controller of the system is generated.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = Not ready to write buffer<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = Ready to write buffer</p> |
| 3   | DMA   | R/W  | 0h    | <p>DMA Interrupt.<br/>           This status is set when an interrupt is required in the ADMA instruction and after the data transfer completion.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = DMA Interrupt detected<br/>           0x1(W) = Status is cleared.<br/>           0x1(R) = No DMA Interrupt</p>  |
| 2   | BGE   | R/W  | 0h    | <p>Block gap event.<br/>           When a stop at block gap is requested (SD_HCTL[16] SBGR bit), this bit is automatically set when transaction is stopped at the block gap during a read or write operation.</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No block gap event<br/>           0x1(W) = Status is cleared<br/>           0x1(R) = Transaction stopped at block gap</p>   |
| 1   | TC    | R/W  | 0h    | <p>Transfer completed.<br/>           This bit is always set when a read/write transfer is completed or between two blocks when the transfer is stopped due to a stop at block gap request (SD_HCTL[16] SBGR bit).</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No transfer complete<br/>           0x1(W) = Status is cleared<br/>           0x1(R) = Data transfer complete</p>  |
| 0   | CC    | R/W  | 0h    | <p>Command complete.<br/>           This bit is set when a 1-to-0 transition occurs in the register command inhibit (SD_PSTATE[0] CMDI bit)</p> <p>0x0(W) = Status bit unchanged<br/>           0x0(R) = No command complete<br/>           0x1(W) = Status is cleared<br/>           0x1(R) = Command complete</p>  |

### 18.5.1.20 SD\_IE Register (offset = 234h) [reset = 0h]

SD\_IE is shown in [Figure 18-56](#) and described in [Table 18-39](#).

This register allows to enable/disable the module to set status bits, on an event-by-event basis.  
 SD\_IE[31:16] = Error Interrupt Status Enable. SD\_IE[15:0] = Normal Interrupt Status Enable.

**Figure 18-56. SD\_IE Register**

| 31          | 30          | 29          | 28         | 27         | 26         | 25          | 24          |
|-------------|-------------|-------------|------------|------------|------------|-------------|-------------|
| Reserved    | BADA_ENABLE | CERR_ENABLE |            | Reserved   |            | ADMA_ENABLE | ACE_ENABLE  |
| R-0h        | R/W-0h      | R/W-0h      |            | R-0h       |            | R/W-0h      | R/W-0h      |
| 23          | 22          | 21          | 20         | 19         | 18         | 17          | 16          |
| Reserved    | DEB_ENABLE  | DCRC_ENABLE | DTO_ENABLE | CIE_ENABLE | CEB_ENABLE | CCRC_ENABLE | CTO_ENABLE  |
| R-0h        | R/W-0h      | R/W-0h      | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h      | R/W-0h      |
| 15          | 14          | 13          | 12         | 11         | 10         | 9           | 8           |
| NULL        |             | Reserved    |            |            | BSR_ENABLE | OBI_ENABLE  | CIRQ_ENABLE |
| R-0h        |             | R-0h        |            |            | R/W-0h     | R/W-0h      | R/W-0h      |
| 7           | 6           | 5           | 4          | 3          | 2          | 1           | 0           |
| CREM_ENABLE | CINS_ENABLE | BRR_ENABLE  | BWR_ENABLE | DMA_ENABLE | BGE_ENABLE | TC_ENABLE   | CC_ENABLE   |
| R/W-0h      | R/W-0h      | R/W-0h      | R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h      | R/W-0h      |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-39. SD\_IE Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-30 | Reserved    | R    | 0h    |  |
| 29    | BADA_ENABLE | R/W  | 0h    | Bad access to data space interrupt enable<br>0x0 = Masked<br>0x1 = Enabled |
| 28    | CERR_ENABLE | R/W  | 0h    | Card error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled               |
| 27-26 | Reserved    | R    | 0h    |  |
| 25    | ADMA_ENABLE | R/W  | 0h    | ADMA error Interrupt Enable<br>0x0 = Masked<br>0x1 = Enabled               |
| 24    | ACE_ENABLE  | R/W  | 0h    | Auto CMD12 error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled         |
| 23    | Reserved    | R    | 0h    |  |
| 22    | DEB_ENABLE  | R/W  | 0h    | Data end bit error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled       |
| 21    | DCRC_ENABLE | R/W  | 0h    | Data CRC error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled           |
| 20    | DTO_ENABLE  | R/W  | 0h    | Data timeout error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled       |
| 19    | CIE_ENABLE  | R/W  | 0h    | Command index error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled      |

**Table 18-39. SD\_IE Register Field Descriptions (continued)**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 18    | CEB_ENABLE  | R/W  | 0h    | Command end bit error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 17    | CCRC_ENABLE | R/W  | 0h    | Command CRC error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 16    | CTO_ENABLE  | R/W  | 0h    | Command timeout error interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 15    | NULL        | R    | 0h    | Fixed to 0.<br>The host driver shall control error interrupts using the Error Interrupt Signal Enable register.<br>Writes to this bit are ignored.  |
| 14-11 | Reserved    | R    | 0h    |   |
| 10    | BSR_ENABLE  | R/W  | 0h    | Boot Status Interrupt Enable A write to this register when SD_CON[BOOT] is cleared to 0 is ignored.<br>0x0 = Masked<br>0x1 = Enabled  |
| 9     | OBI_ENABLE  | R/W  | 0h    | Out-of-band interrupt enable A write to this register when SD_CON[14] OBIE is cleared to 0 is ignored.<br>0x0 = Masked<br>0x1 = Enabled   |
| 8     | CIRQ_ENABLE | R/W  | 0h    | Card interrupt enable.<br>A clear of this bit also clears the corresponding status bit.<br>During 1-bit mode, if the interrupt routine does not remove the source of a card interrupt in the SDIO card, the status bit is reasserted when this bit is set to 1.<br>This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled |
| 7     | CREM_ENABLE | R/W  | 0h    | Card Removal interrupt Enable This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled  |
| 6     | CINS_ENABLE | R/W  | 0h    | Card Insertion interrupt Enable This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled  |
| 5     | BRR_ENABLE  | R/W  | 0h    | Buffer read ready interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 4     | BWR_ENABLE  | R/W  | 0h    | Buffer write ready interrupt enable<br>0x0 = Masked<br>0x1 = Enabled  |
| 3     | DMA_ENABLE  | R/W  | 0h    | DMA interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |

**Table 18-39. SD\_IE Register Field Descriptions (continued)**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 2   | BGE_ENABLE | R/W  | 0h    | Block gap event interrupt enable<br>0x0 = Masked<br>0x1 = Enabled    |
| 1   | TC_ENABLE  | R/W  | 0h    | Transfer completed interrupt enable<br>0x0 = Masked<br>0x1 = Enabled |
| 0   | CC_ENABLE  | R/W  | 0h    | Command completed interrupt enable<br>0x0 = Masked<br>0x1 = Enabled  |

### 18.5.1.21 SD\_ISE Register (offset = 238h) [reset = 0h]

SD\_ISE is shown in [Figure 18-57](#) and described in [Table 18-40](#).

This register allows you to enable/disable the module to set status bits, on an event-by-event basis.  
 SD\_ISE[31:16] = Error Interrupt Signal Enable. SD\_ISE[15:0] = Normal Interrupt Signal Enable.

**Figure 18-57. SD\_ISE Register**

| 31         | 30         | 29         | 28        | 27        | 26        | 25         | 24        |
|------------|------------|------------|-----------|-----------|-----------|------------|-----------|
| Reserved   | BADA_SIGEN | CERR_SIGEN |           | Reserved  |           | ADMA_SIGEN | ACE_SIGEN |
| R-0h       | R/W-0h     | R/W-0h     |           | R-0h      |           | R/W-0h     | R/W-0h    |
| 23         | 22         | 21         | 20        | 19        | 18        | 17         | 16        |
| Reserved   | DEB_SIGEN  | DCRC_SIGEN | DTO_SIGEN | CIE_SIGEN | CEB_SIGEN | CCRC_SIGEN | CTO_SIGEN |
| R-0h       | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    | R/W-0h    | R/W-0h     | R/W-0h    |
| 15         | 14         | 13         | 12        | 11        | 10        | 9          | 8         |
| NULL       |            | Reserved   |           | BSR_SIGEN | OBI_SIGEN | CIRQ_SIGEN |           |
| R-0h       |            | R-0h       |           | R/W-0h    | R/W-0h    | R/W-0h     | R/W-0h    |
| 7          | 6          | 5          | 4         | 3         | 2         | 1          | 0         |
| CREM_SIGEN | CINS_SIGEN | BRR_SIGEN  | BWR_SIGEN | DMA_SIGEN | BGE_SIGEN | TC_SIGEN   | CC_SIGEN  |
| R/W-0h     | R/W-0h     | R/W-0h     | R/W-0h    | R/W-0h    | R/W-0h    | R/W-0h     | R/W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-40. SD\_ISE Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-30 | Reserved   | R    | 0h    |  |
| 29    | BADA_SIGEN | R/W  | 0h    | Bad access to data space interrupt enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 28    | CERR_SIGEN | R/W  | 0h    | Card error interrupt signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 27-26 | Reserved   | R    | 0h    |  |
| 25    | ADMA_SIGEN | R/W  | 0h    | ADMA error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 24    | ACE_SIGEN  | R/W  | 0h    | Auto CMD12 error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 23    | Reserved   | R    | 0h    |  |
| 22    | DEB_SIGEN  | R/W  | 0h    | Data end bit error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 21    | DCRC_SIGEN | R/W  | 0h    | Data CRC error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 20    | DTO_SIGEN  | R/W  | 0h    | Data timeout error signal status enable<br>0x0 = Masked. The host controller provides the clock to the card until the card sends the data or the transfer is aborted.<br>0x1 = Enabled |
| 19    | CIE_SIGEN  | R/W  | 0h    | Command index error signal status enable<br>0x0 = Masked<br>0x1 = Enabled  |

**Table 18-40. SD\_ISE Register Field Descriptions (continued)**

| Bit   | Field      | Type | Reset | Description   |
|-------|------------|------|-------|---|
| 18    | CEB_SIGEN  | R/W  | 0h    | Command end bit error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 17    | CCRC_SIGEN | R/W  | 0h    | Command CRC error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 16    | CTO_SIGEN  | R/W  | 0h    | Command timeout error signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 15    | NULL       | R    | 0h    | Fixed to 0.<br>The host driver shall control error interrupts using the error interrupt signal enable register.<br>Writes to this bit are ignored.  |
| 14-11 | Reserved   | R    | 0h    |   |
| 10    | BSR_SIGEN  | R/W  | 0h    | Boot Status signal status enable.<br>A write to this register when SD_CON[BOOT] is cleared to 0 is ignored<br>0x0 = Masked<br>0x1 = Enabled   |
| 9     | OBI_SIGEN  | R/W  | 0h    | Out-of-band interrupt signal status enable.<br>A write to this register when SD_CON[14] OBIE is cleared to 0 is ignored.<br>0x0 = Masked<br>0x1 = Enabled   |
| 8     | CIRQ_SIGEN | R/W  | 0h    | Card interrupt signal status enable.<br>A clear of this bit also clears the corresponding status bit.<br>During 1-bit mode, if the interrupt routine does not remove the source of a card interrupt in the SDIO card, the status bit is reasserted when this bit is set to 1.<br>This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled |
| 7     | CREM_SIGEN | R/W  | 0h    | Card Removal signal status enable This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled  |
| 6     | CINS_SIGEN | R/W  | 0h    | Card Insertion signal status enable.<br>This bit must be set to 1 when entering in smart idle mode to enable system to identify wake-up event and to allow controller to clear internal wake-up source.<br>0x0 = Masked<br>0x1 = Enabled  |
| 5     | BRR_SIGEN  | R/W  | 0h    | Buffer read ready signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |
| 4     | BWR_SIGEN  | R/W  | 0h    | Buffer write ready signal status enable<br>0x0 = Masked<br>0x1 = Enabled  |
| 3     | DMA_SIGEN  | R/W  | 0h    | DMA signal status enable<br>0x0 = Masked<br>0x1 = Enabled   |

**Table 18-40. SD\_ISE Register Field Descriptions (continued)**

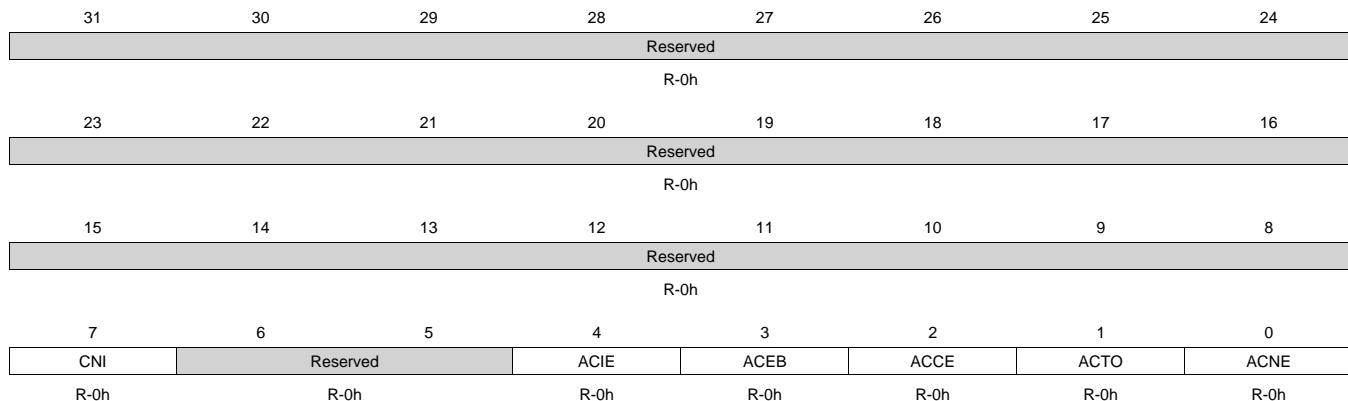
| Bit | Field     | Type | Reset | Description  |
|-----|-----------|------|-------|--|
| 2   | BGE_SIGEN | R/W  | 0h    | Block gap event signal status enable<br>0x0 = Masked<br>0x1 = Enabled    |
| 1   | TC_SIGEN  | R/W  | 0h    | Transfer completed signal status enable<br>0x0 = Masked<br>0x1 = Enabled |
| 0   | CC_SIGEN  | R/W  | 0h    | Command completed signal status enable<br>0x0 = Masked<br>0x1 = Enabled  |

### 18.5.1.22 SD\_AC12 Register (offset = 23Ch) [reset = 0h]

SD\_AC12 is shown in [Figure 18-58](#) and described in [Table 18-41](#).

The host driver may determine which of the errors cases related to Auto CMD12 has occurred by checking this SD\_AC12 register when an auto CMD12 error interrupt occurs. This register is valid only when auto CMD12 is enabled (SD\_CMD[2] ACEN bit) and auto CMD12Error (SD\_STAT[24] ACE bit) is set to 1. These bits are automatically reset when starting a new adtc command with data.

**Figure 18-58. SD\_AC12 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-41. SD\_AC12 Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7    | CNI      | R    | 0h    | Command not issue by auto CMD12 error.<br>If this bit is set to 1, it means that pending command is not executed due to auto CMD12 error ACEB, ACCE, ACTO, or ACNE.<br>0x0 = Not error<br>0x1 = Command not issued   |
| 6-5  | Reserved | R    | 0h    |  |
| 4    | ACIE     | R    | 0h    | Auto CMD12 index error.<br>This bit is a set to 1 when response index differs from corresponding command auto CMD12 index previously emitted.<br>This bit depends on the command index check enable (SD_CMD[20] CICE bit).<br>0x0 = No error<br>0x1 = Auto CMD12 index error |
| 3    | ACEB     | R    | 0h    | Auto CMD12 end bit error.<br>This bit is set to 1 when detecting a 0 at the end bit position of auto CMD12 command response.<br>0x0 = No error<br>0x1 = AutoCMD12 end bit error  |
| 2    | ACCE     | R    | 0h    | Auto CMD12 CRC error.<br>This bit is automatically set to 1 when a CRC7 error is detected in the auto CMD12 command response depending on the enable in the SD_CMD[19] CCCE bit.<br>0x0 = No error<br>0x1 = Auto CMD12 CRC error   |
| 1    | ACTO     | R    | 0h    | Auto CMD12 timeout error.<br>This bit is set to 1 if no response is received within 64 clock cycles from the end bit of the auto CMD12 command.<br>0x0 = No error<br>0x1 = Auto CMD12 time out   |

**Table 18-41. SD\_AC12 Register Field Descriptions (continued)**

| <b>Bit</b> | <b>Field</b> | <b>Type</b> | <b>Reset</b> | <b>Description</b>   |
|------------|--------------|-------------|--------------|--|
| 0          | ACNE         | R           | 0h           | Auto CMD12 not executed.<br>This bit is set to 1 if multiple block data transfer command has started and if an error occurs in command before auto CMD12 starts.<br>0x0 = Auto CMD12 executed<br>0x1 = Auto CMD12 not executed |

### 18.5.1.23 SD\_CAPA Register (offset = 240h) [reset = 0h]

SD\_CAPA is shown in [Figure 18-59](#) and described in [Table 18-42](#).

This register lists the capabilities of the MMC/SD/SDIO host controller.

**Figure 18-59. SD\_CAPA Register**

| 31       | 30       | 29   | 28       | 27       | 26       | 25     | 24     |
|----------|----------|------|----------|----------|----------|--------|--------|
|          | Reserved |      | 64BIT    | Reserved | VS18     | VS30   | VS33   |
|          | R-0h     |      | R/W-0h   | R-0h     | R/W-0h   | R/W-0h | R/W-0h |
| 23       | 22       | 21   | 20       | 19       | 18       | 17     | 16     |
| SRS      | DS       | HSS  | Reserved | AD2S     | Reserved |        | MBL    |
| R-0h     | R-0h     | R-0h | R-0h     | R-0h     | R-0h     |        | R-0h   |
| 15       | 14       | 13   | 12       | 11       | 10       | 9      | 8      |
| Reserved |          |      |          | BCF      |          |        |        |
|          | R-0h     |      |          | R-0h     |          |        |        |
| 7        | 6        | 5    | 4        | 3        | 2        | 1      | 0      |
| TCU      | Reserved |      |          | TCF      |          |        |        |
| R-0h     | R-0h     |      |          | R-0h     |          |        |        |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-42. SD\_CAPA Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-29 | Reserved | R    | 0h    |  |
| 28    | 64BIT    | R/W  | 0h    | 64 Bit System Bus Support.<br>Setting 1 to this bit indicates that the Host Controller supports 64-bit address descriptor mode and is connected to 64-bit address system bus.<br>0x0(R) = 32-bit System bus address<br>0x1(R) = 64-bit System bus address  |
| 27    | Reserved | R    | 0h    |  |
| 26    | VS18     | R/W  | 0h    | Voltage support 1.8 V.<br>Initialization of this register (via a write access to this register) depends on the system capabilities.<br>The host driver shall not modify this register after the initialization.<br>This register is only reinitialized by a hard reset (via mmc_RESET signal).<br>0x0(W) = 1.8 V not supported<br>0x0(R) = 1.8 V not supported<br>0x1(W) = 1.8 V supported<br>0x1(R) = 1.8 V supported |
| 25    | VS30     | R/W  | 0h    | Voltage support 3.0V.<br>Initialization of this register (via a write access to this register) depends on the system capabilities.<br>The host driver shall not modify this register after the initialization.<br>This register is only reinitialized by a hard reset (via mmc_RESET signal).<br>0x0(W) = 3.0 V not supported<br>0x0(R) = 3.0 V not supported<br>0x1(W) = 3.0 V supported<br>0x1(R) = 3.0 V supported  |

**Table 18-42. SD\_CAPA Register Field Descriptions (continued)**

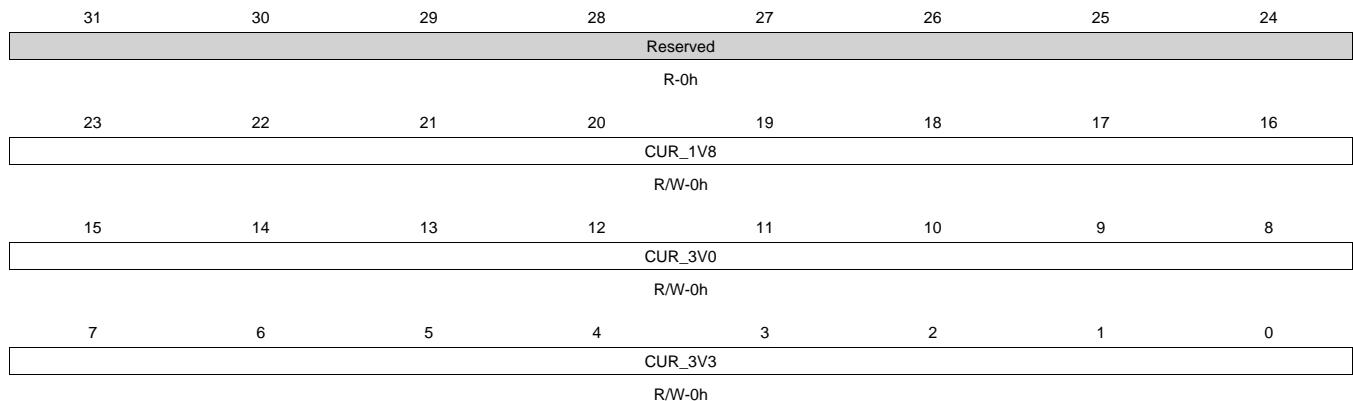
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 24    | VS33     | R/W  | 0h    | Voltage support 3.3V.<br>Initialization of this register (via a write access to this register) depends on the system capabilities.<br>The host driver shall not modify this register after the initialization.<br>This register is only reinitialized by a hard reset (via mmc_RESET signal).<br>0x0(W) = 3.3 V not supported<br>0x0(R) = 3.3 V not supported<br>0x1(W) = 3.3 V supported<br>0x1(R) = 3.3 V supported |
| 23    | SRS      | R    | 0h    | Suspend/resume support (SDIO cards only).<br>This bit indicates whether the host controller supports Suspend/Resume functionality.<br>0x0 = The Host controller does not suspend/resume functionality.<br>0x1 = The Host controller supports suspend/resume functionality.  |
| 22    | DS       | R    | 0h    | DMA support.<br>This bit indicates that the Host controller is able to use DMA to transfer data between system memory and the Host controller directly.<br>0x0 = DMA not supported<br>0x1 = DMA supported   |
| 21    | HSS      | R    | 0h    | High-speed support.<br>This bit indicates that the host controller supports high speed operations and can supply an up-to-52 MHz clock to the card.<br>0x0 = DMA not supported<br>0x1 = DMA supported   |
| 20    | Reserved | R    | 0h    |   |
| 19    | AD2S     | R    | 0h    | This bit indicates whether the Host Controller is capable of using ADMA2.<br>It depends on setting of generic parameter MADMA_EN.<br>0x0 = ADMA2 supported<br>0x1 = ADMA2 not supported   |
| 18    | Reserved | R    | 0h    |   |
| 17-16 | MBL      | R    | 0h    | Maximum block length.<br>This value indicates the maximum block size that the host driver can read and write to the buffer in the host controller.<br>The host controller supports 512 bytes and 1024 bytes block transfers.<br>0x0 = 512 bytes<br>0x1 = 1024 bytes<br>0x2 = 2048 bytes   |
| 15-14 | Reserved | R    | 0h    |   |
| 13-8  | BCF      | R    | 0h    | Base clock frequency for clock provided to the card.<br>ARRAY(0x1bfe1b0)  |
| 7     | TCU      | R    | 0h    | Timeout clock unit.<br>This bit shows the unit of base clock frequency used to detect Data Timeout Error (SD_STAT[20] DTO bit).<br>0x0 = kHz<br>0x1 = MHz   |
| 6     | Reserved | R    | 0h    |   |
| 5-0   | TCF      | R    | 0h    | Timeout clock frequency.<br>The timeout clock frequency is used to detect Data Timeout Error (SD_STAT[20] DTO bit).<br>0x0 = The timeout clock frequency depends on the frequency of the clock provided to the card. The value of the timeout clock frequency is not available in this register.  |

### 18.5.1.24 SD\_CUR\_CAPA Register (offset = 248h) [reset = 0h]

SD\_CUR\_CAPA is shown in [Figure 18-60](#) and described in [Table 18-43](#).

This register indicates the maximum current capability for each voltage. The value is meaningful if the voltage support is set in the capabilities register (SD\_CAPA). Initialization of this register (via a write access to this register) depends on the system capabilities. The host driver shall not modify this register after the initialization. This register is only reinitialized by a hard reset (via mmc\_RESET signal).

**Figure 18-60. SD\_CUR\_CAPA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-43. SD\_CUR\_CAPA Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | Reserved | R    | 0h    |  |
| 23-16 | CUR_1V8  | R/W  | 0h    | Maximum current for 1.8 V<br>0x0(R) = The maximum current capability for this voltage is not available. Feature not implemented. |
| 15-8  | CUR_3V0  | R/W  | 0h    | Maximum current for 3.0 V<br>0x0(R) = The maximum current capability for this voltage is not available. Feature not implemented. |
| 7-0   | CUR_3V3  | R/W  | 0h    | Maximum current for 3.3 V<br>0x0(R) = The maximum current capability for this voltage is not available. Feature not implemented. |

### 18.5.1.25 SD\_FE Register (offset = 250h) [reset = 0h]

SD\_FE is shown in [Figure 18-61](#) and described in [Table 18-44](#).

The Force Event register is not a physically implemented register. Rather, it is an address at which the Error Interrupt Status register can be written. The effect of a write to this address will be reflected in the Error Interrupt Status Register, if corresponding bit of the Error Interrupt Status Enable Register is set.

**Figure 18-61. SD\_FE Register**

| 31       | 30       | 29      | 28      | 27       | 26      | 25       | 24      |
|----------|----------|---------|---------|----------|---------|----------|---------|
| Reserved | FE_BADA  | FE_CERR |         | Reserved |         | FE_ADMAE | FE_ACE  |
| R-0h     | W-0h     | W-0h    |         | R-0h     |         | W-0h     | W-0h    |
| 23       | 22       | 21      | 20      | 19       | 18      | 17       | 16      |
| Reserved | FE_DEB   | FE_DCRC | FE.DTO  | FE_CIE   | FE_CEB  | FE_CCRC  | FE_CTO  |
| R-0h     | W-0h     | W-0h    | W-0h    | W-0h     | W-0h    | W-0h     | W-0h    |
| 15       | 14       | 13      | 12      | 11       | 10      | 9        | 8       |
| Reserved |          |         |         |          |         |          |         |
| R-0h     |          |         |         |          |         |          |         |
| 7        | 6        | 5       | 4       | 3        | 2       | 1        | 0       |
| FE_CNI   | Reserved |         | FE_ACIE | FE_ACEB  | FE_ACCE | FE_ACTO  | FE_ACNE |
| W-0h     | R-0h     |         | W-0h    | W-0h     | W-0h    | W-0h     | W-0h    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-44. SD\_FE Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-30 | Reserved | R    | 0h    |   |
| 29    | FE_BADA  | W    | 0h    | Force Event Bad access to data space.<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced. |
| 28    | FE_CERR  | W    | 0h    | Force Event Card error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                |
| 27-26 | Reserved | R    | 0h    |   |
| 25    | FE_ADMAE | W    | 0h    | Force Event ADMA error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                |
| 24    | FE_ACE   | W    | 0h    | Force Event Auto CMD12 error.<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.         |
| 23    | Reserved | R    | 0h    |   |
| 22    | FE_DEB   | W    | 0h    | Force Event Data End Bit error.<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.       |
| 21    | FE_DCRC  | W    | 0h    | Force Event Data CRC error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.            |
| 20    | FE.DTO   | W    | 0h    | Force Event Data timeout error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.        |
| 19    | FE_CIE   | W    | 0h    | Force Event Command index error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.       |

**Table 18-44. SD\_FE Register Field Descriptions (continued)**

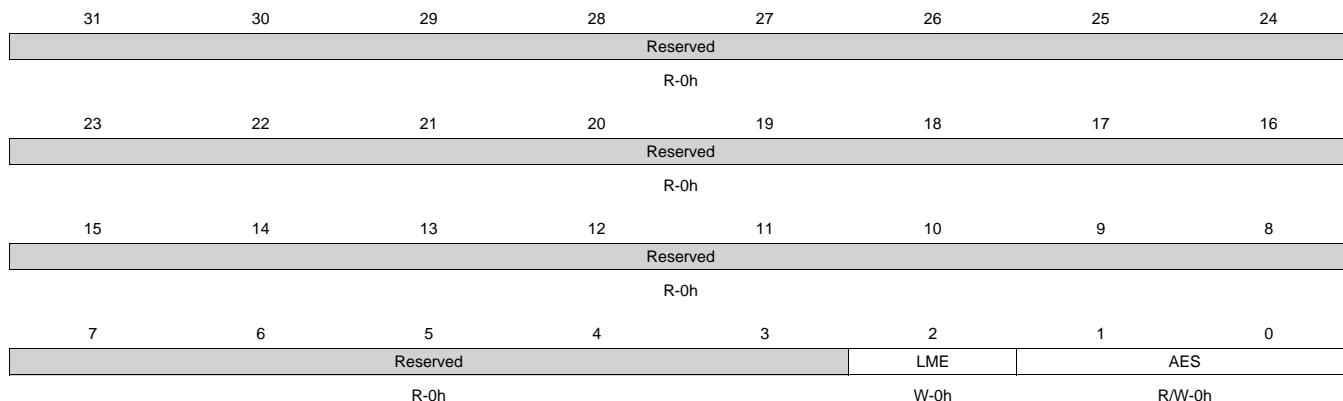
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 18   | FE_CEB   | W    | 0h    | Force Event Command end bit error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                 |
| 17   | FE_CCRC  | W    | 0h    | Force Event Command CRC error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                     |
| 16   | FE_CTO   | W    | 0h    | Force Event Command Timeout error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                 |
| 15-8 | Reserved | R    | 0h    |   |
| 7    | FE_CNI   | W    | 0h    | Force Event Command not issue by Auto CMD12 error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced. |
| 6-5  | Reserved | R    | 0h    |   |
| 4    | FE_ACIE  | W    | 0h    | Force Event Auto CMD12 index error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                |
| 3    | FE_ACEB  | W    | 0h    | Force Event Auto CMD12 end bit error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.              |
| 2    | FE_ACCE  | W    | 0h    | Force Event Auto CMD12 CRC error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.                  |
| 1    | FE_ACTO  | W    | 0h    | Force Event Auto CMD12 timeout error<br>0x0 = No effect; no interrupt<br>0x1 = Interrupt forced.              |
| 0    | FE_ACNE  | W    | 0h    | Force Event Auto CMD12 not executed.<br>0x0 = No effect; no interrupt.<br>0x1 = Interrupt forced.             |

### 18.5.1.26 SD\_ADMAES Register (offset = 254h) [reset = 0h]

SD\_ADMAES is shown in [Figure 18-62](#) and described in [Table 18-45](#).

When an ADMA Error Interrupt has occurred, the ADMA Error States field in this register holds the ADMA state and the ADMA System Address Register holds the address around the error descriptor. For recovering the error, the Host Driver requires the ADMA state to identify the error descriptor address as follows: ST\_STOP: Previous location set in the ADMA System Address register is the error descriptor address. ST\_FDS: Current location set in the ADMA System Address register is the error descriptor address. ST\_CADR: This state is never set because do not generate ADMA error in this state. ST\_TFR: Previous location set in the ADMA System Address register is the error descriptor address. In the case of a write operation, the Host Driver should use ACMD22 to get the number of written block rather than using this information, since unwritten data may exist in the Host Controller. The Host Controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid = 0) at the ST\_FDS state. In this case, ADMA Error State indicates that an error occurs at ST\_FDS state. The Host Driver may find that the Valid bit is not set in the error descriptor.

**Figure 18-62. SD\_ADMAES Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-45. SD\_ADMAES Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-3 | Reserved | R    | 0h    |   |
| 2    | LME      | W    | 0h    | ADMA Length Mismatch Error: While Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length. Total data length cannot be divided by the block length.<br>0x0 = No error<br>0x1 = Error  |
| 1-0  | AES      | R/W  | 0h    | ADMA Error State.<br>This field indicates the state of ADMA when an error occurred during an ADMA data transfer.<br>This field never indicates "10" because ADMA never stops in this state.<br>0x0(R) = ST_STOP (Stop DMA). Contents of the SYS_SDR register<br>0x1(W) = ST_STOP (Stop DMA). Points to the error descriptor.<br>0x2(R) = Never set this state. (Not used)<br>0x3(W) = ST_TFR (Transfer Data). Points to the 'next' of the error descriptor. |

### 18.5.1.27 SD\_ADMASAL Register (offset = 258h) [reset = 0h]

SD\_ADMASAL is shown in [Figure 18-63](#) and described in [Table 18-46](#).

This register holds the byte address of the executing command of the Descriptor table. The 32-bit Address Descriptor uses the lower 32 bits of this register. At the start of ADMA, the Host Driver shall set the start address of the Descriptor table.

**Figure 18-63. SD\_ADMASAL Register**

|           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31        | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADMA_A32B |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-46. SD\_ADMASAL Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description   |
|------|-----------|------|-------|---|
| 31-0 | ADMA_A32B | R/W  | 0h    | The ADMA increments this register address, which points to the next line, whenever fetching a Descriptor line.<br>When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state.<br>The Host Driver shall program the Descriptor Table on a 32-bit boundary and set the 32-bit boundary address to this register.<br>ADMA2 ignores the lower 2 bits of this register and assumes it to be 00b. |

**18.5.1.28 SD\_ADMASAH Register (offset = 25Ch) [reset = 0h]**

SD\_ADMASAH is shown in [Figure 18-64](#) and described in [Table 18-47](#).

**Figure 18-64. SD\_ADMASAH Register**

|           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31        | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADMA_A32B |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-47. SD\_ADMASAH Register Field Descriptions**

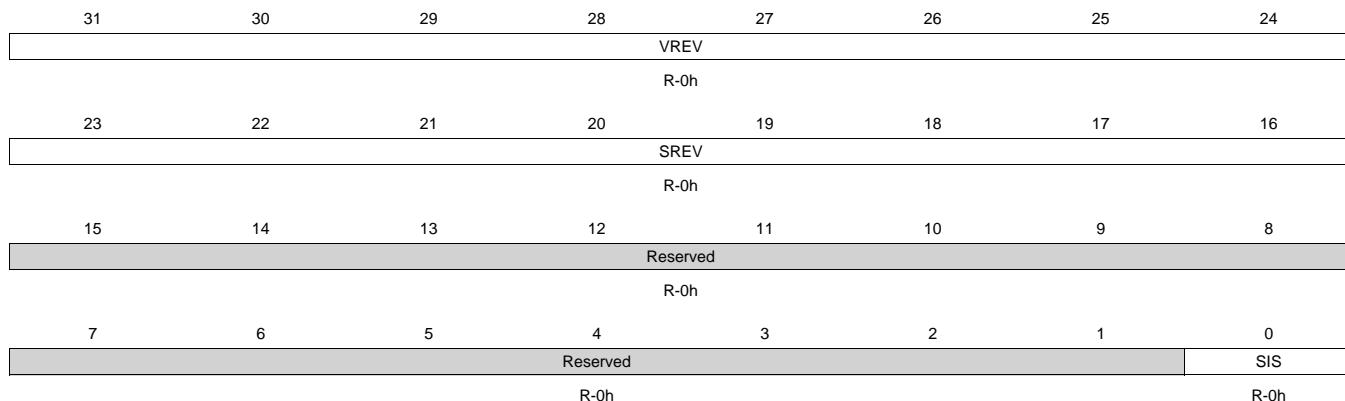
| Bit  | Field     | Type | Reset | Description |
|------|-----------|------|-------|-------------|
| 31-0 | ADMA_A32B | R/W  | 0h    | ADMA_A32B.  |

### 18.5.1.29 SD\_REV Register (offset = 2FCh) [reset = 0h]

SD\_REV is shown in [Figure 18-65](#) and described in [Table 18-48](#).

This register contains the hard coded RTL vendor revision number, the version number of SD specification compliance and a slot status bit. SD\_REV[31:16] = Host Controller Version. SD\_REV[15:0] = Slot Interrupt Status.

**Figure 18-65. SD\_REV Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 18-48. SD\_REV Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-24 | VREV     | R    | 0h    | Vendor Version Number.<br>Bits [7:4] is the major revision, bits [3:0] is the minor revision.<br>Examples: 10h for 1.0 21h for 2.1   |
| 23-16 | SREV     | R    | 0h    | Specification Version Number.<br>This status indicates the Standard SD Host Controller Specification Version.<br>The upper and lower 4-bits indicate the version.<br>0x0 = SD Host Specification Version 1.0   |
| 15-1  | Reserved | R    | 0h    |  |
| 0     | SIS      | R    | 0h    | Slot Interrupt Status.<br>This status bit indicates the inverted state of interrupt signal for the module.<br>By a power on reset or by setting a software reset for all (SD_SYSCTL[24] SRA), the interrupt signal shall be deasserted and this status shall read 0. |

## ***Universal Asynchronous Receiver/Transmitter (UART)***

---

---

This chapter describes the UART of the device.

| Topic  | Page |
|--|------|
| 19.1 Introduction .....                          | 3434 |
| 19.2 Integration .....                           | 3436 |
| 19.3 Functional Description .....                | 3440 |
| 19.4 UART/IrDA/CIR Basic Programming Model ..... | 3483 |
| 19.5 UART Registers .....                        | 3492 |

## 19.1 Introduction

### 19.1.1 UART Mode Features

The general features of the UART/IrDA module when operating in UART mode are:

- 16C750 compatibility
- Baud rate from 300 bps up to 3.6864 Mbps
- Auto-baud between 1200 bps and 115.2 Kbps
- Software/Hardware flow control
  - Programmable Xon/Xoff characters
  - Programmable Auto-RTS and Auto CTS
- Programmable serial interface characteristics
  - 5, 6, 7, or 8-bit characters
  - Even, odd, mark (always 1), space (always 0), or no parity (non-parity bit frame) bit generation and detection
  - 1, 1.5, or 2 stop bit generation
- False start bit detection
- Line break generation and detection
- Modem control functions (CTS, RTS, DSR, DTR, RI, and DCD)
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities

### 19.1.2 IrDA Mode Features

The general features of the UART/IrDA when operating in IrDA mode are:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR) and fast infrared (FIR) communications (very fast infrared (VFIR) is not supported)
- Frame formatting: addition of variable xBOF characters and EOF characters
- Uplink/downlink CRC generation/detection
- Asynchronous transparency (automatic insertion of break character)
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors
- Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection

### 19.1.3 CIR Mode Features

The general features of the UART/IrDA when operating in CIR mode are:

- Support of consumer infrared (CIR) for remote control applications
- Transmit and receive
- Free data format (supports any remote control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3 or 1/4 carrier duty cycle

### 19.1.4 Unsupported UART Features

The following UART/IrDA module features are not supported in this device.

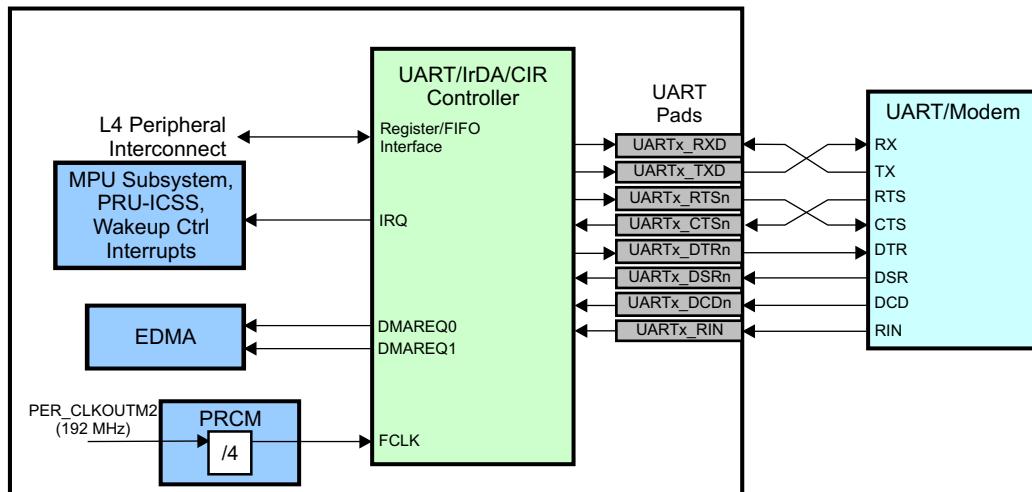
**Table 19-1. Unsupported UART Features**

| Feature                       | Reason                                      |
|-------------------------------|---|
| Full modem control on UART0   | DCD, DSR, DTR, RI not pinned-out            |
| Full modem control on UART2-5 | DCD, DSR, DTR, RI not pinned-out            |
| Device wake-up on UART1-5     | Wake-up not supported - no SWake connection |

## 19.2 Integration

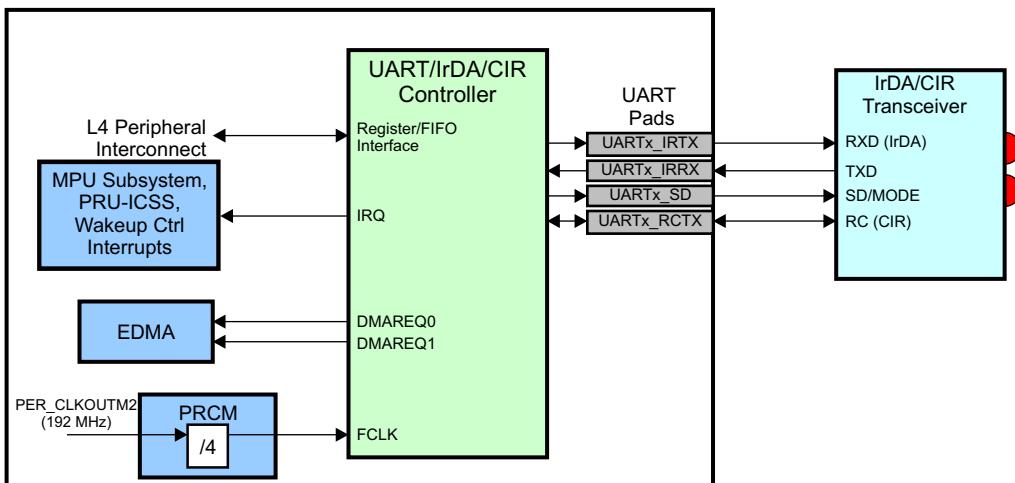
This device contains 6 instantiations of the UART/IrDA (UARTIRDAOCP) peripheral. There are six UART modules called UART0 – UART5. UART0 provides wakeup capability. Only UART 1 provides full modem control signals. All UARTs support IrDA and CIR modes and RTS/CTS flow control (subject to pin muxing configuration). [Figure 19-1](#) shows an example of system connectivity using UART communication with hardware handshake.

**Figure 19-1. UART/IrDA Module — UART Application**



[Figure 19-2](#) shows an example of system connectivity using infrared communication with remote control (consumer infrared).

**Figure 19-2. UART/IrDA Module — IrDA/CIR Application**



### 19.2.1 UART Connectivity Attributes

The general connectivity attributes for each of the UART modules are shown in [Table 19-2](#) and [Table 19-3](#).

**Table 19-2. UART0 Connectivity Attributes**

| Attributes   | Type   |
|--------------|--|
| Power Domain | Wake-Up Domain   |
| Clock Domain | PD_WKUP_L4_WKUP_GCLK (OCP)<br>PD_WKUP_UART0_GFCLK (Func) |

**Table 19-2. UART0 Connectivity Attributes (continued)**

| Attributes          | Type  |
|---------------------|---|
| Reset Signals       | WKUP_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle / Wakeup   |
| Interrupt Requests  | 1 interrupt to MPU Subsystem (UART0INT), PRU-ICSS (nirq) and WakeM3 |
| DMA Requests        | 2 DMA requests to EDMA (TX – UTXEVT0, RX – URXEVT0)                 |
| Physical Address    | L4 Wakeup slave port  |

**Table 19-3. UART1–5 Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain   |
| Clock Domain        | PD_PER_L4LS_GCLK (OCP)<br>PD_PER_UART_GFCLK (Func)  |
| Reset Signals       | PER_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle  |
| Interrupt Requests  | UART1-2<br>1 interrupt per instance to MPU Subsystem (UART1INT, UART2INT) and PRU-ICSS (nirq)<br>UART3-5<br>1 interrupt per instance to only MPU Subsystem (UART3INT, UART4INT, UART5INT) |
| DMA Requests        | 2 DMA requests per instance to EDMA (TX – UTXEVTx, RX – URXEVTx)  |
| Physical Address    | L4 Peripheral slave port  |

### 19.2.2 *UART Clock and Reset Management*

The UART modules use separate functional and bus interface clocks.

**Table 19-4. UART0 Clock Signals**

| Clock Signal     | Max Freq | Reference / Source | Comments             |
|------------------|----------|--------------------|----------------------|
| CLK              |          |                    |                      |
| Interface clock  | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_wkup_l4_wkup_gclk |
| From PRCM        |          |                    |                      |
| FCLK             |          |                    |                      |
| Functional clock | 48 MHz   | PER_CLKOUTM2 / 4   | pd_wkup_uart0_gfclk  |
| from PRCM        |          |                    |                      |

**Table 19-5. UART1–5 Clock Signals**

| Clock Signal     | Max Freq | Reference / Source | Comments                       |
|------------------|----------|--------------------|--------------------------------|
| CLK              |          |                    |                                |
| Interface clock  | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>From PRCM  |
| FCLK             |          |                    |                                |
| Functional clock | 48 MHz   | PER_CLKOUTM2 / 4   | pd_per_uart_gfclk<br>From PRCM |

For UART operation, the functional clock is used to produce a baud rate up to 3.6M bits/s. [Table 19-6](#) lists the supported baud rates, the requested divider, and the corresponding error versus the standard baud rate.

**Table 19-6. UART Mode Baud and Error Rates**

| Baud rate | Over sampling | Divisor | Error (%) |
|-----------|---------------|---------|-----------|
| 300       | 16            | 10000   | 0         |
| 600       | 16            | 5000    | 0         |
| 1200      | 16            | 2500    | 0         |
| 2400      | 16            | 1250    | 0         |
| 4800      | 16            | 625     | 0         |
| 9600      | 16            | 313     | 0.16      |
| 14400     | 16            | 208     | 0.16      |
| 19200     | 16            | 156     | 0.16      |
| 28800     | 16            | 104     | 0.16      |
| 38400     | 16            | 78      | 0.16      |
| 57600     | 16            | 52      | 0.16      |
| 115200    | 16            | 26      | 0.16      |
| 230400    | 16            | 13      | 0.16      |
| 460800    | 13            | 8       | 0.16      |
| 921600    | 13            | 4       | 0.16      |
| 1843200   | 13            | 2       | 0.16      |
| 3000000   | 16            | 1       | 0         |
| 3686400   | 13            | 1       | 0.16      |

For IrDA operation, the internal functional clock divisor allows generation of SIR, MIR, or FIR baud rates as shown in [Table 19-7](#).

**Table 19-7. IrDA Mode Baud and Error Rates**

| Baud rate | IR mode | Encoding | Divisor | Error (%) |
|-----------|---------|----------|---------|-----------|
| 2400      | SIR     | 3/16     | 1250    | 0         |
| 9600      | SIR     | 3/16     | 312     | 0.16      |
| 19200     | SIR     | 3/16     | 156     | 0.16      |
| 38400     | SIR     | 3/16     | 78      | 0.16      |
| 57600     | SIR     | 3/16     | 52      | 0.16      |
| 115200    | SIR     | 3/16     | 26      | 0.16      |
| 576000    | MIR     | 1/4      | 2       | 0         |
| 1152000   | MIR     | 1/4      | 1       | 0         |
| 4000000   | FIR     | 4PPM     | 1       | 0         |

### 19.2.3 UART Pin List

The UART interface pins are listed in [Table 19-8](#). Pin functionality depends on the selected operating mode of the module.

**Table 19-8. UART Pin List**

| Pin                       | Type | Description                      |
|---------------------------|------|----------------------------------|
| UARTx_RXD / IRRX / RCRX   | I    | UART / IrDA / CIR Receive Data   |
| UARTx_TXD / IRTX / RCTX   | OZ   | UART / IrDA / CIR Transmit Data  |
| UARTx_RTSD / SD           | OZ   | UART Request to Send / IrDA Mode |
| UARTx_CTSn                | I    | UART Clear to Send               |
| UARTx_DTRn <sup>(1)</sup> | OZ   | UART Data Terminal Ready         |
| UARTx_DSRn <sup>(1)</sup> | I    | UART Data Set Ready              |
| UARTx_DCDn <sup>(1)</sup> | I    | UART Data Carrier Detect         |
| UARTx_RIN <sup>(1)</sup>  | 1    | UART Ring Indicator              |

<sup>(1)</sup> UART1 only

The UART module can operate in three different modes based on the MODE\_SELECT bits. The signal muxing based on these mode bits is shown in [Table 19-9](#).

**Table 19-9. UART Muxing Control**

| UARTx_TXD / IRTX / RCTX Function | UARTx_RXD / IRRX / RCRX Function | UARTx_RTSD / SD Function | UARTx_CTSn Function | Mode                 |
|----------------------------------|----------------------------------|--------------------------|---------------------|----------------------|
| TXD                              | RXD                              | RTSD                     | CTSn                | UART                 |
| IRTX                             | IRRX                             | SD                       | not used            | IrDA (SIR, MIR, FIR) |
| RCTX                             | RCRX                             | SD                       | not used            | CIR                  |

## 19.3 Functional Description

### 19.3.1 Block Diagram

The UART/IrDA/CIR module can be divided into three main blocks:

- FIFO management
- Mode selection
- Protocol formatting

FIFO management is common to all functions and enables the transmission and reception of data from the host processor point of view.

There are two modes:

- Function mode: Routes the data to the chosen function (UART, IrDA, or CIR) and enables the mechanism corresponding to the chosen function
- Register mode: Enables conditional access to registers

For more information about mode configuration, see [Section 19.3.7, Mode Selection](#).

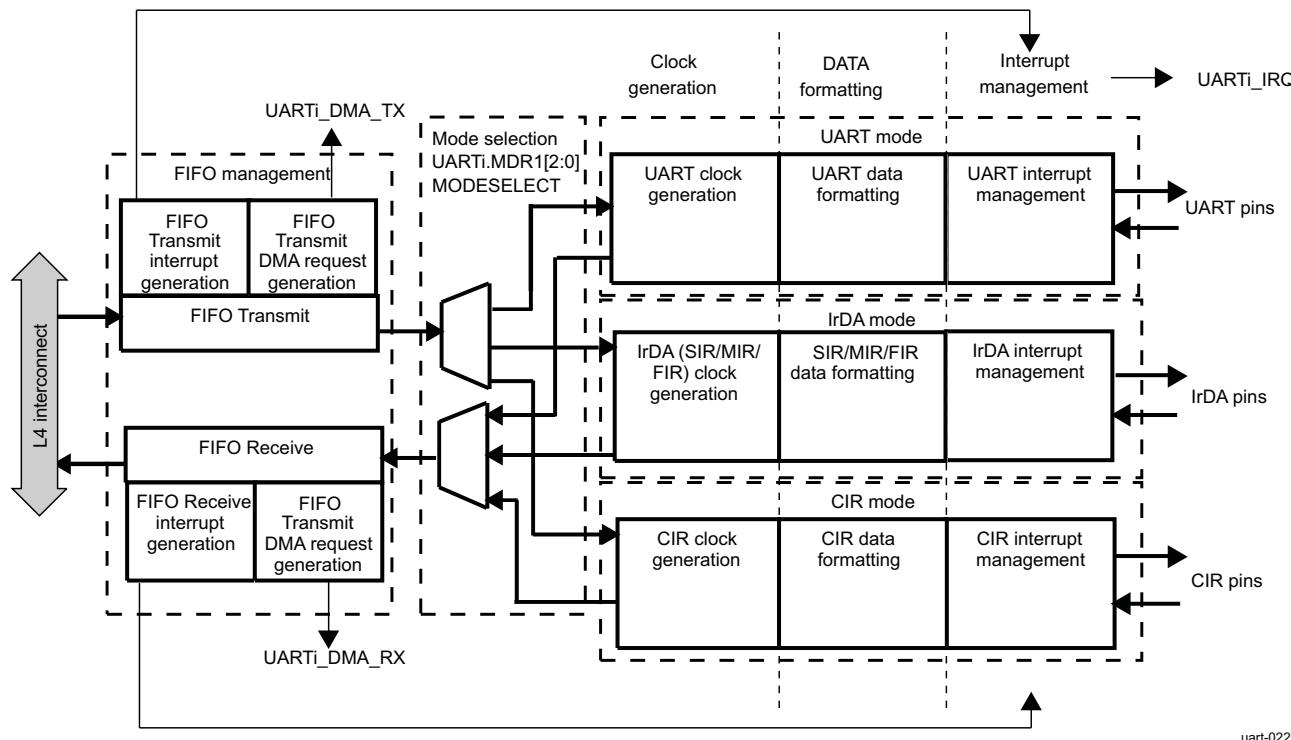
Protocol formatting has three subcategories:

- Clock generation: The 48-MHz input clock generates all necessary clocks.
- Data formatting: Each function uses its own state-machine that is responsible for the transition between FIFO data and frame data associated with it.
- Interrupt management: Different interrupt types are generated depending on the chosen function:
  - UART mode interrupts: Seven interrupts prioritized in six different levels
  - IrDA mode interrupts: Eight interrupts. The interrupt line is activated when any interrupt is generated (there is no priority).
  - CIR mode interrupts: A subset of existing IrDA mode interrupts is used.

In each mode, when an interrupt is generated, the `UART_IIR` register indicates the interrupt type.

In parallel with these functional blocks, a power-saving strategy exists for each function.

[Figure 19-3](#) is the UART/IrDA/CIR block diagram.

**Figure 19-3. UART/IrDA/CIR Functional Specification Block Diagram**


uart-022

### 19.3.2 Clock Configuration

Each UART uses a 48-MHz functional clock for its logic and to generate external interface signals. Each UART uses an interface clock for register accesses. The PRCM module generates and controls all these clocks (for more information, see *Clock Domain Module Attributes*, in [Chapter 8, Power, Reset, and Clock Management](#)).

The idle and wake-up processes use a handshake protocol between the PRCM and the UART (for a description of the protocol, see *Module-Level Clock Management* in [Chapter 8, Power, Reset, and Clock Management](#)). The `UARTi.UART_SYSC[4:3] IDLEMODE` bit field controls UART idle mode.

### 19.3.3 Software Reset

The `UARTi.UART_SYSC[1] SOFTRESET` bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to hardware reset.

### 19.3.4 Power Management

#### 19.3.4.1 UART Mode Power Management

##### 19.3.4.1.1 Module Power Saving

In UART modes, sleep mode is enabled by setting the `UARTi.UART_IER[4] SLEEP_MODE` bit to 1 (when the `UARTi.UART_EFR[4] ENHANCED_EN` bit is set to 1).

Sleep mode is entered when all the following conditions exist:

- The serial data input line, `uarti_rx`, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- The only pending interrupts are THR interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set to modem mode. Therefore, even if the UART has no key role functionally, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked by these clocks, this greatly reduces power consumption. The module wakes up when a change is detected on the uarti\_rx line, when data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

An interrupt can be generated on a wake-up event by setting the UARTi.UART\_SCR[4] RX\_CTS\_WU\_EN bit to 1. To understand how to manage the interrupt, see [Section 19.3.5.2, Wake-Up Interrupt](#).

---

**NOTE:** There must be no writing to the divisor latches, UARTi.UART\_DLL and UARTi.UART\_DLH, to set the baud clock (BCLK) while in sleep mode. It is advisable to disable sleep mode using the UARTi.UART\_IER[4] SLEEP\_MODE bit before writing to the UARTi.UART\_DLL register or the UARTi.UART\_DLH register.

---

#### 19.3.4.1.2 System Power Saving

Sleep and auto-idle modes are embedded power-saving features. Power-reduction techniques can be applied at the system level by shutting down certain internal clock and power domains of the device.

The UART supports an idle req/idle ack handshaking protocol used at the system level to shut down the UART clocks in a clean and controlled manner and to switch the UART from interrupt-generation mode to wake-up generation mode for unmasked events (see the UARTi.UART\_SYSC[2] ENAWAKEUP bit and the UARTi.UART\_WER register).

For more information, see *Module Level Clock Management* in [Chapter 8, Power, Reset, and Clock Management](#).

#### 19.3.4.2 IrDA/CIR Mode Power Management

##### 19.3.4.2.1 Module Power Saving

In IrDA/CIR modes, sleep mode is enabled by setting the UARTi.MDR[3] IR\_SLEEP bit to 1.

Sleep mode is entered when all the following conditions exist:

- The serial data input line, uarti.rx\_irrx, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when a change is detected on the uarti\_rx\_irrx line or when data is written to the TX FIFO.

##### 19.3.4.2.2 System Power Saving

System power saving for the IrDA/CIR mode has the same function as for the UART mode (see [Section 19.3.4.1.2, System Power Saving](#)).

#### 19.3.4.3 Local Power Management

[Table 19-10](#) describes power-management features available for the UART.

---

**NOTE:** For information about source clock gating and sleep/wake-up transitions description, see *Module-Level Clock Management* in [Chapter 8, Power, Reset, and Clock Management](#).

---

**Table 19-10. Local Power-Management Features**

| Feature                | Registers               | Description  |
|------------------------|-------------------------|--|
| Clock autogating       | UART_SYSC[0] AUTOIDLE   | This bit allows local power optimization in the module by gating the UARTI_ICLK clock on interface activity or gating the UARTI_FCLK clock on internal activity. |
| Slave idle modes       | UART_SYSC[4:3] IDLEMODE | Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available   |
| Clock activity         | N/A                     | Feature not available  |
| Master standby modes   | N/A                     | Feature not available  |
| Global wake-up enable  | UART_SYSC[2] ENAWAKEUP  | This bit enables the wake-up feature at module level.  |
| Wake-Up sources enable | N/A                     | Feature not available  |

### 19.3.5 Interrupt Requests

The UART IrDA CIR module generates interrupts. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR). The UART, IrDA, and CIR modes have different interrupts in the UART IrDA CIR module and therefore have different IER and IIR mappings according to the selected mode.

#### 19.3.5.1 UART Mode Interrupt Management

##### 19.3.5.1.1 *UART Interrupts*

UART mode includes seven possible interrupts prioritized to six levels.

When an interrupt is generated, the interrupt identification register (UARTI.UART\_IIR) sets the UARTI.UART\_IIR[0] IT\_PENDING bit to 0 to indicate that an interrupt is pending, and indicates the type of interrupt through the UARTI.UART\_IIR[5:1] bit field. [Table 19-11](#) summarizes the interrupt control functions.

**Table 19-11. UART Mode Interrupts**

| UART_IIR[5:0] | Priority Level | Interrupt Type                             | Interrupt Source   | Interrupt Reset Method   |
|---------------|----------------|--|--|--|
| 000001        | None           | None                                       | None   | None   |
| 000110        | 1              | Receiver line status                       | OE, FE, PE, or BI errors occur in characters in the RX FIFO.                 | FE, PE, BI: Read the UART_RHR register. OE: Read the UART_LSR register.                                    |
| 001100        | 2              | RX time-out                                | Stale data in RX FIFO  | Read the UART_RHR register.  |
| 000100        | 2              | RHR interrupt                              | DRDY (data ready) (FIFO disable)   | Read the UART_RHR register until the interrupt condition disappears.                                       |
|               |                |  | RX FIFO above trigger level (FIFO enable)                                    |  |
| 000010        | 3              | THR interrupt                              | TFE (UART_THR empty) (FIFO disable)  | Write to the UART_THR until the interrupt condition disappears.  |
|               |                |  | TX FIFO below trigger level (FIFO enable)                                    |  |
| 000000        | 4              | Modem status                               | See the UART_MSR register.   | Read the UART_MSR register.  |
| 010000        | 5              | XOFF interrupt/special character interrupt | Receive XOFF characters/special character                                    | Receive XON character(s), if XOFF interrupt/read of the UART_IIR register, if special character interrupt. |
| 100000        | 6              | CTS, RTS, DSR                              | RTS pin or CTS pin or DSR change state from active (low) to inactive (high). | Read the UART_IIR register.  |

For the receiver-line status interrupt, the RX\_FIFO\_STS bit (UARTi.UART\_LSR[7]) generates the interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special character detection caused the interrupt, the interrupt is cleared by a read of the UARTi.UART\_IIR register.

### 19.3.5.2 Wake-Up Interrupt

Wake-up interrupt is a special interrupt that works differently from other interrupts. This interrupt is enabled when the UARTi.UART\_SCR[4] RXCTSDSRWAKEUPENABLE bit is set to 1. The UARTi.UART\_IIR register is not modified when this occurs; the UARTi.UART\_SSR[1] RXCTSDSRWAKEUPSTS bit must be checked to detect a wake-up event.

When a wake-up interrupt occurs, it can be cleared only by resetting the UARTi.UART\_SCR[4] RXCTSDSRWAKEUPENABLE bit. This bit must be re-enabled (set to 1) after the current wake-up interrupt event is processed to detect the next incoming wake-up event.

A wake-up interrupt can also occur if the WER[7] TXWAKEUPEN bit is set to 1 and one of the following occurs:

- THR interrupt occurred if it is enabled (omitted if TX DMA request is enabled).
- TX DMA request occurred if it is enabled.
- TX\_STATUS\_IT occurred if it is enabled (only IrDA and CIR modes). Cannot be used with THR interrupt.

#### CAUTION

Wake-Up interface implementation in IrDA mode is based on the UARTi\_SIDLEACK low-to-high transition instead of the UARTi\_SIDLEACK state.

This does not ensure wake-up event generation as expected when configured in smart-idle mode, and the system wakes up for a short period.

### 19.3.5.3 IrDA Mode Interrupt Management

#### 19.3.5.3.1 IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UARTi.UART\_IER). The interrupt status of the device can be checked by reading the interrupt identification register (UARTi.UART\_IIR).

UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UARTi.UART\_IER and UARTi.UART\_IIR mappings, depending on the selected mode.

IrDA modes have eight possible interrupts (see [Table 19-12](#)). The interrupt line is activated when any interrupt is generated (there is no priority).

**Table 19-12. IrDA Mode Interrupts**

| UART_IIR Bit | Interrupt Type | Interrupt Source                          | Interrupt Reset Method   |
|--------------|----------------|---|--|
| 0            | RHR interrupt  | DRDY (data ready) (FIFO disable)          | Read the UART_RHR register until the interrupt condition disappears. |
|              |                | RX FIFO above trigger level (FIFO enable) |  |
| 1            | THR interrupt  | TFE (UART_THR empty) (FIFO disable)       | Write to the UART_THR until the interrupt condition disappears.      |
|              |                | TX FIFO below trigger level (FIFO enable) |  |

**Table 19-12. IrDA Mode Interrupts (continued)**

| UART_IIR Bit | Interrupt Type                 | Interrupt Source   | Interrupt Reset Method   |
|--------------|--------------------------------|--|--|
| 2            | Last byte in RX FIFO           | Last byte of frame in RX FIFO is available to be read at the RHR port.   | Read the UART_RHR register.  |
| 3            | RX overrun                     | Write to the UART_RHR register when the RX FIFO is full.   | Read UART_RESUME register.   |
| 4            | Status FIFO interrupt          | Status FIFO triggers level reached.  | Read STATUS FIFO.  |
| 5            | TX status                      | 1. UART_THR empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error.<br>OR<br>2. Transmission of the last bit of the IrDA frame completed successfully. | 1. Read the UART_RESUME register.<br>OR<br>2. Read the UART_IIR register.  |
| 6            | Receiver line status interrupt | CRC, ABORT, or frame-length error is written into the STATUS FIFO.   | Read the STATUS FIFO (read until empty - maximum of eight reads required). |
| 7            | Received EOF                   | Received end-of-frame  | Read the UART_IIR register.  |

### 19.3.5.4 CIR Mode Interrupt Management

#### 19.3.5.4.1 CIR Interrupts

The CIR function generates interrupts that can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UARTi.UART\_IER). The interrupt status of the device can be checked by reading the interrupt identification register (UARTi.UART\_IIR).

UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UARTi.UART\_IER and UARTi.UART\_IIR mappings, depending on the selected mode.

Table 19-13 lists the interrupt modes to be maintained. In CIR mode, the sole purpose of the UARTi.UART\_IIR[5] bit is to indicate that the last bit of infrared data was passed to the uart\_cts\_rctx pin.

**Table 19-13. CIR Mode Interrupts**

| UART_IIR Bit Number | Interrupt Type   | Interrupt Source  | Interrupt Reset Method   |
|---------------------|------------------|---|--|
| 0                   | RHR interrupt    | DRDY (data ready) (FIFO disable)<br><br>RX FIFO above trigger level (FIFO enable)       | Read UART_RHR until interrupt condition disappears.                      |
| 1                   | THR interrupt    | TFE (UART_THR empty) (FIFO disable)<br><br>TX FIFO below trigger level (FIFO enable)    | Write to the UART_THR register until the interrupt condition disappears. |
| 2                   | RX_STOP_IT       | Receive stop interrupt (depending on value set in the BOF Length Register (UART_EBLR)). | Read IIR   |
| 3                   | RX overrun       | Write to RHR when RX FIFO is full.  | Read RESUME register.  |
| 4                   | N/A for CIR mode | N/A for CIR mode  | N/A for CIR mode   |
| 5                   | TX status        | Transmission of the last bit of the frame is complete successfully.                     | Read the UART_IIR register.  |
| 6                   | N/A for CIR mode | N/A for CIR mode  | N/A for CIR mode   |
| 7                   | N/A for CIR mode | N/A for CIR mode  | N/A for CIR mode   |

### 19.3.6 FIFO Management

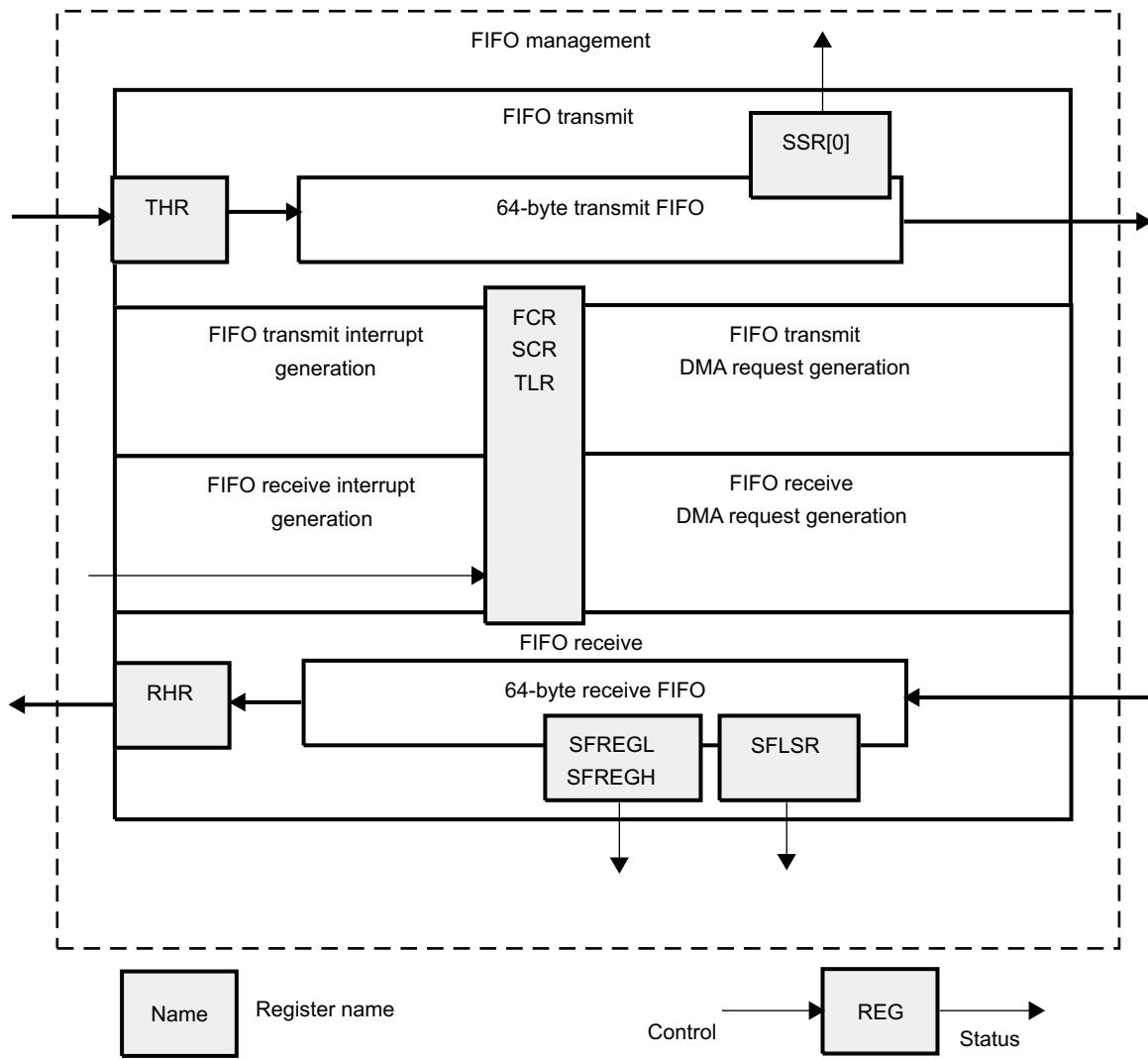
The FIFO is accessed by reading and writing the `UARTi.UART_RHR` and `UARTi.UART_THR` registers. Parameters are controlled using the FIFO control register (`UARTi.UART_FCR`) and supplementary control register (`UARTi.UART_SCR`). Reading the `UARTi.UART_SSR[0]` TX\_FIFO\_FULL bit at 1 means the FIFO is full.

The `UARTi.UART_TLR` register controls the FIFO trigger level, which enables DMA and interrupt generation. After reset, transmit (TX) and receive (RX) FIFOs are disabled; thus, the trigger level is the default value of 1 byte. [Figure 19-4](#) shows the FIFO management registers.

**NOTE:** Data in the `UARTi.UART_RHR` register is not overwritten when an overflow occurs.

**NOTE:** The `UARTi.UART_SFLSR`, `UARTi.UART_SFREGL`, and `UARTi.UART_SFREGH` status registers are used in IrDA mode only. For use, see [Section 19.3.8.2.6, IrDA Data Formatting](#).

**Figure 19-4. FIFO Management Registers**



uart-023

### 19.3.6.1 FIFO Trigger

#### 19.3.6.1.1 Transmit FIFO Trigger

Table 19-14 lists the TX FIFO trigger level settings.

**Table 19-14. TX FIFO Trigger Level Setting Summary**

| UART_SCR[6] | UART_TLR[3:0] | TX FIFO Trigger Level   |
|-------------|---------------|---|
| 0           | = 0x0         | Defined by the UARTi.UART_FCR[5:4] TX_FIFO_TRIG bit field (8,16, 32, or 56 spaces)  |
| 0           | != 0x0        | Defined by the UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA bit field (from 4 to 60 spaces with a granularity of 4 spaces)  |
| 1           | Value         | Defined by the concatenated value of TX_FIFO_TRIG_DMA and TX_FIFO_TRIG (from 1 to 63 spaces with a granularity of 1 space)<br><br><b>Note:</b> The combination of TX_FIFO_TRIG_DMA = 0x0 and TX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of one space required). All zeros result in unpredictable behavior. |

#### 19.3.6.1.2 Receive FIFO Trigger

Table 19-15 lists the RX FIFO trigger level settings.

**Table 19-15. RX FIFO Trigger Level Setting Summary**

| UART_SCR[7] | UART_TLR[7:4] | RX FIFO Trigger Level   |
|-------------|---------------|---|
| 0           | = 0x0         | Defined by the UARTi.UART_FCR[7:6] RX_FIFO_TRIG bit field (8,16, 56, or 60 characters)  |
| 0           | != 0x0        | Defined by the UARTi.UART_TLR[7:4] RX_FIFO_TRIG_DMA bit field (from 4 to 60 characters with a granularity of 4 characters)  |
| 1           | Value         | Defined by the concatenated value of RX_FIFO_TRIG_DMA and RX_FIFO_TRIG (from 1 to 63 characters with a granularity of 1 character)<br><br><b>Note:</b> The combination of RX_FIFO_TRIG_DMA = 0x0 and RX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of one character required). All zeros result in unpredictable behavior. |

The receive threshold is programmed using the UARTi.UART\_TCR[7:4] RX\_FIFO\_TRIG\_START and UARTi.UART\_TCR[3:0] RX\_FIFO\_TRIG\_HALT bit fields:

- Trigger levels from 0 to 60 bytes are available with a granularity of 4 (trigger level = 4 x [4-bit register value]).
- To ensure correct device operation, ensure that RX\_FIFO\_TRIG\_HALT RX\_FIFO\_TRIG when auto-RTS is enabled.

$$\text{Delay} = [4 + 16 \times (1 + \text{CHAR\_LENGTH} + \text{Parity} + \text{Stop} 0.5)] \times \text{Baud\_rate} + 4 \times \text{FCLK}$$

---

**NOTE:** The RTS signal is deasserted after the UART module receives the data over RX\_FIFO\_TRIG\_HALT. Delay means how long the UART module takes to deassert the RTS signal after reaching RX\_FIFO\_TRIG\_HALT.

---

- In FIFO interrupt mode with flow control, ensure that the trigger level to HALT transmission is greater than or equal to the RX FIFO trigger level (the UARTi.UART\_TCR[7:4] RX\_FIFO\_TRIG\_START bit field or the UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG bit field); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent when a byte is received.

### 19.3.6.2 FIFO Interrupt Mode

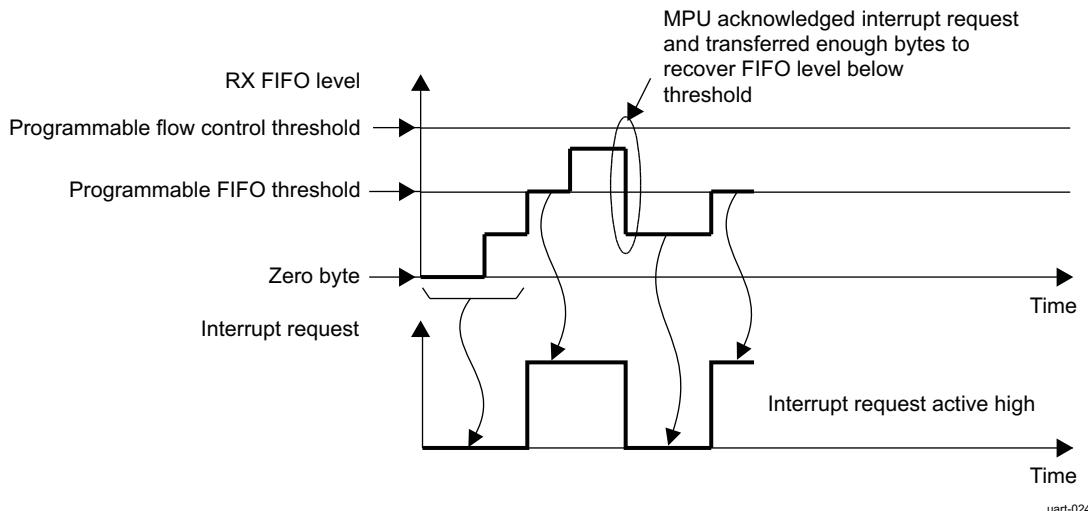
In FIFO interrupt mode (the FIFO control register  $\text{UART}_i.\text{UART\_FCR}[0]$  FIFO\_EN bit is set to 1 and relevant interrupts are enabled by the  $\text{UART}_i.\text{UART\_IER}$  register), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the RX/TX FIFO threshold (the  $\text{UART}_i.\text{UART\_TLR}[7:4]$  RX\_FIFO\_TRIG\_DMA and  $\text{UART}_i.\text{UART\_TLR}[3:0]$  TX\_FIFO\_TRIG\_DMA bit fields or the  $\text{UART}_i.\text{UART\_FCR}[7:6]$  RX\_FIFO\_TRIG and  $\text{UART}_i.\text{UART\_FCR}[5:4]$  TX\_FIFO\_TRIG bit fields, respectively) is reached.

The interrupt signals instruct the MPU to transfer data to the destination (from the UART in receive mode and/or from any source to the UART FIFO in transmit mode).

When UART flow control is enabled with interrupt capabilities, the UART flow control FIFO threshold (the  $\text{UART}_i.\text{UART\_TCR}[3:0]$  RX\_FIFO\_TRIG\_HALT bit field) must be greater than or equal to the RX FIFO threshold.

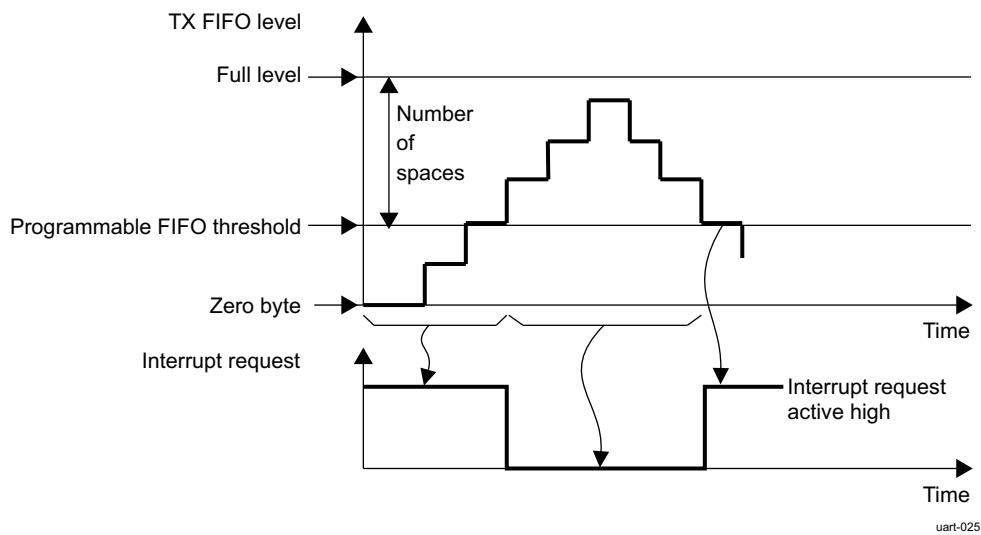
[Figure 19-5](#) shows the generation of the RX FIFO interrupt request.

**Figure 19-5. RX FIFO Interrupt Request Generation**



In receive mode, no interrupt is generated until the RX FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the MPU has handled enough bytes to put the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

[Figure 19-6](#) shows the generation of the TX FIFO interrupt request.

**Figure 19-6. TX FIFO Interrupt Request Generation**


In transmit mode, an interrupt request is automatically asserted when the TX FIFO is empty. This request is deasserted when the TX FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements is transmitted to go below the TX FIFO threshold.

#### 19.3.6.3 FIFO Polled Mode Operation

In FIFO polled mode (the `UARTi.UART_FCR[0]` FIFO\_EN bit is set to 0 and the relevant interrupts are disabled by the `UARTi.UART_IER` register), the status of the receiver and transmitter can be checked by polling the line status register (`UARTi.UART_LSR`).

This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically determined by sending interrupts to the MPU.

#### 19.3.6.4 FIFO DMA Mode Operation

Although DMA operation includes four modes (DMA modes 0 through 3), assume that mode 1 is used. (Mode 2 and mode 3 are legacy modes that use only one DMA request for each module.)

In mode 2, the remaining DMA request is used for RX. In mode 3, the remaining DMA request is used for TX.

DMA requests in mode 2 and mode 3 use the following signals:

- S\_DMA\_48
- S\_DMA\_50
- S\_DMA\_52/D\_DMA\_10
- S\_DMA\_54

The following signals are not used by the module in mode 2 and mode 3:

- S\_DMA\_49
- S\_DMA\_51
- S\_DMA\_53/D\_DMA\_11
- S\_DMA\_55

These signals can be selected as follows:

- When the `UARTi.UART_SCR[0]` DMA\_MODE\_CTL bit is set to 0, setting the `UARTi.UART_FCR[3]` DMA\_MODE\_CTL bit to 0 enables DMA mode 0. Setting the DMA\_MODE\_CTL bit to 1 enables DMA mode 1.
- When the DMA\_MODE\_CTL bit is set to 1, the `UARTi.UART_FCR[2:1]` DMA\_MODE\_2 bit field

determines DMA mode 0 to mode 3 based on the supplementary control register (UART\_SCR) description.

For example:

- If no DMA operation is desired, set the DMA\_MODE\_CTL bit to 1 and the DMA\_MODE\_2 bit field to 0x0. (The DMA\_MODE bit is discarded.)
- If DMA mode 1 is desired, set the DMA\_MODE\_CTL bit to 0 and the DMA\_MODE bit to 1, or set the DMA\_MODE\_CTL bit to 1 and the DMA\_MODE\_2 bit field to 01. (The DMA\_MODE bit is discarded.)

If the FIFOs are disabled (the UARTi.UART\_FCR[0] FIFO\_EN bit is set to 0), the DMA occurs in single-character transfers.

When DMA mode 0 is programmed, the signals associated with DMA operation are not active.

Depending on UART\_MDR3[2] SET\_DMA\_TX\_THRESHOLD, the threshold can be programmed different ways:

- SET\_TX\_DMA\_THRESHOLD = 1:

The threshold value will be the value of the UART\_TX\_DMA\_THRESHOLD register. If SET\_TX\_DMA\_THRESHOLD + TX trigger spaces 64, then the default method of threshold is used: threshold value = TX FIFO size.

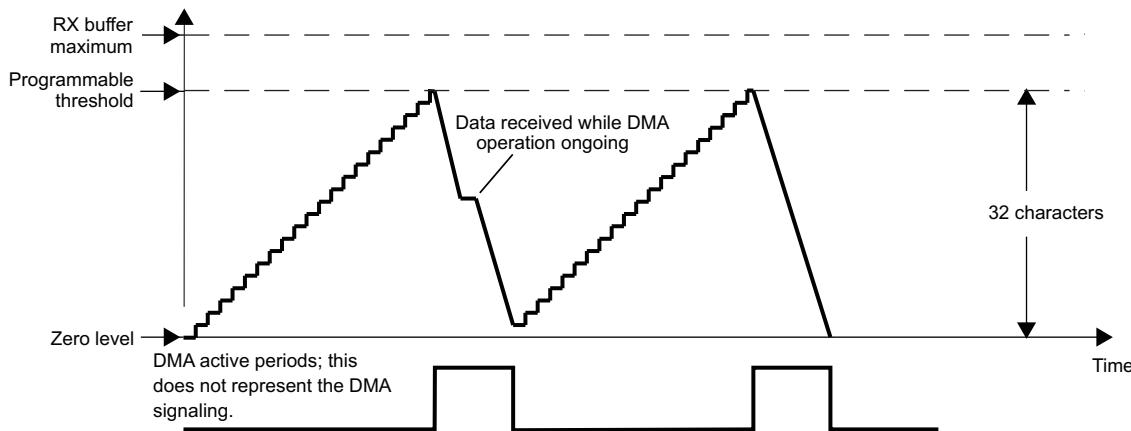
- SET\_TX\_DMA\_THRESHOLD = 0:

The threshold value = TX FIFO size - TX trigger space. The TX DMA line is asserted if the TX FIFO level is lower than the threshold. It remains asserted until TX trigger spaces number of bytes are written into the FIFO. The DMA line is then deasserted and the FIFO level is compared with the threshold value.

#### 19.3.6.4.1 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 19-7 through Figure 19-10 show the supported DMA operations.

**Figure 19-7. Receive FIFO DMA Request Generation (32 Characters)**

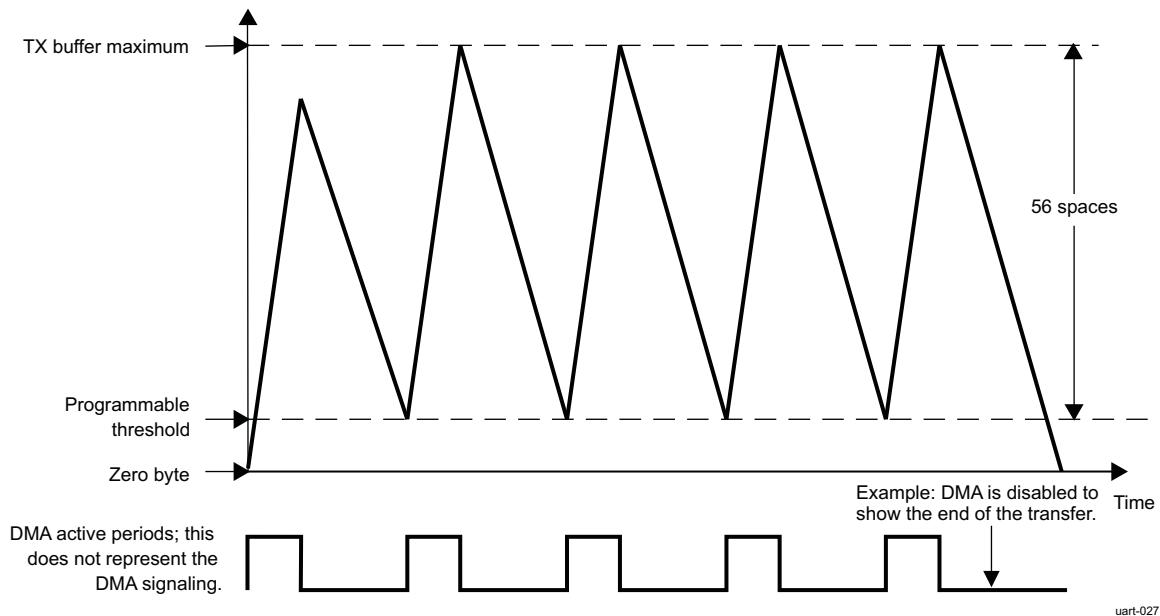


uart-026

In receive mode, a DMA request is generated when the RX FIFO reaches its threshold level defined in the trigger level register (UARTi.UART\_TLR). This request is deasserted when the number of bytes defined by the threshold level is read by the sDMA.

In transmit mode, a DMA request is automatically asserted when the TX FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the UARTi.UART\_TLR register is written by the sDMA. If an insufficient number of characters is written, the DMA request stays active.

**Figure 19-8. Transmit FIFO DMA Request Generation (56 Spaces)**

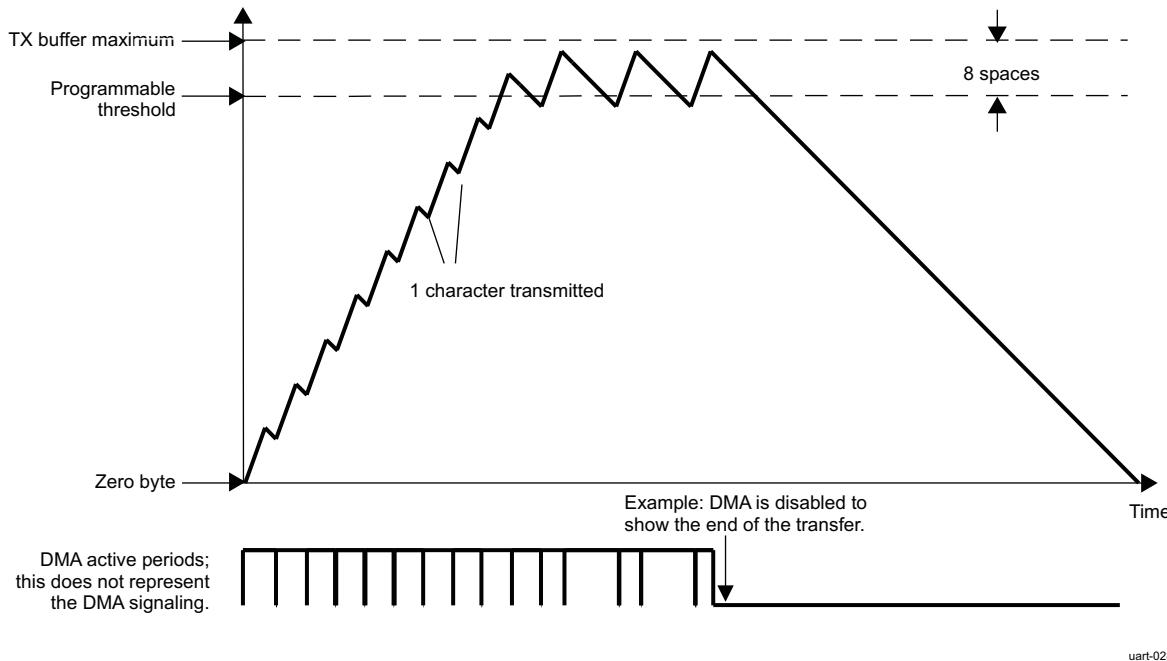


The DMA request is again asserted if the FIFO can receive the number of bytes defined by the `UARTi.UART_TLR` register.

The threshold can be programmed in a number of ways. [Figure 19-8](#) shows a DMA transfer operating with a space setting of 56 that can arise from using the auto settings in the `UARTi.UART_FCR[5:4]` `TX_FIFO_TRIG` bit field or the `UARTi.UART_TLR[3:0]` `TX_FIFO_TRIG_DMA` bit field concatenated with the `TX_FIFO_TRIG` bit field.

The setting of 56 spaces in the `UART/IrDA/CIR` module must correlate with the settings of the sDMA so that the buffer does not overflow (program the DMA request size of the LH controller to equal the number of spaces in the `UART/IrDA/CIR` module).

[Figure 19-9](#) shows an example with eight spaces to show the buffer level crossing the space threshold. The LH DMA controller settings must correspond to those of the `UART/IrDA/CIR` module.

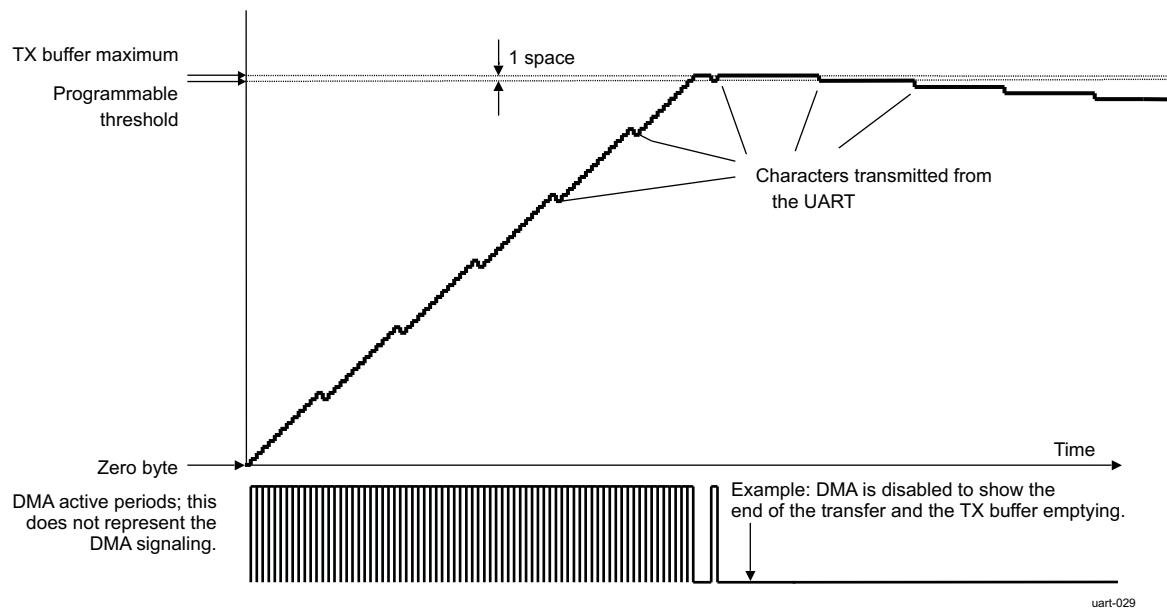
**Figure 19-9. Transmit FIFO DMA Request Generation (8 Spaces)**


uart-028

The next example shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer (see [Figure 19-10](#)). The buffer is filled faster than the baud rate at which data is transmitted to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer.

On two occasions, the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers.

The DMA settings must correspond to the system LH DMA controller settings to ensure correct operation of this logic.

**Figure 19-10. Transmit FIFO DMA Request Generation (1 Space)**


uart-029

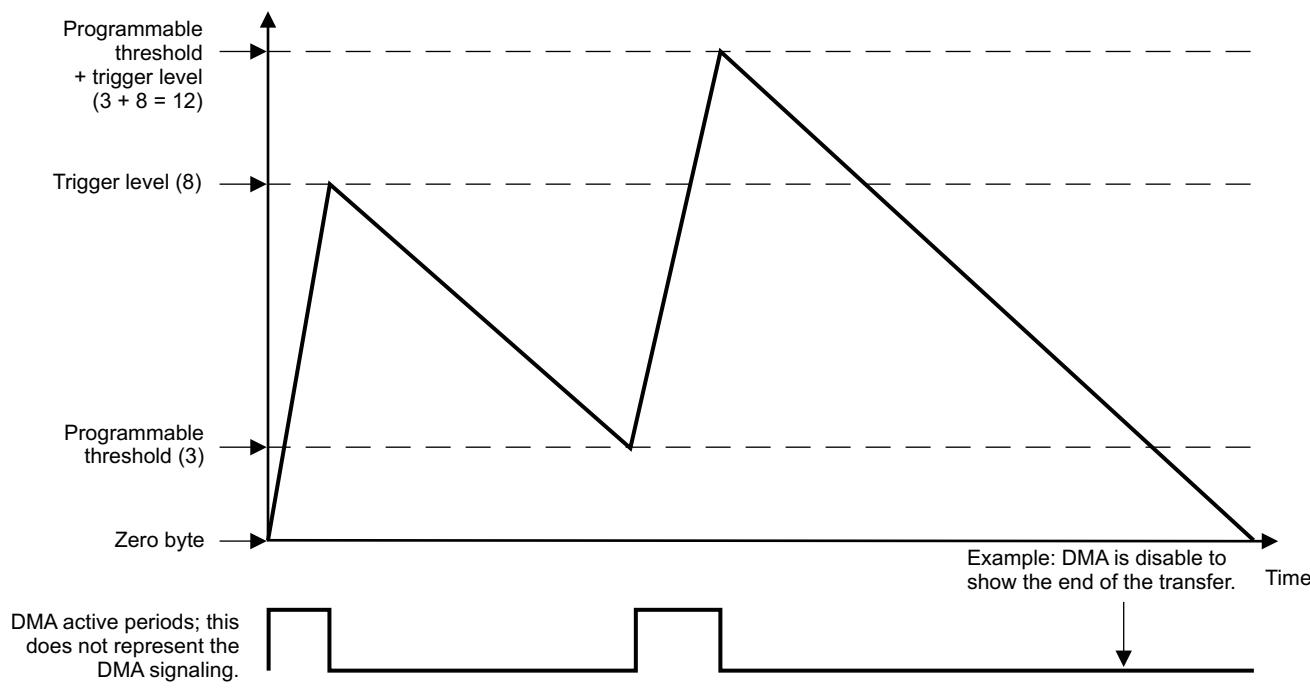
The final example shows the setting of eight spaces, but setting the TX DMA threshold directly by setting the `UART_MDR3[1] NONDEFAULT_FREQ` bit and the `UART_TX_DMA_THRESHOLD` register (see [Figure 19-11](#)). In the example, `UART_TX_DMA_THRESHOLD[2:0]` `TX_DMA_THRESHOLD = 3` and the trigger level is 8. The buffer is filled at a faster rate than the baud rate transmits data to the TX pin. The buffer is filled with 8 bytes and the DMA operations stop transferring data to the transmit buffer. When the buffer is emptied to the threshold level by transmission, the DMA operation activates again to fill the buffer with 8 bytes.

Eventually, the buffer is emptied at the rate specified by the baud rate settings of the `UART_DLL` and `UART_DLH` registers.

If the selected threshold level plus the trigger level exceed the maximum buffer size, the original TX DMA threshold method is used to prevent TX overrun, regardless of the value of the `UART_MDR3[1] NONDEFAULT_FREQ` bit.

The DMA settings must correspond to the settings of the system local host DMA controller to ensure the correct operation of this logic.

**Figure 19-11. Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming.  
(Threshold = 3; Spaces = 8)**

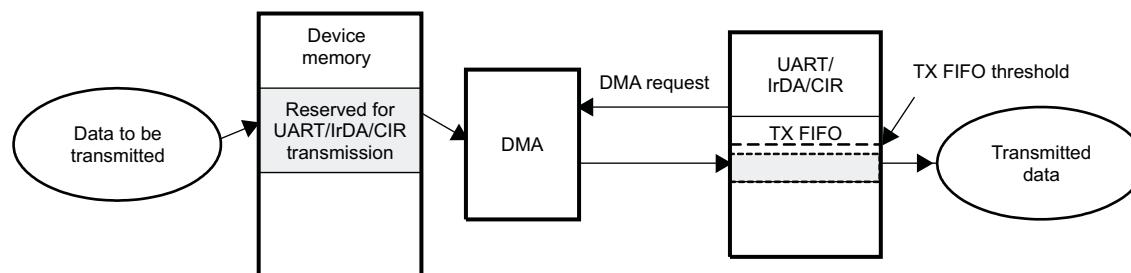


uart-035

#### 19.3.6.4.2 DMA Transmission

[Figure 19-12](#) shows DMA transmission.

**Figure 19-12. DMA Transmission**



uart-030

1. Data to be transmitted are put in the device memory reserved for UART/IrDA/CIR transmission by the DMA:
  - (a) Until the TX FIFO trigger level is not reached, a DMA request is generated
  - (b) An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
2. Data in the TX FIFO are automatically transmitted.
3. The end of the transmission is signaled by the `UARTi.UART_THR` empty (TX FIFO empty).

---

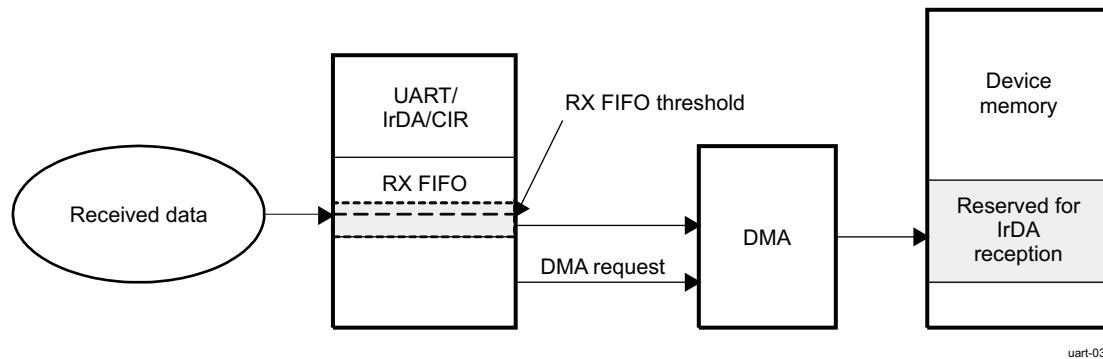
**NOTE:** In IrDA mode, the transmission does not end immediately after the TX FIFO empties, at which point the last data byte, the CRC field, and the stop flag still must be transmitted; thus, the end of transmission occurs a few milliseconds after the `UARTi.UART_THR` register empties.

---

#### 19.3.6.4.3 DMA Reception

Figure 19-13 shows DMA reception.

**Figure 19-13. DMA Reception**



1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA:
  - (a) At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
  - (b) An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

#### 19.3.7 Mode Selection

##### 19.3.7.1 Register Access Modes

###### 19.3.7.1.1 Operational Mode and Configuration Modes

Register access depends on the register access mode, although register access modes are not correlated to functional mode selection. Three different modes are available:

- Operational mode
- Configuration mode A
- Configuration mode B

Operational mode is the selected mode when the function is active; serial data transfer can be performed in this mode.

Configuration mode A and configuration mode B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in the operational mode. The modes are used when the module is inactive (no serial data transfer processed) and only for initialization or reconfiguration of the module.

The value of the `UARTi.UART_LCR` register determines the register access mode (see [Table 19-16](#)).

**Table 19-16. UART/IrDA/CIR Register Access Mode Programming (Using `UART_LCR`)**

| Mode                 | Condition   |
|----------------------|---|
| Configuration mode A | <code>UART_LCR[7] = 0x1</code> and <code>UART_LCR[7:0] != 0xBF</code> |
| Configuration mode B | <code>UART_LCR[7] = 0x1</code> and <code>UART_LCR[7:0] = 0xBF</code>  |
| Operational mode     | <code>UART_LCR[7] = 0x0</code>  |

### 19.3.7.1.2 Register Access Submode

In each access register mode (operational mode or configuration mode A/B), some register accesses are conditional on the programming of a submode (`MSR_SPR`, `TCR_TLR`, and `XOFF`).

[Table 19-17](#) through [Table 19-19](#) summarize the register access submodes.

**Table 19-17. Subconfiguration Mode A Summary**

| Mode                 | Condition  |
|----------------------|--|
| <code>MSR_SPR</code> | ( <code>UART_EFR[4] = 0x0</code> or <code>UART_MCR[6] = 0x0</code> ) |
| <code>TCR_TLR</code> | <code>UART_EFR[4] = 0x1</code> and <code>UART_MCR[6] = 0x1</code>    |

**Table 19-18. Subconfiguration Mode B Summary**

| Mode                 | Condition  |
|----------------------|--|
| <code>TCR_TLR</code> | <code>UART_EFR[4] = 0x1</code> and <code>UART_MCR[6] = 0x1</code>    |
| <code>XOFF</code>    | ( <code>UART_EFR[4] = 0x0</code> or <code>UART_MCR[6] = 0x0</code> ) |

**Table 19-19. Suboperational Mode Summary**

| Mode                 | Condition   |
|----------------------|---|
| <code>MSR_SPR</code> | <code>UART_EFR[4] = 0x0</code> or <code>UART_MCR[6] = 0x0</code>  |
| <code>TCR_TLR</code> | <code>UART_EFR[4] = 0x1</code> and <code>UART_MCR[6] = 0x1</code> |

### 19.3.7.1.3 Registers Available for the Register Access Modes

[Table 19-20](#) lists the names of the register bits in each access register mode. Gray shading indicates that the register does not depend on the register access mode (available in all modes).

**Table 19-20. UART/IrDA/CIR Register Access Mode Overview**

| Address Offset | Registers            |          |                      |                 |                  |              |
|----------------|----------------------|----------|----------------------|-----------------|------------------|--------------|
|                | Configuration Mode A |          | Configuration Mode B |                 | Operational Mode |              |
| Read           | Write                | Read     | Write                | Read            | Write            |              |
| 0x000          | UART_DLL             | UART_DLL | UART_DLL             | UART_DLL        | UART_RHR         | UART_THR     |
| 0x004          | UART_DLH             | UART_DLH | UART_DLH             | UART_DLH        | UART_IER         | UART_IER     |
| 0x008          | UART_IIR             | UART_FCR | UART_EFR             | UART_EFR        | UART_IIR         | UART_FCR     |
| 0x00C          | UART_LCR             | UART_LCR | UART_LCR             | UART_LCR        | UART_LCR         | UART_LCR     |
| 0x010          | UART_MCR             | UART_MCR | UART_XON1_ADD R1     | UART_XON1_AD R1 | UART_MCR DR1     | UART_MCR DR1 |

Table 19-20. UART/IrDA/CIR Register Access Mode Overview (continued)

| Address<br>Offset | Registers  |  |  |  |  |  |
|-------------------|--|--|--|--|--|--|
|                   | Configuration Mode A                                 |  | Configuration Mode B                                   |  | Operational Mode                                     |  |
|                   | Read   | Write  | Read   | Write  | Read   | Write  |
| 0x014             | UART_LSR   | –  | UART_XON2_ADD_R2                                       | UART_XON2_AD   | UART_LSR   | –  |
| 0x018             | UART_MSR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TCR | UART_TCR<br><sup>(2)</sup>                           | UART_TCR<br><sup>(2)</sup> / <sub>(3)</sub> UART_XOFF1 | UART_TCR<br><sup>(2)</sup> / <sub>(3)</sub> UART_XOFF1 | UART_MSR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TCR | UART_TCR<br><sup>(2)</sup>                           |
| 0x01C             | UART_SPR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TLR | UART_SPR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TLR | UART_TLR<br><sup>(2)</sup> / <sub>(3)</sub> UART_XOFF2 | UART_TLR<br><sup>(2)</sup> / <sub>(3)</sub> UART_XOFF2 | UART_SPR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TLR | UART_SPR<br><sup>(1)</sup> / <sub>(2)</sub> UART_TLR |
| 0x020             | UART_MDR1  | UART_MDR1  | UART_MDR1  | UART_MDR1  | UART_MDR1  | UART_MDR1  |
| 0x024             | UART_MDR2  | UART_MDR2  | UART_MDR2  | UART_MDR2  | UART_MDR2  | UART_MDR2  |
| 0x028             | UART_SFLSR   | UART_TXFLL   | UART_SFLSR   | UART_TXFLL   | UART_SFLSR   | UART_TXFLL   |
| 0x02C             | UART_RESUME_E  | UART_TXFLH   | UART_RESUME  | UART_TXFLH   | UART_RESUME  | UART_TXFLH   |
| 0x030             | UART_SFREG_L   | UART_RXFLL   | UART_SFREGL  | UART_RXFLL   | UART_SFREGL  | UART_RXFLL   |
| 0x034             | UART_SFREG_H   | UART_RXFLH   | UART_SFREGH  | UART_RXFLH   | UART_SFREGH  | UART_RXFLH   |
| 0x038             | UART_UASR  | –  | UART_UASR  | –  | UART_BLR   | UART_BLR   |
| 0x03C             | –  | –  | –  | –  | UART_ACREG   | UART_ACREG   |
| 0x040             | UART_SCR   | UART_SCR   | UART_SCR   | UART_SCR   | UART_SCR   | UART_SCR   |
| 0x044             | UART_SSR   | –  | UART_SSR   | –  | UART_SSR   | –  |
| 0x048             | –  | –  | –  | –  | UART_EBLR  | UART_EBLR  |
| 0x050             | UART_MVR   | –  | UART_MVR   | –  | UART_MVR   | –  |
| 0x054             | UART_SYSC  | UART_SYSC  | UART_SYSC  | UART_SYSC  | UART_SYSC  | UART_SYSC  |
| 0x058             | UART_SYSS  | –  | UART_SYSS  | –  | UART_SYSS  | –  |
| 0x05C             | UART_WER   | UART_WER   | UART_WER   | UART_WER   | UART_WER   | UART_WER   |
| 0x060             | UART_CFPS  | UART_CFPS  | UART_CFPS  | UART_CFPS  | UART_CFPS  | UART_CFPS  |
| 0x064             | UART_RXFIFO_LVL                                      | UART_RXFIFO_LVL                                      | UART_RXFIFO_LVL  | UART_RXFIFO_LVL  | UART_RXFIFO_LVL                                      | UART_RXFIFO_LVL                                      |
| 0x068             | UART_TXFIFO_LVL                                      | UART_TXFIFO_LVL                                      | UART_TXFIFO_LVL  | UART_TXFIFO_LVL  | UART_TXFIFO_LVL                                      | UART_TXFIFO_LVL                                      |
| 0x06C             | UART_IER2  | UART_IER2  | UART_IER2  | UART_IER2  | UART_IER2  | UART_IER2  |
| 0x070             | UART_ISR2  | UART_ISR2  | UART_ISR2  | UART_ISR2  | UART_ISR2  | UART_ISR2  |
| 0x074             | UART_FREQ_SEL  | UART_FREQ_S  | UART_FREQ_SEL  | UART_FREQ_SE   | UART_FREQ_SEL  | UART_FREQ_SEL  |
| 0x080             | UART_MDR3  | UART_MDR3  | UART_MDR3  | UART_MDR3  | UART_MDR3  | UART_MDR3  |
| 0x084             | UART_TX_DM_A_THRESHOL_D                              | UART_TX_DMA_THRESHOLD                                | UART_TX_DMA_THRESHOLD                                  | UART_TX_DMA_THRESHOLD                                  | UART_TX_DMA_THRESHOLD                                | UART_TX_DMA_THRESHOLD                                |

<sup>(1)</sup> MSR\_SPR mode is active (see [Section 19.3.7.1.2, Register Access Submode](#))

<sup>(2)</sup> TCR\_TLR mode is active (see [Section 19.3.7.1.2, Register Access Submode](#))

<sup>(3)</sup> XOFF mode is active (see [Section 19.3.7.1.2, Register Access Submode](#))

### 19.3.7.2 UART/IrDA (SIR, MIR, FIR)/CIR Mode Selection

To select a mode, set the `UARTi.UART_MDR1[2:0]` MODESELECT bit field (see [Table 19-21](#)).

**Table 19-21. UART Mode Selection**

| Value | Mode               |
|-------|--------------------|
| 0x0:  | UART 16x mode      |
| 0x1:  | SIR mode           |
| 0x2:  | UART 16x auto-baud |
| 0x3:  | UART 13x mode      |
| 0x4:  | MIR mode           |
| 0x5:  | FIR mode           |
| 0x6:  | CIR mode           |

MODESELECT is effective when the module is in operational mode (see [Section 19.3.7.1, Register Access Modes](#)).

#### 19.3.7.2.1 Registers Available for the UART Function

Only the registers listed in [Table 19-22](#) are used for the UART function.

**Table 19-22. UART Mode Register Overview<sup>(1)</sup> <sup>(2)</sup>**

| Address<br>Offset | Registers             |                       |                         |                         |                            |                        |
|-------------------|-----------------------|-----------------------|-------------------------|-------------------------|----------------------------|------------------------|
|                   | Configuration Mode A  |                       | Configuration Mode B    |                         | Operational Mode           |                        |
|                   | Read                  | Write                 | Read                    | Write                   | Read                       | Write                  |
| 0x000             | UART_DLL              | UART_DLL              | UART_DLL                | UART_DLL                | UART_RHR                   | UART_THR               |
| 0x004             | UART_DLH              | UART_DLH              | UART_DLH                | UART_DLH                | UART_IER(UART)             | UART_IER(UA<br>RT)     |
| 0x008             | UART_IIR              | UART_FCR              | UART_EFR[4]             | UART_EFR[4]             | UART_IIR(UART)             | UART_FCR(U<br>ART)     |
| 0x00C             | UART_LCR              | UART_LCR              | UART_LCR                | UART_LCR                | UART_LCR                   | UART_LCR               |
| 0x010             | UART_MCR              | UART_MCR              | UART_XON1_ADD<br>R1     | UART_XON1_AD            | UART_MCR                   | UART_MCR               |
| 0x014             | UART_LSR(UA<br>RT)    | –                     | UART_XON2_ADD<br>R2     | UART_XON2_AD            | UART_LSR(UART)<br>–<br>DR2 | –                      |
| 0x018             | UART_MSR/U<br>ART_TCR | UART_TCR              | UART_XOFF1/UAR<br>T_TCR | UART_XOFF1/U<br>ART_TCR | UART_MSR/UART<br>_TCR      | UART_TCR               |
| 0x01C             | UART_TLR/UA<br>RT_SPR | UART_TLR/UA<br>RT_SPR | UART_TLR/UART_<br>XOFF2 | UART_TLR/UAR<br>T_XOFF2 | UART_TLR/UART_<br>SPR      | UART_TLR/UA<br>RT_SPR  |
| 0x020             | UART_MDR1             | UART_MDR1[2:<br>0]    | UART_MDR1[2:0]          | UART_MDR1[2:0]          | UART_MDR1[2:0]             | UART_MDR1[2<br>:<br>0] |
| 0x024             | UART_MDR2             | UART_MDR2             | UART_MDR2               | UART_MDR2               | UART_MDR2                  | UART_MDR2              |
| 0x028             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x02C             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x030             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x034             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x038             | UART_UASR             | –                     | UART_UASR               | –                       | –                          | –                      |
| 0x03C             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x040             | UART_SCR              | UART_SCR              | UART_SCR                | UART_SCR                | UART_SCR                   | UART_SCR               |
| 0x044             | UART_SSR              | –                     | UART_SSR                | –                       | UART_SSR                   | –                      |
| 0x048             | –                     | –                     | –                       | –                       | –                          | –                      |
| 0x050             | UART_MVR              | –                     | UART_MVR                | –                       | UART_MVR                   | –                      |
| 0x054             | UART_SYSC             | UART_SYSC             | UART_SYSC               | UART_SYSC               | UART_SYSC                  | UART_SYSC              |
| 0x058             | UART_SYSS             | –                     | UART_SYSS               | –                       | UART_SYSS                  | –                      |

<sup>(1)</sup> REGISTER\_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.

<sup>(2)</sup> REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 19-22. UART Mode Register Overview<sup>(1) (2)</sup> (continued)

| Address<br>Offset | Registers               |                       |                       |                       |                       |                       |
|-------------------|-------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
|                   | Configuration Mode A    |                       | Configuration Mode B  |                       | Operational Mode      |                       |
|                   | Read                    | Write                 | Read                  | Write                 | Read                  | Write                 |
| 0x05C             | UART_WER                | UART_WER              | UART_WER              | UART_WER              | UART_WER              | UART_WER              |
| 0x060             | -                       | -                     | -                     | -                     | -                     | -                     |
| 0x064             | UART_RXFIFO_LVL         | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       |
| 0x068             | UART_TXFIFO_LVL         | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       |
| 0x06C             | UART_IER2               | UART_IER2             | UART_IER2             | UART_IER2             | UART_IER2             | UART_IER2             |
| 0x070             | UART_ISR2               | UART_ISR2             | UART_ISR2             | UART_ISR2             | UART_ISR2             | UART_ISR2             |
| 0x074             | UART_FREQ_SEL           | UART_FREQ_S           | UART_FREQ_SEL         | UART_FREQ_SE          | UART_FREQ_SEL         | UART_FREQ_SEL         |
| 0x080             | UART_MDR3               | UART_MDR3             | UART_MDR3             | UART_MDR3             | UART_MDR3             | UART_MDR3             |
| 0x084             | UART_TX_DM_A_THRESHOL_D | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD |

### 19.3.7.2.2 Registers Available for the IrDA Function

Only the registers listed in Table 19-23 are used for the IrDA function.

Table 19-23. IrDA Mode Register Overview<sup>(1) (2)</sup>

| Address<br>Offset | Registers            |                   |                      |                |                   |                   |
|-------------------|----------------------|-------------------|----------------------|----------------|-------------------|-------------------|
|                   | Configuration Mode A |                   | Configuration Mode B |                | Operational Mode  |                   |
|                   | Read                 | Write             | Read                 | Write          | Read              | Write             |
| 0x000             | UART_DLL             | UART_DLL          | UART_DLL             | UART_DLL       | UART_RHR          | UART_THR          |
| 0x004             | UART_DLH             | UART_DLH          | UART_DLH             | UART_DLH       | UART_IER(IrDA)    | UART_IER(IrDA)    |
| 0x008             | UART_IIR             | UART_FCR          | UART_EFR[4]          | UART_EFR[4]    | UART_IIR(IrDA)    | UART_FCR(IrDA)    |
| 0x00C             | UART_LCR[7]          | UART_LCR[7]       | UART_LCR[7]          | UART_LCR[7]    | UART_LCR[7]       | UART_LCR[7]       |
| 0x010             | -                    | -                 | UART_XON1_ADDR1      | UART_XON1ADDR1 | -                 | -                 |
| 0x014             | UART_LSR(IrDA)       | -                 | UART_XON2_ADDR2      | UART_XON2ADDR2 | UART_LSR(IrDA)    | -                 |
| 0x018             | UART_MSR/UART_TCR    | UART_TCR          | UART_TCR             | UART_TCR       | UART_MSR/UART_TCR | UART_TCR          |
| 0x01C             | UART_TLR/UART_SPR    | UART_TLR/UART_SPR | UART_TLR             | UART_TLR       | UART_TLR/UART_SPR | UART_TLR/UART_SPR |
| 0x020             | UART_MDR1            | UART_MDR1         | UART_MDR1            | UART_MDR1      | UART_MDR1         | UART_MDR1         |
| 0x024             | UART_MDR2            | UART_MDR2         | UART_MDR2            | UART_MDR2      | UART_MDR2         | UART_MDR2         |
| 0x028             | UART_SFSLR           | UART_TXFL         | UART_SFSLR           | UART_TXFL      | UART_SFSLR        | UART_TXFL         |
| 0x02C             | UART_RESUME          | UART_TXFLH_E      | UART_RESUME          | UART_TXFLH     | UART_RESUME       | UART_TXFLH        |
| 0x030             | UART_SFREG_L         | UART_RXFL         | UART_SFREGL          | UART_RXFL      | UART_SFREGL       | UART_RXFL         |
| 0x034             | UART_SFREG_H         | UART_RXFLH        | UART_SFREGH          | UART_RXFLH     | UART_SFREGH       | UART_RXFLH        |

<sup>(1)</sup> REGISTER\_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.

<sup>(2)</sup> REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

**Table 19-23. IrDA Mode Register Overview<sup>(1)</sup> <sup>(2)</sup> (continued)**

| Address<br>Offset | Registers                     |                           |                           |                           |                           |                               |
|-------------------|-------------------------------|---------------------------|---------------------------|---------------------------|---------------------------|-------------------------------|
|                   | Configuration Mode A          |                           | Configuration Mode B      |                           | Operational Mode          |                               |
|                   | Read                          | Write                     | Read                      | Write                     | Read                      | Write                         |
| 0x038             | —                             | —                         | —                         | —                         | UART_BLR                  | UART_BLR                      |
| 0x03C             | —                             | —                         | —                         | —                         | UART_ACREG                | UART_ACREG                    |
| 0x040             | UART_SCR                      | UART_SCR                  | UART_SCR                  | UART_SCR                  | UART_SCR                  | UART_SCR                      |
| 0x044             | UART_SSR                      | —                         | UART_SSR                  | —                         | UART_SSR                  | —                             |
| 0x048             | —                             | —                         | —                         | —                         | UART_EBLR                 | UART_EBLR                     |
| 0x050             | UART_MVR                      | —                         | UART_MVR                  | —                         | UART_MVR                  | —                             |
| 0x054             | UART_SYSC                     | UART_SYSC                 | UART_SYSC                 | UART_SYSC                 | UART_SYSC                 | UART_SYSC                     |
| 0x058             | UART_SYSS                     | —                         | UART_SYSS                 | —                         | UART_SYSS                 | —                             |
| 0x05C             | UART_WER[6:<br>4]             | UART_WER[6:4]             | UART_WER[6:4]             | UART_WER[6:4]             | UART_WER[6:4]             | UART_WER[6:<br>4]             |
| 0x060             | —                             | —                         | —                         | —                         | —                         | —                             |
| 0x064             | UART_RXFIFO_<br>_LVL          | UART_RXFIFO_<br>_LVL      | UART_RXFIFO_LVL           | UART_RXFIFO_L_<br>VL      | UART_RXFIFO_LV<br>_LVL    | UART_RXFIFO_<br>_LVL          |
| 0x068             | UART_TXFIFO_<br>_LVL          | UART_TXFIFO_<br>_LVL      | UART_TXFIFO_LVL           | UART_TXFIFO_L_<br>VL      | UART_TXFIFO_LV<br>_LVL    | UART_TXFIFO_<br>_LVL          |
| 0x06C             | UART_IER2                     | UART_IER2                 | UART_IER2                 | UART_IER2                 | UART_IER2                 | UART_IER2                     |
| 0x070             | UART_ISR2                     | UART_ISR2                 | UART_ISR2                 | UART_ISR2                 | UART_ISR2                 | UART_ISR2                     |
| 0x074             | UART_FREQ_<br>SEL             | UART_FREQ_S               | UART_FREQ_SEL             | UART_FREQ_SE<br>L         | UART_FREQ_SEL             | UART_FREQ_<br>SEL             |
| 0x080             | UART_MDR3                     | UART_MDR3                 | UART_MDR3                 | UART_MDR3                 | UART_MDR3                 | UART_MDR3                     |
| 0x084             | UART_TX_DM<br>A_THRESHOL<br>D | UART_TX_DMA<br>_THRESHOLD | UART_TX_DMA_TH<br>RESHOLD | UART_TX_DMA_<br>THRESHOLD | UART_TX_DMA_T<br>HRESHOLD | UART_TX_DM<br>A_THRESHOL<br>D |

### 19.3.7.2.3 Registers Available for the CIR Function

Only the registers listed in [Table 19-24](#) are used for the CIR function.

**Table 19-24. CIR Mode Register Overview<sup>(1)</sup> <sup>(2)</sup>**

| Address<br>Offset | Registers             |                       |                      |             |                       |                       |
|-------------------|-----------------------|-----------------------|----------------------|-------------|-----------------------|-----------------------|
|                   | Configuration Mode A  |                       | Configuration Mode B |             | Operational Mode      |                       |
|                   | Read                  | Write                 | Read                 | Write       | Read                  | Write                 |
| 0x000             | UART_DLL              | UART_DLL              | UART_DLL             | UART_DLL    | —                     | UART_THR              |
| 0x004             | UART_DLH              | UART_DLH              | UART_DLH             | UART_DLH    | UART_IER(CIR)         | UART_IER(CL<br>R)     |
| 0x008             | UART_IIR              | UART_FCR              | UART_EFR             | UART_EFR    | UART_IIR(CIR)         | UART_FCR(CL<br>R)     |
| 0x00C             | UART_LCR              | UART_LCR[7]           | UART_LCR[7]          | UART_LCR[7] | UART_LCR[7]           | UART_LCR[7]           |
| 0x010             | —                     | —                     | —                    | —           | —                     | —                     |
| 0x014             | UART_LSR(IrD<br>A)    | —                     | —                    | —           | UART_LSR(IrDA)        | —                     |
| 0x018             | UART_MSR/U<br>ART_TCR | UART_TCR              | UART_TCR             | UART_TCR    | UART_MSR/UART<br>_TCR | UART_TCR              |
| 0x01C             | UART_TLR/UA<br>RT_SPR | UART_TLR/UA<br>RT_SPR | UART_TLR             | UART_TLR    | UART_TLR/UART_<br>SPR | UART_TLR/UA<br>RT_SPR |

<sup>(1)</sup> REGISTER\_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.

<sup>(2)</sup> REGISTER\_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 19-24. CIR Mode Register Overview<sup>(1) (2)</sup> (continued)

| Address<br>Offset | Registers              |                       |                       |                       |                       |                       |
|-------------------|------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
|                   | Configuration Mode A   |                       | Configuration Mode B  |                       | Operational Mode      |                       |
|                   | Read                   | Write                 | Read                  | Write                 | Read                  | Write                 |
| 0x020             | UART_MDR1[3:0]         | UART_MDR1[3:0]        | UART_MDR1[3:0]        | UART_MDR1[3:0]        | UART_MDR1[3:0]        | UART_MDR1[3:0]        |
| 0x024             | UART_MDR2              | UART_MDR2             | UART_MDR2             | UART_MDR2             | UART_MDR2             | UART_MDR2             |
| 0x028             | —                      | —                     | —                     | —                     | —                     | —                     |
| 0x02C             | UART_RESUME            | —                     | UART_RESUME           | —                     | UART_RESUME           | —                     |
| 0x030             | —                      | —                     | —                     | —                     | —                     | —                     |
| 0x034             | —                      | —                     | —                     | —                     | —                     | —                     |
| 0x038             | —                      | —                     | —                     | —                     | —                     | —                     |
| 0x03C             | —                      | —                     | —                     | —                     | UART_ACREG            | UART_ACREG            |
| 0x040             | UART_SCR               | UART_SCR              | UART_SCR              | UART_SCR              | UART_SCR              | UART_SCR              |
| 0x044             | UART_SSR               | —                     | UART_SSR              | —                     | UART_SSR              | —                     |
| 0x048             | —                      | —                     | —                     | —                     | UART_EBLR             | UART_EBLR             |
| 0x050             | UART_MVR               | —                     | UART_MVR              | —                     | UART_MVR              | —                     |
| 0x054             | UART_SYSC              | UART_SYSC             | UART_SYSC             | UART_SYSC             | UART_SYSC             | UART_SYSC             |
| 0x058             | UART_SYSS              | —                     | UART_SYSS             | —                     | UART_SYSS             | —                     |
| 0x05C             | UART_WER[6:4]          | UART_WER[6:4]         | UART_WER[6:4]         | UART_WER[6:4]         | UART_WER[6:4]         | UART_WER[6:4]         |
| 0x060             | UART_CFPS              | UART_CFPS             | UART_CFPS             | UART_CFPS             | UART_CFPS             | UART_CFPS             |
| 0x064             | UART_RXFIFO_LVL        | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       | UART_RXFIFO_LVL       |
| 0x068             | UART_TXFIFO_LVL        | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       | UART_TXFIFO_LVL       |
| 0x06C             | UART_IER2              | UART_IER2             | UART_IER2             | UART_IER2             | UART_IER2             | UART_IER2             |
| 0x070             | UART_ISR2              | UART_ISR2             | UART_ISR2             | UART_ISR2             | UART_ISR2             | UART_ISR2             |
| 0x074             | UART_FREQ_SEL          | UART_FREQ_S           | UART_FREQ_SEL         | UART_FREQ_SE          | UART_FREQ_SEL         | UART_FREQ_SEL         |
| 0x080             | UART_MDR3              | UART_MDR3             | UART_MDR3             | UART_MDR3             | UART_MDR3             | UART_MDR3             |
| 0x084             | UART_TX_DM_A_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD | UART_TX_DMA_THRESHOLD |

### 19.3.8 Protocol Formatting

The UART/IRDA module can operate in seven different modes:

1. UART 16x mode ( $\leq 230.4$  Kbits/s), UART16x  $\leq 460$ Kbits/s if MDR3[1] is set
2. UART 16x mode with autobauding ( $\geq 1200$  bits/s and  $\leq 115.2$  Kbits/s) if MDR3[1] is not set
3. UART 13x mode ( $\geq 460.8$  Kbits/s) if MDR3[1] is not set
4. IrDA SIR mode ( $\leq 115.2$  Kbits/s) if MDR3[1] is not set
5. IrDA MIR mode (0.576 and 1.152 Mbits/s) if MDR3[1] is not set
6. IrDA FIR mode (4 Mbits/s) if MDR3[1] is not set
7. CIR mode (programmable modulation rates specific to remote control applications) if MDR3[1] is not set

The module performs a serial-to-parallel conversion on received data characters and a parallel-to-serial conversion on transmitted data characters by the processor. The complete status of each channel of the module and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The module can be placed in an alternate mode (FIFO mode) to relieve the processor of excessive software overhead by buffering received/transmitted characters.

Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels. Both interrupts and DMA are available to control the data flow between the LH and the module.

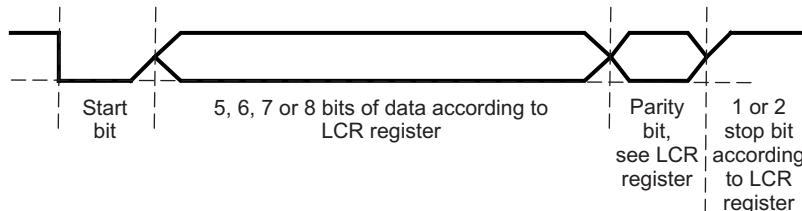
#### 19.3.8.1 UART Mode

The UART uses a wired interface for serial communication with a remote device.

The UART module is functionally compatible with the TL16C750 UART and is also functionally compatible to earlier designs, such as the TL16C550. The UART module can use hardware or software flow control to manage transmission and reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

The UART modem module is enhanced with an autobauding functionality which in control mode allows to automatically set the speed, the number of bit per character, the parity selected.

**Figure 19-14. UART Data Format**

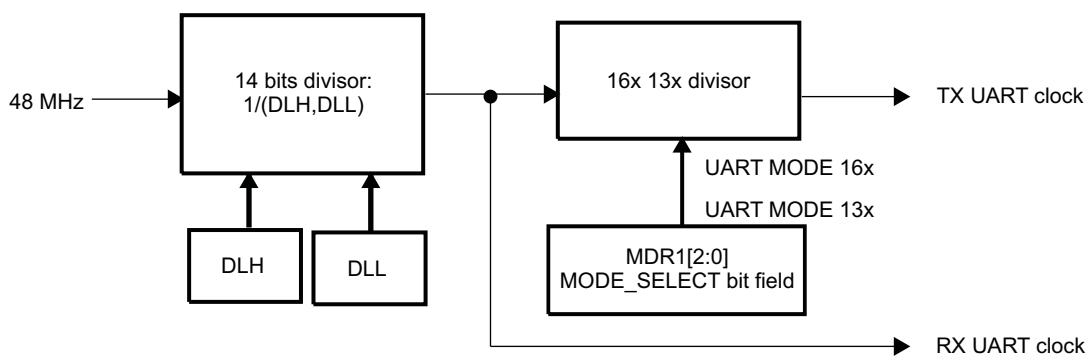


##### 19.3.8.1.1 UART Clock Generation: Baud Rate Generation

The UART function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

[Figure 19-15](#) shows the baud rate generator and associated controls.

**Figure 19-15. Baud Rate Generation**



#### CAUTION

Before initializing or modifying clock parameter controls (UARTi.UART\_DLH, UARTi.UART\_DLL), MODE\_SELECT = DISABLE (UARTi.UART\_MDR1[2:0]) must be set to 0x7. Failure to observe this rule can result in unpredictable module behavior.

### 19.3.8.1.2 Choosing the Appropriate Divisor Value

Two divisor values are:

- UART 16x mode: Divisor value = Operating frequency/(16x baud rate)
- UART 13x mode: Divisor value = Operating frequency/(13x baud rate)

[Table 19-25](#) describes the UART baud rate settings.

**Table 19-25. UART Baud Rate Settings (48-MHz Clock)**

| Baud Rate   | Baud Multiple | DLH,DLL (Decimal) | DLH,DLL (Hex) | Actual Baud Rate | Error (%) |
|-------------|---------------|-------------------|---------------|------------------|-----------|
| 0.3 kbps    | 16x           | 10000             | 0x27, 0x10    | 0.3 kbps         | 0         |
| 0.6 kbps    | 16x           | 5000              | 0x13, 0x88    | 0.6 kbps         | 0         |
| 1.2 kbps    | 16x           | 2500              | 0x09, 0xC4    | 1.2 kbps         | 0         |
| 2.4 kbps    | 16x           | 1250              | 0x04, 0xE2    | 2.4 kbps         | 0         |
| 4.8 kbps    | 16x           | 625               | 0x02, 0x71    | 4.8 kbps         | 0         |
| 9.6 kbps    | 16x           | 312               | 0x01, 0x38    | 9.6153 kbps      | +0.16     |
| 14.4 kbps   | 16x           | 208               | 0x00, 0xD0    | 14.423 kbps      | +0.16     |
| 19.2 kbps   | 16x           | 156               | 0x00, 0x9C    | 19.231 kbps      | +0.16     |
| 28.8 kbps   | 16x           | 104               | 0x00, 0x68    | 28.846 kbps      | +0.16     |
| 38.4 kbps   | 16x           | 78                | 0x00, 0x4E    | 38.462 kbps      | +0.16     |
| 57.6 kbps   | 16x           | 52                | 0x00, 0x34    | 57.692 kbps      | +0.16     |
| 115.2 kbps  | 16x           | 26                | 0x00, 0x1A    | 115.38 kbps      | +0.16     |
| 230.4 kbps  | 16x           | 13                | 0x00, 0x0D    | 230.77 kbps      | +0.16     |
| 460.8 kbps  | 13x           | 8                 | 0x00, 0x08    | 461.54 kbps      | +0.16     |
| 921.6 kbps  | 13x           | 4                 | 0x00, 0x04    | 923.08 kbps      | +0.16     |
| 1.843 Mbps  | 13x           | 2                 | 0x00, 0x02    | 1.846 Mbps       | +0.16     |
| 3.6884 Mbps | 13x           | 1                 | 0x00, 0x01    | 3.6923 Mbps      | +0.16     |

### 19.3.8.1.3 UART Data Formatting

The UART can use hardware flow control to manage transmission and reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals.

The UART is enhanced with the autobauding function. In control mode, autobauding lets the speed, the number of bits per character, and the parity selected be set automatically.

#### 19.3.8.1.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the `UARTi.UART_LCR` register.

Character length is specified using the `UARTi.UART_LCR[1:0] CHAR_LENGTH` bit field.

The number of stop-bits is specified using the `UARTi.UART_LCR[2] NB_STOP` bit.

The parity bit is programmed using the `UARTi.UART_LCR[5:3]` `PARITY_EN`, `PARITY_TYPE_1`, and `PARITY_TYPE_2` bit fields (see [Table 19-26](#)).

**Table 19-26. UART Parity Bit Encoding**

| PARITY_EN | PARITY_TYPE_1 | PARITY_TYPE_2 | Parity      |
|-----------|---------------|---------------|-------------|
| 0         | N/A           | N/A           | No parity   |
| 1         | 0             | 0             | Odd parity  |
| 1         | 1             | 0             | Even parity |
| 1         | 0             | 1             | Forced 1    |
| 1         | 1             | 1             | Forced 0    |

### 19.3.8.1.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled and disabled independently by programming the `UARTi.UART_EFR[7:6] AUTO_CTS_EN` and `AUTO_RTS_EN` bit fields, respectively.

With auto-CTS, `uarti_cts` must be active before the module can transmit data.

Auto-RTS activates the `uarti_rts` output only when there is enough room in the RX FIFO to receive data. It deactivates the `uarti_rts` output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the `UARTi.UART_TCR` register determine the levels at which `uarti_rts` is activated and deactivated.

If auto-CTS and auto-RTS are enabled, data transmission does not occur unless the RX FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If auto-CTS and auto-RTS are not enabled, overrun errors occur if the transmit data rate exceeds the RX FIFO latency.

- Auto-RTS:

Auto-RTS data flow control originates in the receiver block. The RX FIFO trigger levels used in auto-RTS are stored in the `UARTi.UART_TCR` register. `uarti_rts` is active if the RX FIFO level is below the HALT trigger level in the `UARTi.UART_TCR[3:0] RX_FIFO_TRIG_HALTED` bit field. When the RX FIFO HALT trigger level is reached, `uarti_rts` is deasserted. The sending device (for example, another UART) can send an additional byte after the trigger level is reached because it may not recognize the deassertion of RTS until it begins sending the additional byte.

`uarti_rts` is automatically reasserted when the RX FIFO reaches the RESUME trigger level programmed by the `UARTi.UART_TCR[7:4] RX_FIFO_TRIG_START` bit field. This reassertion requests the sending device to resume transmission.

In this case, `uarti_rts` is an active-low signal.

- Auto-CTS:

The transmitter circuitry checks `uarti_cts` before sending the next data byte. When `uarti_cts` is active, the transmitter sends the next byte. To stop the transmitter from sending the next byte, `uarti_cts` must be deasserted before the middle of the last stop-bit currently sent.

The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the `uarti_cts` state changes do not have to trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can result.

In this case, `uarti_cts` is an active-low signal.

### 19.3.8.1.3.3 Software Flow Control

Software flow control is enabled through the enhanced feature register (`UARTi.UART_EFR`) and the modem control register (`UARTi.UART_MCR`). Different combinations of software flow control can be enabled by setting different combinations of the `UARTi.UART_EFR[3:0]` bit field (see [Table 19-27](#)).

Two other enhanced features relate to software flow control:

- XON any function (`UARTi.UART_MCR[5]`): Operation resumes after receiving any character after the XOFF character is recognized. If special character detect is enabled and special character is received after XOFF1, it does not resume transmission. The special character is stored in the RX FIFO.

---

**NOTE:** The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (`UARTi.UART_EFR[5]`): Incoming data is compared to XOFF2. When the special character is detected, the XOFF interrupt (`UARTi.UART_IIR[4]`) is set, but it does not halt transmission. The XOFF interrupt is cleared by a read of `UARTi.UART_IIR`. The special character is transferred to the RX FIFO. Special character does not work with XON2, XOFF2, or sequential XOFFs.

**Table 19-27. UART\_EFR[3:0] Software Flow Control Options**

| Bit 3 | Bit 2 | Bit 1 | Bit 0 | TX, RX Software Flow Controls                             |
|-------|-------|-------|-------|---|
| 0     | 0     | X     | X     | No transmit flow control                                  |
| 1     | 0     | X     | X     | Transmit XON1, XOFF1                                      |
| 0     | 1     | X     | X     | Transmit XON2, XOFF2                                      |
| 1     | 1     | X     | X     | Transmit XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup>          |
| X     | X     | 0     | 0     | No receive flow control                                   |
| X     | X     | 1     | 0     | Receiver compares XON1, XOFF1                             |
| X     | X     | 0     | 1     | Receiver compares XON2, XOFF2                             |
| X     | X     | 1     | 1     | Receiver compares XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup> |

<sup>(1)</sup> In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.  
 XON1 is defined in the UARTi.UART\_XON1\_ADDR1[7:0] XON\_WORD1 bit field. XON2 is defined in the UARTi.UART\_XON2\_ADDR2[7:0] XON\_WORD2 bit field.  
 XOFF1 is defined in the UARTi.UART\_XOFF1[7:0] XOFF\_WORD1 bit field. XOFF2 is defined in the UARTi.UART\_XOFF2[7:0] XOFF\_WORD2 bit field.

#### 19.3.8.1.3.3.1 Receive (RX)

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases, XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission stops after transmission of the current character completes. Detection of XOFF also sets the UARTi.UART\_IIR[4] bit (if enabled by UARTi.UART\_IER[5]) and causes the interrupt line to go low.

To resume transmission, an XON1/2 character must be received (in certain cases, XON1 and XON2 must be received sequentially). When the correct XON characters are received, the UARTi.UART\_IIR[4] bit is cleared and the XOFF interrupt disappears.

---

**NOTE:** When a parity, framing, or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

---

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when UARTi.UART\_EFR[1:0] = 0x2, if XON1 and XOFF1 characters are received, they are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (UARTi.UART\_EFR[1:0] = 0x3), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

#### 19.3.8.1.3.3.2 Transmit (TX)

Two XOFF1 characters are transmitted when the RX FIFO passes the trigger level programmed by UARTi.UART\_TCR[3:0]. As soon as the RX FIFO reaches the trigger level programmed by UARTi.UART\_TCR[7:4], two XON1 characters are sent, so the data transfer recovers.

---

**NOTE:** If software flow control is disabled after an XOFF character is sent, the module transmits XON characters automatically to enable normal transmission.

---

The transmission of XOFF(s)/XON(s) follows the same protocol as transmission of an ordinary byte from the TX FIFO. This means that even if the word length is 5, 6, or 7 characters, the 5, 6, or 7 LSBs of XOFF1/2 and XON1/2 are transmitted. The 5, 6, or 7 bits of a character are seldom transmitted, but this function is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

### 19.3.8.1.3.4 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an "at" (AT) command (ASCII code). These characteristics are used to receive data after an AT and to send data.

The following AT commands are valid:

|    |      |      |
|----|------|------|
| AT | DATA | <CR> |
| at | DATA | <CR> |
| A/ |      |      |
| a/ |      |      |

A line break during the acquisition of the sequence AT is not recognized, and an echo function is not implemented in hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. A/ or a/ is used to instruct the software to repeat the last received AT command; therefore, an a/ always follows an AT, and transfer characteristics are not expected to change between an AT and an a/.

When a valid AT is received, AT and all subsequent data, including the final <CR> (0x0D), are saved to the RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved in the RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The `UARTi.UART_UASR` register reflects the correct settings for the baud rate detected. Interrupt activity can continue in this fashion when a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- Speed:
  - 115.2K baud
  - 57.6K baud
  - 38.4K baud
  - 28.8K baud
  - 19.2K baud
  - 14.4K baud
  - 9.6K baud
  - 4.8K baud
  - 2.4K baud
  - 1.2K baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

---

**NOTE:** The combination of 7-bit character plus space parity is not supported.

---

Autobauding mode is selected when the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field is set to 0x2. In UART autobauding mode, `UARTi.UART_DLL`, `UARTi.UART_DLH`, and `UARTi.UART_LCR[5:0]` bit field settings are not used; instead, `UART_UASR` is updated with the configuration detected by the autobauding logic.

#### UART\_UASR Autobauding Status Register Use

This register is used to set up transmission according to the characteristics of the previous reception instead of the `UARTi.UART_LCR`, `UARTi.UART_DLL`, and `UARTi.UART_DLH` registers when the UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), the **UART<sub>i</sub>.UART\_MDR[2:0] MODE\_SELECT** bit field must be set to reset state (0x7) and then to the UART in autobauding mode (0x2) or to the UART in standard mode (0x0).

Use limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115,200 bps (10 possibilities)

#### **19.3.8.1.3.5 Error Detection**

When the **UART<sub>i</sub>.UART\_LSR** register is read, the **UART<sub>i</sub>.UART\_LSR[4:2]** bit field reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (the next character to be read). Therefore, reading the **UART<sub>i</sub>.UART\_LSR** register and then reading the **UART<sub>i</sub>.UART\_RHR** register identifies errors in a character.

Reading the **UART<sub>i</sub>.UART\_RHR** register updates the BI, FE, and PE bits (see [Table 19-11](#) for the UART mode interrupts).

The **UART<sub>i</sub>.UART\_LSR[7] RX\_FIFO\_STS** bit is set when there is an error in the RX FIFO and is cleared only when no errors remain in the RX FIFO.

---

**NOTE:** Reading the **UART<sub>i</sub>.UART\_LSR** register does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the **UART<sub>i</sub>.UART\_RHR** register.

---

Reading the **UART<sub>i</sub>.UART\_LSR** register clears the OE bit if it is set (see [Table 19-11](#) for the UART mode interrupts).

#### **19.3.8.1.3.6 Overrun During Receive**

Overrun during receive occurs if the RX state-machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the MPU with the **UART<sub>i</sub>.UART\_IIR[5:1] IT\_TYPE** bit field set to 0x3 (receiver line status error) and discards the remaining portion of the frame.

Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received, the MPU must:

- Reset the RX FIFO.
- Read the **UART<sub>i</sub>.UART\_RESUME** register, which clears the internal flag.

#### **19.3.8.1.3.7 Time-Out and Break Conditions**

##### **19.3.8.1.3.7.1 Time-Out Counter**

An RX idle condition is detected when the receiver line (**uarti\_rx**) is high for a time that equals 4x the programmed word length + 12 bits. **uarti\_rx** is sampled midway through each bit.

For sleep mode, the counter is reset when there is activity on **uarti\_rx**.

For the time-out interrupt, the counter counts only when there is data in the RX FIFO, and the count is reset when there is activity on **uarti\_rx** or when the **UART<sub>i</sub>.UART\_RHR** register is read.

##### **19.3.8.1.3.7.2 Break Condition**

When a break condition occurs, **uarti\_tx** is pulled low. A break condition is activated by setting the **UART<sub>i</sub>.UART\_LCR[6] BREAK\_EN** bit. The break condition is not aligned on word stream (a break condition can occur in the middle of a character). The only way to send a break condition on a full character is:

1. Reset the TX FIFO (if enabled).
2. Wait for the transmit shift register to empty (the **UART<sub>i</sub>.UART\_LSR[6] TX\_SR\_E** bit is set to 1).

3. Take a guard time according to stop-bit definition.
4. Set the BREAK\_EN bit to 1.

The break condition is asserted while the BREAK\_EN bit is set to 1.

The time-out counter and break condition apply only to UART modem operation and not to IrDA/CIR mode operation.

### 19.3.8.2 IrDA Mode

#### 19.3.8.2.1 SIR Mode

In slow infrared (SIR) mode, data transfers take place between the LH and peripheral devices at speeds of up to 115200 bauds. A SIR transmit frame starts with start flags (either a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data, CRC-16, and ends with a stop flag (0xC1). The bit format for a single word uses a single start bit, eight data bits, and one stop bit. The format is unaffected by the use and settings of the LCR register.

Note that BLR[6] is used to select whether to use 0xC0 or 0xFF start patterns when multiple start flags are required.

The SIR transmit state machine attaches start flags, CRC-16, and stop flags. It checks the outgoing data to establish if data transparency is required.

SIR transparency is carried out if the outgoing data, between the start and stop flags, contains 0xC0, 0xC1, or 0x7D. If one of these is about to be transmitted, then the SIR state machine sends an escape character (0x7D) first, then inverts the fifth bit of the real data to be sent and sends this data immediately after the 0x7D character.

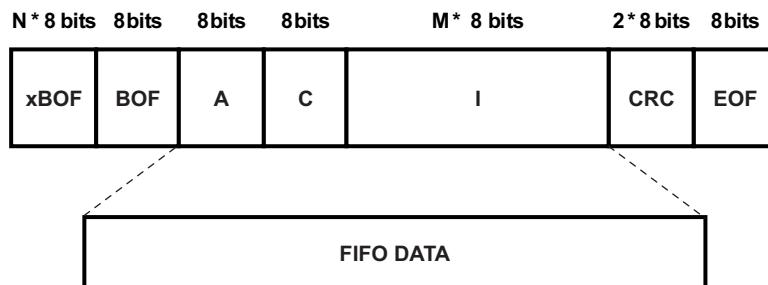
The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors, such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

Data can be transferred both ways by the module, but when the device is transmitting the IR RX circuitry is automatically disabled by hardware. See bit 5 in [Section 19.5.1.26, Auxiliary Control Register](#), for a description of the logical operation. **Note:** This applies to all three modes SIR, MIR, and FIR.

The infrared output in SIR mode can either be 1.6μs or 3/16 encoding, selected by the PULSETYPE bit of the Auxiliary Control Register (ACREG[7]). In 1.6μs encoding, the infrared pulse width is 1.6μs and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baud-rate). The receiver supports both 3/16 and 1.6μs pulse duration by default. The transmitting device must send at least two start flags at the start of each frame for back-to-back frames. **Note:** Reception supports variable-length stop bits.

#### 19.3.8.2.1.1 Frame Format

**Figure 19-16. IrDA SIR Frame Format**



The CRC is applied on the address (A), control (C) and information (I) bytes.

**Note:** The two words of CRC are written in the FIFO in reception.

### 19.3.8.2.1.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte of the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), or 0x7D (control escape) it does the following.

#### In transmission

1. Inserts a control escape (CE) byte preceding the byte.
2. Complements bit 5 of the byte (i.e., exclusive OR's the byte with 0x20).

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

#### In reception

For the A, C, I, CRC field:

1. Compare the byte with CE byte, and if not equal send it to the CRC detector and store it in the RX FIFO.
2. If equal to CE, discard the CE byte.
3. Complements the bit 5 of the byte following the CE.
4. Send the complemented byte to the CRC detector and store it in the RX FIFO.

### 19.3.8.2.1.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the following sequence: 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming the ABORTEN bit of the Auxiliary Control Register (ACREG[1]). When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags. The receiver treats a frame as an aborted frame when a 0x7D character, followed immediately by a 0xC1 character, has been received without transparency.

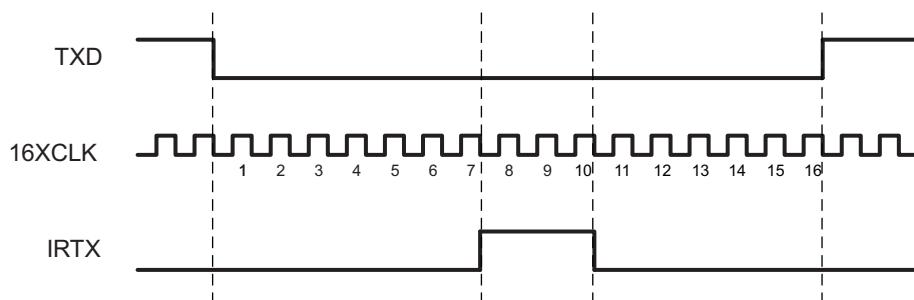
### 19.3.8.2.1.4 Pulse Shaping

In SIR mode, both the 3/16th and the 1.6 $\mu$ s pulse duration methods are supported in receive and transmit. ACREG[7] selects the pulse width method in transmit mode.

### 19.3.8.2.1.5 Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics. While the serial data input to the (TXD) is high, the output (IRTX) is always low, and the counter used to form a pulse on IRTX is continuously cleared. After TXD resets to 0, IRTX rises on the falling edge of the 7th 16XCLK. On the falling edge of the 10th 16XCLK pulse, IRTX falls, creating a 3-clock-wide pulse. While TXD stays low, a pulse is transmitted during the 7th to the 10th clock of each 16-clock bit cycle.

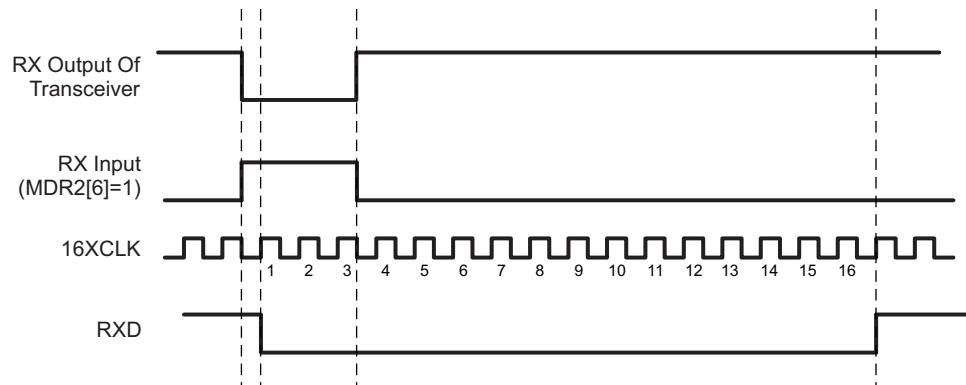
**Figure 19-17. IrDA Encoding Mechanism**



### 19.3.8.2.1.6 Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RX, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD stays low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RX, RXD remains high.

**Figure 19-18. IrDA Decoding Mechanism**



The operation of the RX input can be disabled with DISIRRX bit of the Auxiliary Control Register (ACREG[5]). Furthermore, the MDR2[6] can be used to invert the signal from the transceiver (RX output) pin to the IRRX logic inside the UART.

### 19.3.8.2.1.7 IR Address Checking

In all IR modes, if address checking has been enabled, only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with XON1/ADDR1 and XON2/ADDR2 registers. Selecting address1 checking is done by setting EFR[0] to 1; address2 checking is done by setting EFR[1] to 1.

Setting EFR[1:0] to 0 disables all address checking operations. If both bits are set, then the incoming frame is checked for both private and public addresses. If address checking is disabled, then all received frames are written into the reception FIFO.

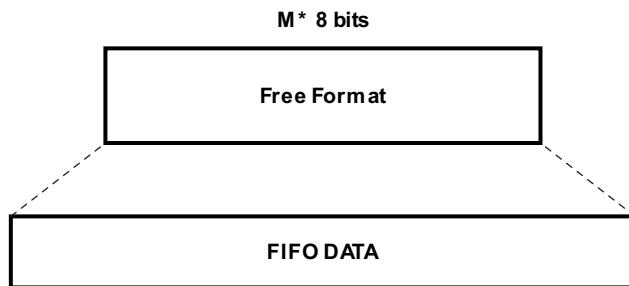
### 19.3.8.2.1.8 SIR Free Format Mode

To allow complete software flexibility in the transmission and reception of Infrared data packets, the SIR free format mode is a sub-function of the existing SIR mode such that all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values. In the transmission phase, the mode uses the IRTX pin, as in SIR mode.

This mode corresponds to a UART mode with a pulse modulation of 3/16 of baud-rate pulse width.

For example, a normal SIR packet has BOF control and CRC error checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs. In SIR free format mode, only the data termed the FIFO DATA area, illustrated in [Figure 19-19](#), would be transmitted and received.

**Figure 19-19. SIR Free Format Mode**



In this mode, the entire FIFO data packet is to be constructed (encoded and decoded) by the LH software.

The SIR free format mode is selected by setting the module in UART mode ( $MDR1[2:0] = 000$ ) and the  $MDR2[3]$  register bit to one to allow the pulse shaping. As the bit format is to remain the same, some UART mode configuration registers need to be set at specific value:

- $LCR[1:0] = "11"$  (8 data bits)
- $LCR[2] = 0$  (1 stop bit)
- $LCR[3] = 0$  (no parity)
- $ACREG[7] = 0$  (3/16 of baud-rate pulse width)

The features defined through  $MDR2[6]$  and  $ACREG[5]$  are also supported.

**Note:** - All other configuration registers need to be at the reset value. The UART mode interrupts are used for the SIR free format mode, but many of them are not relevant (e.g., XOFF, RTS, CTS, Modem status register).

#### 19.3.8.2.2 MIR Mode

In medium infrared (MIR) mode, data transfers take place between LH and peripheral devices at 0.576 or 1.152 Mbits/s speed. A MIR transmit frame starts with start flags (at least two), followed by a frame data, CRC-16, and ends with a stop flag.

**Figure 19-20. MIR Transmit Frame Format**



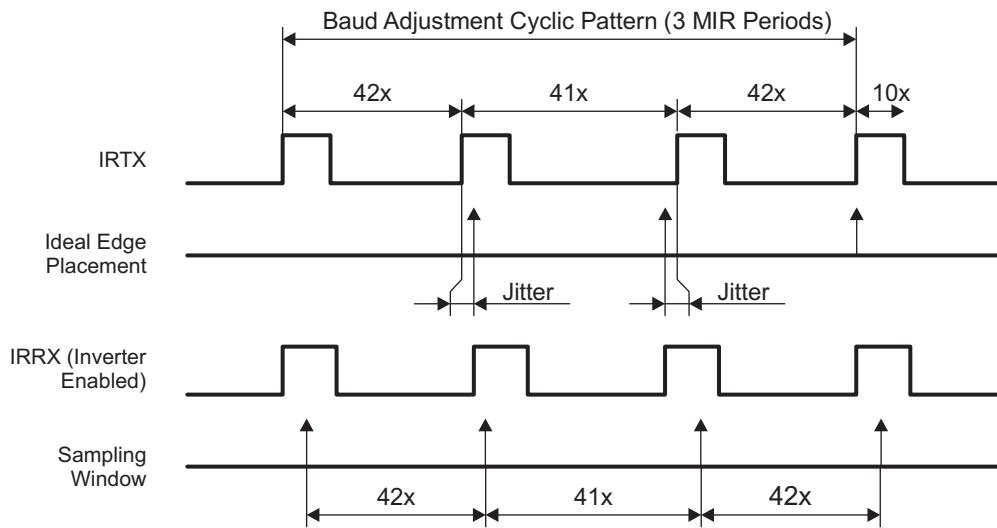
On transmit, the MIR state machine attaches start flags, CRC-16, and stop flags. It also looks for five consecutive values of 1 in the frame data and automatically inserts a zero after five consecutive values of one (this is called bit stuffing).

On receive, the MIR receive state machine recovers the receive clock, removes the start flags, de-stuffs the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors, such as frame abort, CRC error, or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find possible errors of received frame.

Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See bit 5 in [Section 19.5.1.26, Auxiliary Control Register](#), for a description of the logical operation. **Note:** This applies to all three modes SIR, MIR and FIR.

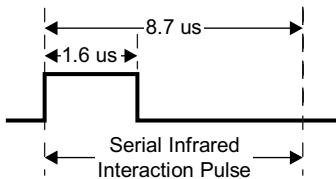
##### 19.3.8.2.2.1 MIR Encoder/Decoder

In order to meet MIR baud-rate tolerance of +/-0.1% with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected. The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4 but within tolerances defined by the IrDA specifications.

**Figure 19-21. MIR BAUD Rate Adjustment Mechanism**


#### 19.3.8.2.2.2 SIP Generation

In MIR and FIR operation modes, the transmitter needs to send a serial infrared interaction pulse (SIP) at least once every 500 ms. The purpose of the SIP is to let slow devices (operating in SIR mode) know that the medium is currently occupied. The SIP pulse is shown in [Figure 19-22](#)

**Figure 19-22. SIP Pulse**


When the SIPMODE bit of Mode Definition Register 1 (MDR1[6]) equals 1, the TX state machine will always send one SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SENDSIP bit of the Auxiliary Control Register (ACREG[3]). The system (LH) can set ACREG[3] at least once every 500ms. The advantage of this approach over the default approach is that the TX state machine does not need to send the SIP at the end of each frame which may reduce the overhead required

#### 19.3.8.2.3 FIR Mode

In fast infrared mode (FIR), data transfers take place between LH and peripheral devices at 4 Mbit/s speed. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC-32, and ends with a stop flag.

**Figure 19-23. FIR Transmit Frame Format**


On transmit, the FIR transmit state machine attaches the preamble, start flag, CRC-32, and stop flag. It also encodes the transmit data into 4PPM format. It also generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes the start flag, decodes the 4PPM incoming data, and determines frame boundary with a reception of the stop flag. It also checks for errors such as an illegal symbol, a CRC error, and a frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

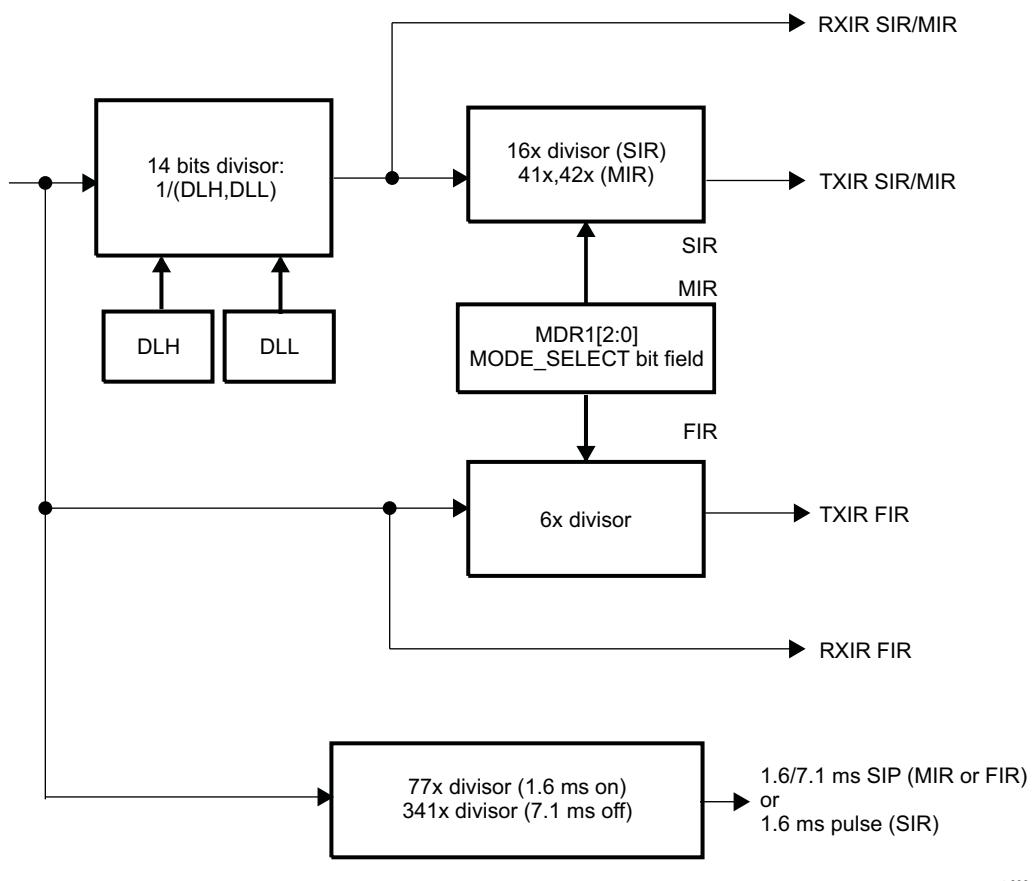
Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See bit 5 in [Section 19.5.1.26, Auxiliary Control Register](#), for a description of the logical operation. **Note:** This applies to all three modes of SIR, MIR, and FIR.

#### 19.3.8.2.4 IrDA Clock Generation: Baud Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

[Figure 19-24](#) shows the baud rate generator and associated controls.

**Figure 19-24. Baud Rate Generator**



uart-033

**CAUTION**

Before initializing or modifying clock parameter controls (UARTi.UART\_DLH, UARTi.UART\_DLL), MODE\_SELECT=DISABLE (UARTi.UART\_MDR1[2:0]) must be set to 0x7). Failure to observe this rule can result in unpredictable module behavior.

#### 19.3.8.2.5 Choosing the Appropriate Divisor Value

Three divisor values are:

- SIR mode: Divisor value = Operating frequency/(16x baud rate)
- MIR mode: Divisor value = Operating frequency/(41x/42x baud rate)
- FIR mode: Divisor value = None

[Table 19-28](#) lists the IrDA baud rate settings.

**Table 19-28. IrDA Baud Rate Settings**

| Baud Rate     | IR Mode | Baud Multiple | Encoding | DLH, DLL<br>(Decimal) | Actual Baud<br>Rate        | Error (%) | Source Jitter<br>(%) | Pulse Duration |
|---------------|---------|---------------|----------|-----------------------|----------------------------|-----------|----------------------|----------------|
| 2.4 kbps      | SIR     | 16x           | 3/16     | 1250                  | 2.4 kbps                   | 0         | 0                    | 78.1 $\mu$ s   |
| 9.6 kbps      | SIR     | 16x           | 3/16     | 312                   | 9.6153 kbps                | +0.16     | 0                    | 19.5 $\mu$ s   |
| 19.2 kbps     | SIR     | 16x           | 3/16     | 156                   | 19.231 kbps                | +0.16     | 0                    | 9.75 $\mu$ s   |
| 38.4 kbps     | SIR     | 16x           | 3/16     | 78                    | 38.462 kbps                | +0.16     | 0                    | 4.87 $\mu$ s   |
| 57.6 kbps     | SIR     | 16x           | 3/16     | 52                    | 57.692 kbps                | +0.16     | 0                    | 3.25 $\mu$ s   |
| 115.2 kbps    | SIR     | 16x           | 3/16     | 26                    | 115.38 kbps                | +0.16     | 0                    | 1.62 $\mu$ s   |
| 0.576<br>Mbps | MIR     | 41x/42x       | 1/4      | 2                     | 0.5756 Mbps <sup>(1)</sup> | 0         | +1.63/-0.80          | 416 ns         |
| 1.152<br>Mbps | MIR     | 41x/42x       | 1/4      | 1                     | 1.1511 Mbps <sup>(1)</sup> | 0         | +1.63/-0.80          | 208 ns         |
| 4 Mbps        | FIR     | 6x            | 4 PPM    | —                     | 4 Mbps                     | 0         | 0                    | 125 ns         |

<sup>(1)</sup> Average value

---

**NOTE:** Baud rate error and source jitter table values do not include 48-MHz reference clock error and jitter.

---

### 19.3.8.2.6 IrDA Data Formatting

The methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

#### 19.3.8.2.6.1 IR RX Polarity Control

The UARTi.UART\_MDR2[6] IRRXINVERT bit provides the flexibility to invert the uarti\_rx\_irrx pin in the UART to ensure that the protocol at the output of the transceiver has the same polarity at module level. By default, the uarti\_rx\_irrx pin is inverted because most transceivers invert the IR receive pin.

#### 19.3.8.2.6.2 IrDA Reception Control

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Operation of the uarti\_rx\_irrx input can be disabled by the UARTi.UART\_ACREG[5] DIS\_IR\_RX bit.

#### 19.3.8.2.6.3 IR Address Checking

In all IR modes, when address checking is enabled, only frames intended for the device are written to the RX FIFO. This restriction avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives, with the UARTi.UART\_XON1\_ADDR1[7:0] XON\_WORD1 and UARTi.UART\_XON2\_ADDR2[7:0] XON\_WORD2 bit fields.

Setting the UART\_EFR[0] bit to 1 selects address1 checking. Setting the UART\_EFR[1] bit to 1 selects address2 checking. Setting the UART\_EFR[1:0] bit field to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for private and public addresses.

If address checking is disabled, all received frames write to the RX FIFO.

#### 19.3.8.2.6.4 Frame Closing

A transmission frame can be terminated in two ways:

- Frame-length method: Set the UARTi.UART\_MDR1[7] FRAME\_END\_MODE bit to 0. The MPU writes the value of the frame length to the UARTi.UART\_TXFLH and UARTi.UART\_TXFLL registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the value of the frame length.
- Set-EOT bit method: Set the FRAME\_END\_MODE bit to 1. The MPU writes 1 to the UARTi.UART\_ACREG[0] EOT bit just before it writes the last byte to the TX FIFO. When the MPU writes the last byte to the TX FIFO, the device internally sets the tag bit for that character in the TX FIFO. As the TX state-machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and correctly terminate the frame.

#### 19.3.8.2.6.5 Store and Controlled Transmission

In store and controlled transmission (SCT) mode, the MPU starts writing data to the TX FIFO. Then, after writing a part of a frame (for a bigger frame) or an entire frame (a small frame; that is, a supervisory frame), the MPU writes 1 to the UARTi.UART\_ACREG[2] SCTX\_EN bit (deferred TX start) to start transmission.

SCT mode is enabled by setting the UARTi.UART\_MDR1[5] SCT bit to 1. This transmission method differs from normal mode, in which data transmission starts immediately after data is written to the TX FIFO. SCT mode is useful for sending short frames without TX underrun.

#### 19.3.8.2.6.6 Error Detection

When the UARTi.UART\_LSR register is read, the UARTi.UART\_LSR[4:2] bit field reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read).

The error is triggered by an interrupt (for IrDA mode interrupts, see [Table 19-12](#)). The STATUS FIFO must be read until empty (a maximum of eight reads is required).

#### 19.3.8.2.6.7 Underrun During Transmission

Underrun during transmission occurs when the TX FIFO is empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission.

Underrun also causes an internal flag to be set, which disables additional transmissions. Before the next frame can be transmitted, the MPU must:

- Reset the TX FIFO.
- Read the UARTi.UART\_RESUME register, which clears the internal flag.

This function can be disabled by the UARTi.UART\_ACREG[4] DIS\_TX\_UNDERRUN bit, compensated by the extension of the stop-bit in transmission if the TX FIFO is empty.

#### 19.3.8.2.6.8 Overrun During Receive

Overrun during receive for the IrDA mode has the same function as that for the UART mode (see [Section 19.3.8.1.3.6, Overrun During Receive](#)).

#### 19.3.8.2.6.9 Status FIFO

In IrDA modes, a status FIFO records the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written to the status FIFO.

Reading the UARTi.UART\_SFREGH[3:0] MSB and UARTi.UART\_SFREGL[3:0] (LSB) bit fields obtains the frame length. The frame error status is read in the UARTi.UART\_SFLSR register. Reading the UARTi.UART\_SFLSR register increments the status FIFO read pointer. Because the status FIFO is eight entries deep, it can hold the status of eight frames.

The MPU uses the frame-length information to locate the frame boundary in the received frame data. The MPU can screen bad frames using the error status information and can later request the sender to resend only the bad frames.

This status FIFO can be used effectively in DMA mode because the MPU must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

#### **19.3.8.2.7 SIR Mode Data Formatting**

This section provides specific instructions for SIR mode programming.

##### **19.3.8.2.7.1 Abort Sequence**

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

A transmission frame can be aborted by setting the `UARTi.UART_ACREG[1] ABORT_EN` bit to 1. When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

When a 0x7D character followed immediately by a 0xC1 character is received without transparency, the receiver treats a frame as an aborted frame.

##### **CAUTION**

When the TX FIFO is not empty and the `UARTi.UART_MDR1[5] SCT` bit is set to 1, the UART IrDA starts a new transfer with data of a previous frame when the aborted frame is sent. Therefore, the TX FIFO must be reset before sending an aborted frame.

##### **19.3.8.2.7.2 Pulse Shaping**

SIR mode supports the 3/16 or the 1.6- $\mu$ s pulse duration methods in receive and transmit. The `UARTi.UART_ACREG[7] PULSE_TYPE` bit selects the pulse width method in the transmit mode.

##### **19.3.8.2.7.3 SIR Free Format Programming**

The SIR FF mode is selected by setting the module in the UART mode (`UARTi.UART_MDR1[2:0] MODE_SELECT = 0x0`) and the `UARTi.UART_MDR2[3] UART_PULSE` bit to 1 to allow pulse shaping.

Because the bit format stays the same, some UART mode configuration registers must be set at specific values:

- `UARTi.UART_LCR[1:0] CHAR_LENGTH` bit field = 0x3 (8 data bits)
- `UARTi.UART_LCR[2] NB_STOP` bit = 0x0 (1 stop-bit)
- `UARTi.UART_LCR[3] PARITY_EN` bit = 0x0 (no parity)

The UART mode interrupts are used for the SIR FF mode, but many are not relevant (XOFF, RTS, CTS, modem status register, etc.).

##### **19.3.8.2.8 MIR and FIR Mode Data Formatting**

This section describes common instructions for FIR and MIR mode programming.

At the end of a frame reception, the MPU reads the line status register (`UARTi.UART_LSR`) to detect errors in the received frame.

When the `UARTi.UART_MDR1[6] SIP_MODE` bit is set to 1, the TX state-machine always sends one SIP at the end of a transmission frame. However, when the `SIP_MODE` bit is set to 0, SIP transmission depends on the `UARTi.UART_ACREG[3] SEND_SIP` bit.

The MPU can set the `SEND_SIP` bit at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not have to send the SIP at the end of each frame, thus reducing the overhead required.

### 19.3.8.3 CIR Mode

In consumer infrared mode, the infrared operation is designed to function as a programmable (universal) remote control. By setting the MDR1 register, the UART can be set to CIR mode in the same way as the other IrDA modes are set using the MDR1 register.

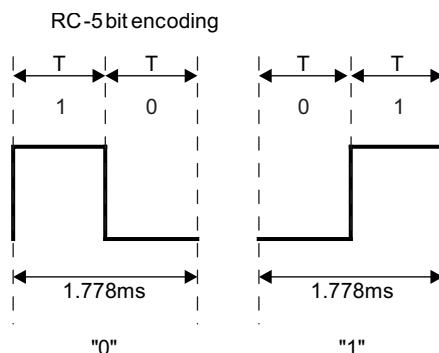
The CIR mode uses a variable pulse width modulation technique (based on multiples of a programmable T period) to encompass the various formats of infrared encoding for remote control applications. The CIR logic is to transmit and receive data packets according to the user definable frame structure and packet content.

#### 19.3.8.3.1 Consumer IR Encoding

There are two distinct methods of encoding for remote control applications. The first uses time extended bit forms i.e. a variable pulse distance (or duration) whereby the difference between a logic one and logic zero is the length of the pulse width; and the second is the use of a bi-phase where the encoding of the logic zero and one is in the change of signal level from 1→0 or 0→1 respectively. Japanese manufacturers tend to favor the use of pulse duration encoding whereas European manufacturers favor the use of bi-phase encoding.

The CIR mode is designed to use a completely flexible free format encoding where a digit '1' from the TX/RX FIFO is to be transmitted/received as a modulated pulse with duration T. Equally, a '0' is to be transmitted/received as a blank duration T. The protocol of the data is to be constructed and deciphered by the host CPU. For example, the RC-5 protocol using Manchester encoding can be emulated as using a "01" pair for one and "10" pair for a zero.

**Figure 19-25. RC-5 Bit Encoding**

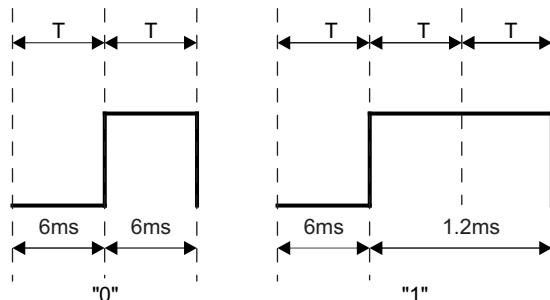


Since the CIR mode logic does not impose a fixed format for infrared packets of data, the CPU software is at liberty to define the format through the use of simple data structures that will then be modulated into an industry standard, such as RC5 or SIRC. To send a sequence of "0101" in RC5, the host software must write an eight bit binary character of "10011001" to the data TX FIFO of the UART.

For SIRC, the modulation length (i.e. multiples of T) is the method to distinguish between a "1" or a "0". The following SIRC digits show the difference in encoding between this and RC5 for example. Note: the pulse width is extended for "1" digits.

**Figure 19-26. SIRC Bit Encoding**

SIRC bit encoding



To construct comprehensive packets that constitute remote control commands, the host software must combine a number of eight bit data characters in a sequence that follows one of the universally accepted formats. For illustrative purposes, a standard RC5 frame is described below (the SIRC format follows this). Each of the above fields in RC5 can be considered as two T pulses (digital bits) from the TX and RX FIFOs.

The standard RC5 format as seen by the UART\_IrDA in CIR mode.

**Figure 19-27. RC-5 Standard Packet Format**

|    |    |   |    |    |    |    |    |    |    |    |    |    |    |
|----|----|---|----|----|----|----|----|----|----|----|----|----|----|
| S1 | S2 | T | A4 | A3 | A2 | A1 | A0 | C5 | C4 | C3 | C2 | C1 | C0 |
|----|----|---|----|----|----|----|----|----|----|----|----|----|----|

Where:

S1, S2: Start bits (always 1)

T: Toggle bit

A4–A0: Address (or system) bits

C5–C0: Command bits

The toggle bit T changes each time a new command is transmitted to allow detection of pressing the same key twice (or effectively receiving the same data from the host consecutively). Since a code is being sent as long as the CPU transmits characters to the UART for transmission, a brief delay in the transmission of the same command would be detected by the use of the toggle bit. The address bits define the machine or device that the Infrared transmission is intended for and the command defines the operation.

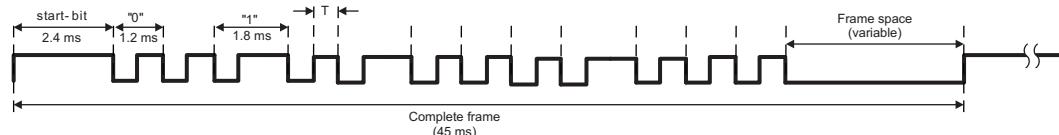
To accommodate an extended RC5 format, the S2 bit is replaced by a further command bit (C6) that allows the command range to increase to 7-bits. This format is known as the extended RC-5 format.

The SIRC encoding uses the duration of modulation for mark and space; hence the duration of data bits inside the standard frame length will vary depending upon the logic 1 content. The packet format and bit encoding is illustrated below. There is one start bit of two milliseconds and control codes followed by data that constitute the whole frame.

**Figure 19-28. SIRC Packet Format**

|   |    |    |    |    |    |    |    |    |    |    |    |    |
|---|----|----|----|----|----|----|----|----|----|----|----|----|
| S | C0 | C1 | C2 | C3 | C4 | C5 | C6 | D0 | D1 | D2 | D3 | D4 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|

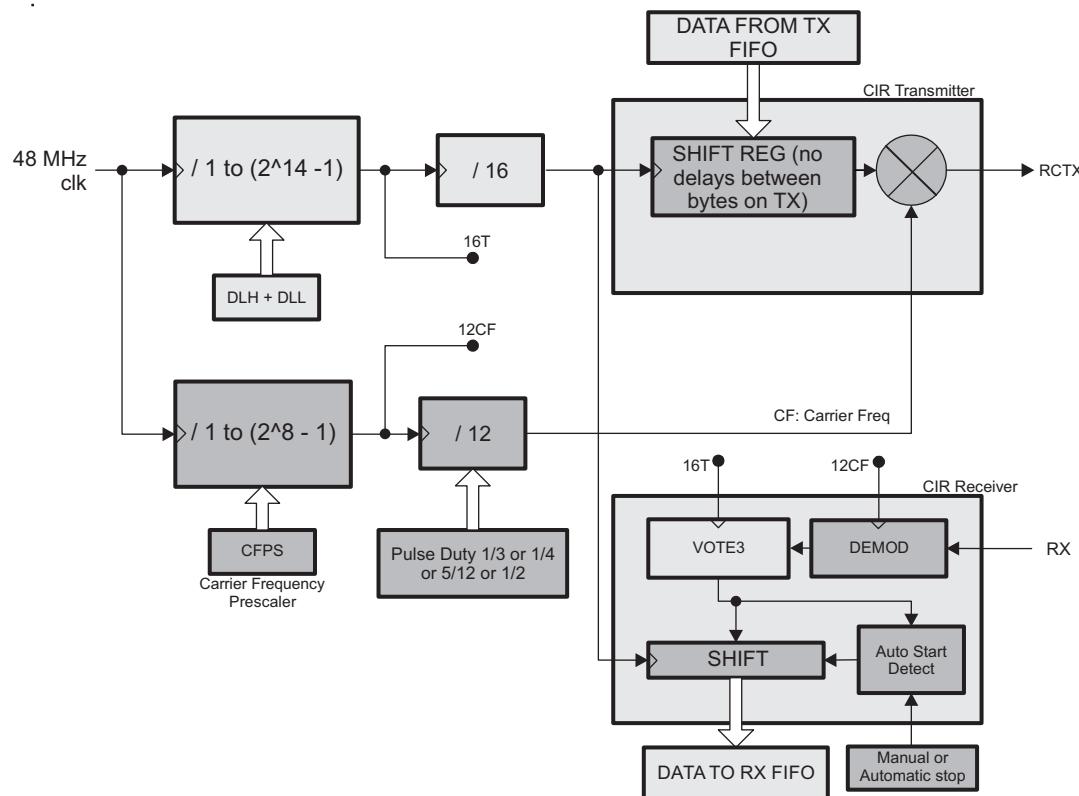
It should be noted that the encoding must take a standard duration but the contents of the data may vary. This implies that the control software for emitting and receiving data packets must exercise a scheme of inter-packet delay, where the emission of successive packets can only be done after a real time delay has expired.

**Figure 19-29. SIRC Bit Transmission Example**


It is beyond the scope of this document to describe all encoding methods and techniques, the previous information was provided to illustrate the consideration required to employ different encoding methods for different industry standard protocols. The user should refer to industry standard documentation for specific methods of encoding and protocol usage.

#### 19.3.8.3.2 CIR Mode Operation

Depending on the encoding method (variable pulse distance / bi-phase), the LH should develop a data structure that combines the 1 and 0 with a T period in order to encode the complete frame to transmit. This can then be transmitted to the infrared output with a method of modulation shown in the following diagram.

**Figure 19-30. CIR Mode Block Components**


In transmission, the LH software must exercise an element of real time control for transmitting data packets; they must each be emitted at a constant delay from the start bits of each of the individual packets which means when sending a series of packets, the packet to packet delay must respect a specific delay. To control this delay 2 methods can be used:

- By filling the TX FIFO with a number of zero bit which is transmitted with a T period.
- By using an external system timer which controls the delay either between each start of frame or between the end of a frame and the start of the next one. This can be performed:
  - By controlling the start of the frame through the configuration register MDR1[5] and ACREG[2] depending on the timer status (in case of control the delay between each start of frame).
  - By using the TX\_STATUS interrupt IIR[5] to pre-load the next frame in the TX FIFO and to control the start of the timer (in case of control the delay between end of frame and start of next frame).

In reception, there are two ways to stop it :

- The LH can disable the reception by setting the ACREG[5] to 1 when it considers that the reception is finished because a large number of 0 has been received. To receive a new frame, the ACREG[5] must be set to 0.
- A specific mechanism, depending on the value set in the BOF length register (EBLR), allows for automatically stopping the reception. If the value set in the EBLR register is different than 0, this feature is enabled and counts a number of bits received at 0. When the counter achieves the value defined in the EBLR register, the reception is automatically stopped and RX\_STOP\_IT (IIR[2]) is set. When a 1 is detected on the RCRX pin, the reception is automatically enabled.

**Note:** There's a limitation when receiving data in UART CIR mode. The IrDA transceivers on the market have a common characteristic that shrinks the hold time of the received modulation pulse. The UART filtering schema on receiving is based on the same encoding mechanism used in transmission.

For the following scenario:

- Shift register period: 0.9 us
- Modulation frequency: 36 KHz
- Duty cycle: 1/4 of a modulation frequency period

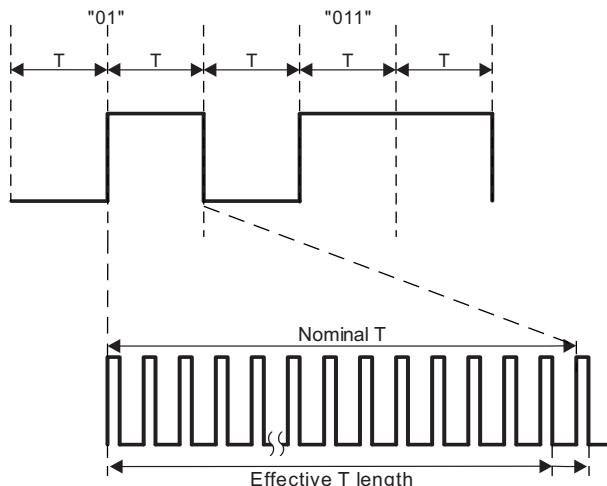
The data sent in these conditions would look like 7us pulses within 28us period. The UART expects to receive similar incoming data on receive, but available transceiver timing characteristics typically send 2us modulated pulses. Those will be filtered out and RX FIFO won't receive any data.

This does not affect UART CIR mode in transmission.

**Note:** The CIR RX demodulation can be bypassed by setting the MDR3[0] register bit.

### 19.3.8.3.3 Carrier Modulation

Looking closer at the actual modulation pulses of the infrared data stream, it should be noted that each modulated pulse that constitutes a digit is in fact a train of on/off pulses.

**Figure 19-31. CIR Pulse Modulation**

A minimum of 4 modulation pulses per bit is required by the module.

Based on the requested modulation frequency, the CFPS register must be set with the correct dividing value to provide the more accurate pulse frequency:

$$\text{Dividing value} = (\text{FCLK}/12)/\text{MODfreq}$$

Where FCLK = System clock frequency (48 MHz)

12 = real value of BAUD multiple

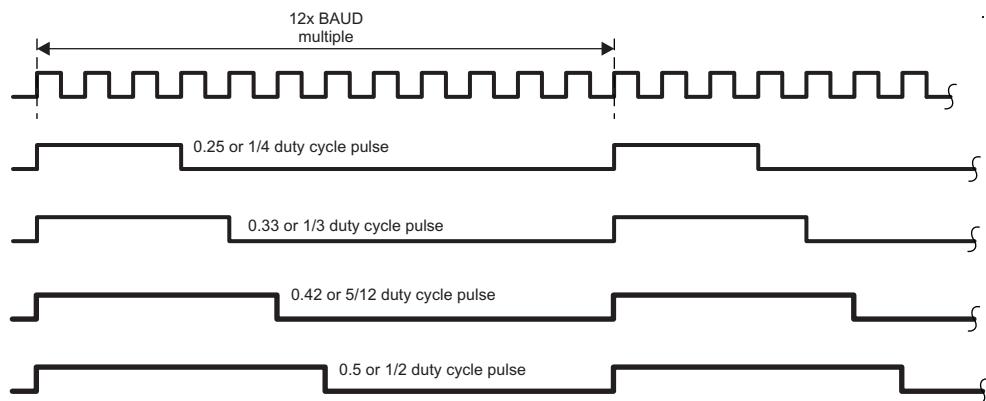
MODfreq = Effective frequency of the modulation (MHz)

**Example:** For a targeted modulation frequency of 36 kHz, the CFPS value must be set to 111 in decimal which provide an modulation frequency of 36.04 kHz.

**Note:** The CFPS register is to start with a reset value of 105 (decimal) which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user defined by the pulse duty register bits in the MDR2 configuration register.

| MDR2[5:4] | Duty Cycle (High Level) |
|-----------|-------------------------|
| 00        | 1/4                     |
| 01        | 1/3                     |
| 10        | 5/12                    |
| 11        | 1/2                     |

**Figure 19-32. CIR Modulation Duty Cycle**

The transmission logic ensures that all pulses are transmitted completely; i.e., there is no cut off of any pulses during its transmission. Furthermore, while transmitting continuous bytes back-to-back, no delay is inserted between two transmitted bytes. **Note:** The CIR RX demodulation can be bypassed by setting the MDR3[0] register bit. This bit will not affect the transmission modulation.

#### 19.3.8.3.4 Frequency Divider Values

The data transferred is a succession of pulse with a T period. Depending on the standards used, the T period is defined through the DLL and DLH registers which defined the value to divide the functional clock (48 MHz):

Dividing value =  $(FCLK/16)/T_{freq}$

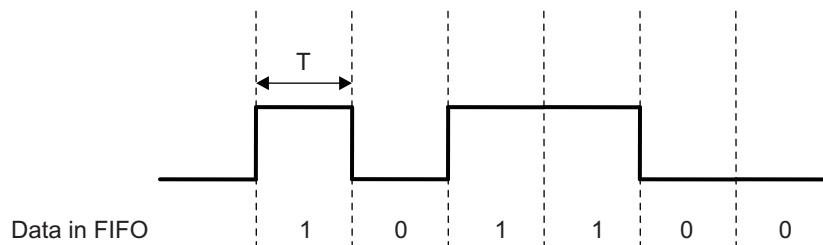
Where FCLK = System clock frequency (48 MHz)

16 = real value of BAUD multiple

$T_{freq}$  = Effective frequency of the T pulse (MHz)

In an example case using a variable pulse duration definitions:

**Figure 19-33. Variable Pulse Duration Definitions**



For a logical "1", the pulse duration is equal to  $2T$  and for a logical "0", it's equal to  $4T$ .

If  $T = 0.56$  ms, the value coded into the DLH and DLL register must be 1680 in decimal.

## 19.4 UART/IrDA/CIR Basic Programming Model

### 19.4.1 UART Programming Model

#### 19.4.1.1 Quick Start

This section describes the procedure for operating the UART with FIFO and DMA or interrupts. This three-part procedure ensures the quick start of the UART. It does not cover every UART feature.

The first programming model covers software reset of the UART. The second programming model describes FIFO and DMA configuration. The last programming model describes protocol, baud rate, and interrupt configuration.

---

**NOTE:** Each programming model can be used independently of the other two; for instance, reconfiguring the FIFOs and DMA settings only.

Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see [Section 19.3.7.1, Register Access Modes](#).

---

#### 19.4.1.1.1 Software Reset

To clear the UART registers, perform the following steps:

1. Initiate a software reset:  
Set the `UARTi.UART_SYSC[1]` SOFTRESET bit to 1.
2. Wait for the end of the reset operation:  
Poll the `UARTi.UART_SYSS[0]` RESETDONE bit until it equals 1.

#### 19.4.1.1.2 FIFOs and DMA Settings

To enable and configure the receive and transmit FIFOs and program the DMA mode, perform the following steps:

1. Switch to register configuration mode B to access the `UARTi.UART_EFR` register:
  - (a) Save the current `UARTi.UART_LCR` register value.
  - (b) Set the `UARTi.UART_LCR` register value to `0x00BF`.
2. Enable register submode `TCR_TLR` to access the `UARTi.UART_TLR` register (part 1 of 2):
  - (a) Save the `UARTi.UART_EFR[4]` ENHANCED\_EN value.
  - (b) Set the `UARTi.UART_EFR[4]` ENHANCED\_EN bit to 1.
3. Switch to register configuration mode A to access the `UARTi.UART_MCR` register:  
Set the `UARTi.UART_LCR` register value to `0x0080`.
4. Enable register submode `TCR_TLR` to access the `UARTi.UART_TLR` register (part 2 of 2):
  - (a) Save the `UARTi.UART_MCR[6]` TCR\_TLR value.
  - (b) Set the `UARTi.UART_MCR[6]` TCR\_TLR bit to 1.
5. Enable the FIFO; load the new FIFO triggers (part 1 of 3) and the new DMA mode (part 1 of 2):  
Set the following bits to the desired values:
  - `UARTi.UART_FCR[7:6]` RX\_FIFO\_TRIG
  - `UARTi.UART_FCR[5:4]` TX\_FIFO\_TRIG
  - `UARTi.UART_FCR[3]` DMA\_MODE
  - `UARTi.UART_FCR[0]` FIFO\_ENABLE (0: Disable the FIFO; 1: Enable the FIFO)

**NOTE:** The UARTi.UART\_FCR register is not readable.

6. Switch to register configuration mode B to access the UARTi.UART\_EFR register:  
Set the UARTi.UART\_LCR register value to 0x00BF.
7. Load the new FIFO triggers (part 2 of 3):  
Set the following bits to the desired values:
  - UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA
  - UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA
8. Load the new FIFO triggers (part 3 of 3) and the new DMA mode (part 2 of 2):  
Set the following bits to the desired values:
  - UARTi.UART\_SCR[7] RX\_TRIG\_GRANU1
  - UARTi.UART\_SCR[6] TX\_TRIG\_GRANU1
  - UARTi.UART\_SCR[2:1] DMA\_MODE\_2
  - UARTi.UART\_SCR[0] DMA\_MODE\_CTL
9. Restore the UARTi.UART\_EFR[4] ENHANCED\_EN value saved in Step 2a.
10. Switch to register configuration mode A to access the UARTi.UART\_MCR register:  
Set the UARTi.UART\_LCR register value to 0x0080.
11. Restore the UARTi.UART\_MCR[6] TCR\_TLR value saved in Step 4a.
12. Restore the UARTi.UART\_LCR value saved in Step 1a.

Triggers are used to generate interrupt and DMA requests. See [Section 19.3.6.1.1, Transmit FIFO Trigger](#), to choose the following values:

- UARTi.UART\_FCR[5:4] TX\_FIFO\_TRIG
- UARTi.UART\_TLR[3:0] TX\_FIFO\_TRIG\_DMA
- UARTi.UART\_SCR[6] TX\_TRIG\_GRANU1

Triggers are used to generate interrupt and DMA requests. See [Section 19.3.6.1.2, Receive FIFO Trigger](#), to choose the following values:

- UARTi.UART\_FCR[7:6] RX\_FIFO\_TRIG
- UARTi.UART\_TLR[7:4] RX\_FIFO\_TRIG\_DMA
- UARTi.UART\_SCR[7] RX\_TRIG\_GRANU1

DMA mode enables DMA requests. See [Section 19.3.6.4, FIFO DMA Mode Operation](#), to choose the following values:

- UARTi.UART\_FCR[3] DMA\_MODE
- UARTi.UART\_SCR[2:1] DMA\_MODE\_2
- UARTi.UART\_SCR[0] DMA\_MODE\_CTL

#### 19.4.1.1.3 Protocol, Baud Rate, and Interrupt Settings

To program the protocol, baud rate, and interrupt settings, perform the following steps:

1. Disable UART to access the UARTi.UART\_DLL and UARTi.UART\_DLH registers:  
Set the UARTi.UART\_MDR1[2:0] MODE\_SELECT bit field to 0x7.
2. Switch to register configuration mode B to access the UARTi.UART\_EFR register:  
Set the UARTi.UART\_LCR register value to 0x00BF.
3. Enable access to the UARTi.UART\_IER[7:4] bit field:
  - (a) Save the UARTi.UART\_EFR[4] ENHANCED\_EN value.
  - (b) Set the UARTi.UART\_EFR[4] ENHANCED\_EN bit to 1.
4. Switch to register operational mode to access the UARTi.UART\_IER register:  
Set the UARTi.UART\_LCR register value to 0x0000.

5. Clear the `UARTi.UART_IER` register (set the `UARTi.UART_IER[4]` SLEEP\_MODE bit to 0 to change the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers). Set the `UARTi.UART_IER` register value to 0x0000.
6. Switch to register configuration mode B to access the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_LCR` register value to 0x00BF.
7. Load the new divisor value:  
Set the `UARTi.UART_DLL[7:0]` CLOCK\_LSB and `UARTi.UART_DLH[5:0]` CLOCK\_MSB bit fields to the desired values.
8. Switch to register operational mode to access the `UARTi.UART_IER` register:  
Set the `UARTi.UART_LCR` register value to 0x0000.
9. Load the new interrupt configuration (0: Disable the interrupt; 1: Enable the interrupt):  
Set the following bits to the desired values:
  - `UARTi.UART_IER[7]` CTS\_IT
  - `UARTi.UART_IER[6]` RTS\_IT
  - `UARTi.UART_IER[5]` XOFF\_IT
  - `UARTi.UART_IER[4]` SLEEP\_MODE
  - `UARTi.UART_IER[3]` MODEM\_STS\_IT
  - `UARTi.UART_IER[2]` LINE\_STS\_IT
  - `UARTi.UART_IER[1]` THR\_IT
  - `UARTi.UART_IER[0]` RHR\_IT
10. Switch to register configuration mode B to access the `UARTi.UART_EFR` register:  
Set the `UARTi.UART_LCR` register value to 0x00BF.
11. Restore the `UARTi.UART_EFR[4]` ENHANCED\_EN value saved in Step 3a.
12. Load the new protocol formatting (parity, stop-bit, character length) and switch to register operational mode:  
Set the `UARTi.UART_LCR[7]` DIV\_EN bit to 0.  
Set the `UARTi.UART_LCR[6]` BREAK\_EN bit to 0.  
Set the following bits to the desired values:
  - `UARTi.UART_LCR[5]` PARITY\_TYPE\_2
  - `UARTi.UART_LCR[4]` PARITY\_TYPE\_1
  - `UARTi.UART_LCR[3]` PARITY\_EN
  - `UARTi.UART_LCR[2]` NB\_STOP
  - `UARTi.UART_LCR[1:0]` CHAR\_LENGTH
13. Load the new UART mode:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to the desired value.

See [Section 19.3.8.1.2, Choosing the Appropriate Divisor Value](#), to choose the following values:

- `UARTi.UART_DLL[7:0]` CLOCK\_LSB
- `UARTi.UART_DLH[5:0]` CLOCK\_MSB
- `UARTi.UART_MDR1[2:0]` MODE\_SELECT

See [Section 19.3.8.1.3.1, Frame Formatting](#), to choose the following values:

- `UARTi.UART_LCR[5]` PARITY\_TYPE\_2
- `UARTi.UART_LCR[4]` PARITY\_TYPE\_1
- `UARTi.UART_LCR[3]` PARITY\_EN
- `UARTi.UART_LCR[2]` NB\_STOP
- `UARTi.UART_LCR[1:0]` CHAR\_LENGTH

### 19.4.1.2 Hardware and Software Flow Control Configuration

This section describes the programming steps to enable and configure hardware and software flow control. Hardware and software flow control cannot be used at the same time.

---

**NOTE:** Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see [Section 19.3.7.1, Register Access Modes](#).

---

#### 19.4.1.2.1 Hardware Flow Control Configuration

To enable and configure hardware flow control, perform the following steps:

1. Switch to register configuration mode A to access the `UARTi.UART_MCR` register:
  - (a) Save the current `UARTi.UART_LCR` register value.
  - (b) Set the `UARTi.UART_LCR` register value to 0x0080.
2. Enable register submode `TCR_TLR` to access the `UARTi.UART_TCR` register (part 1 of 2):
  - (a) Save the `UARTi.UART_MCR[6]` `TCR_TLR` value.
  - (b) Set the `UARTi.UART_MCR[6]` `TCR_TLR` bit to 1.
3. Switch to register configuration mode B to access the `UARTi.UART_EFR` register:  
Set the `UARTi.UART_LCR` register value to 0x00BF.
4. Enable register submode `TCR_TLR` to access the `UARTi.UART_TCR` register (part 2 of 2):
  - (a) Save the `UARTi.UART_EFR[4]` `ENHANCED_EN` value.
  - (b) Set the `UARTi.UART_EFR[4]` `ENHANCED_EN` bit to 1.
5. Load the new start and halt trigger values for hardware flow control:  
Set the following bits to the desired values:
  - `UARTi.UART_TCR[7:4]` `AUTO_RTS_START`
  - `UARTi.UART_TCR[3:0]` `AUTO_RTS_HALT`
6. Enable or disable receive and transmit hardware flow control mode and restore the `UARTi.UART_EFR[4]` `ENHANCED_EN` value saved in Step 4a.  
Set the following bits to the desired values:
  - `UARTi.UART_EFR[7]` `AUTO_CTS_EN` (0: Disable; 1: Enable)
  - `UARTi.UART_EFR[6]` `AUTO_RTS_EN` (0: Disable; 1: Enable)  
Restore the `UARTi.UART_EFR[4]` `ENHANCED_EN` bit to the saved value.
7. Switch to register configuration mode A to access the `UARTi.UART_MCR` register:  
Set the `UARTi.UART_LCR` register value to 0x0080.
8. Restore the `UARTi.UART_MCR[6]` `TCR_TLR` value saved in Step 2a.
9. Restore the `UARTi.UART_LCR` value saved in Step 1a.

See [Section 19.3.8.1.3.2, Hardware Flow Control](#), to choose the following values:

- `UARTi.UART_EFR[7]` `AUTO_CTS_EN`
- `UARTi.UART_EFR[6]` `AUTO_RTS_EN`
- `UARTi.UART_TCR[7:4]` `AUTO_RTS_START`
- `UARTi.UART_TCR[3:0]` `AUTO_RTS_HALT`

#### 19.4.1.2.2 Software Flow Control Configuration

To enable and configure software flow control, perform the following steps:

1. Switch to register configuration mode B to access the `UARTi.UART_EFR` register.

- (a) Save the current `UARTi.UART_LCR` register value.  
 (b) Set the `UARTi.UART_LCR` register value to `0x00BF`.
2. Enable register submode XOFF to access the `UARTi.UART_XOFF1` and `UARTi.UART_XOFF2` registers:
    - (a) Save the `UARTi.UART_EFR[4]` ENHANCED\_EN value.
    - (b) Set the `UARTi.UART_EFR[4]` ENHANCED\_EN bit to 0.
  3. Load the new software flow control characters:  
 Set the following bits to the desired values:
    - `UARTi.UART_XON1_ADDR1[7:0]` XON\_WORD1
    - `UARTi.UART_XON2_ADDR2[7:0]` XON\_WORD2
    - `UARTi.UART_XOFF1[7:0]` XOFF\_WORD1
    - `UARTi.UART_XOFF2[7:0]` XOFF\_WORD2
  4. Enable access to the `UARTi.UART_MCR[7:5]` bit field and enable register submode TCR\_TLR to access the `UARTi.UART_TCR` register (part 1 of 2):
 

Set the `UARTi.UART_EFR[4]` ENHANCED\_EN bit to 1.
  5. Switch to register configuration mode A to access the `UARTi.UART_MCR` register:  
 Set the `UARTi.UART_LCR` register value to `0x0080`.
  6. Enable register submode TCR\_TLR to access the `UARTi.UART_TCR` register (part 2 of 2) and enable or disable XON any function:
    - (a) Save the `UARTi.UART_MCR[6]` TCR\_TLR value.
    - (b) Set the `UARTi.UART_MCR[6]` TCR\_TLR bit to 1.
    - (c) Set the `UARTi.UART_MCR[5]` XON\_EN bit to the desired value (0: Disable; 1: Enable).
  7. Switch to register configuration mode B to access the `UARTi.UART_EFR` register:  
 Set the `UARTi.UART_LCR` register value to `0x00BF`.
  8. Load the new start and halt trigger values for software flow control:  
 Set the following bits to the desired values:
    - `UARTi.UART_TCR[7:4]` AUTO\_RTS\_START
    - `UARTi.UART_TCR[3:0]` AUTO\_RTS\_HALT
  9. Enable or disable special character function and load the new software flow control mode and restore the `UARTi.UART_EFR[4]` ENHANCED\_EN value saved in Step 2a:  
 Set the following bits to the desired values:
    - `UARTi.UART_EFR[5]` SPEC\_CHAR (0: Disable; 1: Enable)
    - `UARTi.UART_EFR[3:0]` SW\_FLOW\_CONTROL
 Restore the `UARTi.UART_EFR[4]` ENHANCED\_EN bit to the saved value.
  10. Switch to register configuration mode A to access the `UARTi.UART_MCR` register:  
 Set the `UARTi.UART_LCR` register value to `0x0080`.
  11. Restore the `UARTi.UART_MCR[6]` TCR\_TLR bit value saved in Step 6a.
  12. Restore the `UARTi.UART_LCR` value saved in Step 1a.
- See [Section 19.3.8.1.3.3, Software Flow Control](#), to choose the following values:
- `UARTi.UART_EFR[5]` SPEC\_CHAR
  - `UARTi.UART_EFR[3:0]` SW\_FLOW\_CONTROL
  - `UARTi.UART_TCR[7:4]` AUTO\_RTS\_START
  - `UARTi.UART_TCR[3:0]` AUTO\_RTS\_HALT
  - `UARTi.UART_XON1_ADDR1[7:0]` XON\_WORD1
  - `UARTi.UART_XON2_ADDR2[7:0]` XON\_WORD2
  - `UARTi.UART_XOFF1[7:0]` XOFF\_WORD1

- UARTi.UART\_XOFF2[7:0] XOFF\_WORD2

## 19.4.2 IrDA Programming Model

### 19.4.2.1 SIR Mode

#### 19.4.2.1.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with parity forced to 1, baud rate = 112.5KB, FIFOs disabled, 2 stop-bits, and 8-bit word length:

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to the `UART_DLL` and `UART_DLH` registers (the `UART_LCR[7]` DIV\_EN bit = 1):  
`UARTi.UART_LCR` = 0x80 (Data format is unaffected by the use and settings of the `UARTi.UART_LCR` register in IrDA mode.)
3. Load the new baud rate (115.2 kbps):  
`UARTi.UART_DLL` = 0x1A  
`UARTi.UART_DLH` = 0x00
4. Set SIR mode:  
`UARTi.UART_MDR1[2:0]` MODE\_SELECT = 0x1
5. Disable access to the `UART_DLL` and `UART_DLH` registers and switch to register operational mode:  
`UARTi.UART_LCR` = 0x00.
6. Optional: Enable the RHR interrupt:  
`UARTi.UART_IER[0]` RHR\_IT = 0x1

#### 19.4.2.1.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 6-byte frame with no parity, baud rate = 112.5 kbps, FIFOs disabled, 3/16 encoding, 2 stop-bits, and 7-bit word length:

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to the `UART_EFR` register:  
`UARTi.UART_LCR` = 0xBF
3. Enable the enhanced features (the `UART_EFR[4]` ENAHNCED\_EN bit = 1):  
Set the `UARTi.UART_EFR` register value to 0x10.
4. Grant access to the `UART_DLL` and `UART_DLH` registers (the `UART_LCR[7]` DIV\_EN bit = 1):  
`UARTi.UART_LCR` = 0x80 (Data format is unaffected by the use and settings of the `UARTi.UART_LCR` register in IrDA mode.)
5. Load the new baud rate (115.2 kbps):  
`UARTi.UART_DLL` = 0x1A  
`UARTi.UART_DLH` = 0x00
6. Set SIR mode (the `UART_MDR1[2:0]` MODE\_SELECT bit field = 0x1):  
`UARTi.UART_MDR1` = 0x01
7. Disable access to the `UART_DLL` and `UART_DLH` registers and switch to register operational mode:  
`UARTi.UART_LCR` = 0x00.
8. Force DTR output to active:  
`UARTi.UART_MCR[0]` DTR = 1
9. Optional: Enable the THR interrupt:  
`UARTi.UART_IER[1]` THR\_IT = 1
10. Set transmit frame length to 6 bytes:  
`UARTi.UART_TXFLL` = 0x06
11. Set 7 starts of frame transmission:  
`UARTi.UART_EBLR` = 0x08
12. Optional: Set SIR pulse width to be 1.6 us:  
`UARTi.UART_ACREG[7]` PULSE\_TYPE = 1
13. Load the `UART_THR` register with the data to be transmitted.

## 19.4.2.2 MIR Mode

### 19.4.2.2.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with no parity, baud rate = 1.152 Mbps, and FIFOs disabled.

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to the `UART_DLL` and `UART_DLH` registers (`UART_LCR[7]` DIV\_EN bit = 1):  
`UARTi.UART_LCR` = 0x80 (Data format is unaffected by the use and settings of the `UARTi.UART_LCR` register in IrDA mode.)
3. Load the new baud rate (1.152 Mbps):  
`UARTi.UART_DLL` = 0x01  
`UARTi.UART_DLH` = 0x00
4. Set MIR mode:  
`UARTi.UART_MDR1[2:0]` MODE\_SELECT = 0x4
5. Disable access to the `UART_DLL` and `UART_DLH` registers and switch to register operational mode:  
`UARTi.UART_LCR` = 0x00
6. Force DTR output to active (`UART_MCR[0]` DTR = 1):  
Force RTS output to active (`UART_MCR[1]` RTS = 1).  
`UARTi.UART_MCR` = 0x3
7. Optional: Enable the RHR interrupt:  
`UARTi.UART_IER[0]` RHR\_IT = 1

### 19.4.2.2.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 60-byte frame with no parity, baud rate = 1.152 Mbps, and FIFOs disabled:

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to the `UART_DLL` and `UART_DLH` registers (`UART_LCR[7]` DIV\_EN bit = 1):  
`UARTi.UART_LCR` = 0x80 (Data format is unaffected by the use and settings of the `UARTi.UART_LCR` register in IrDA mode.)
3. Load the new baud rate (1.152 Mbps):  
`UARTi.UART_DLL` = 0x01  
`UARTi.UART_DLH` = 0x00
4. Set MIR mode:  
`UARTi.UART_MDR1[2:0]` MODE\_SELECT = 0x4
5. Disable access to the `UART_DLL` and `UART_DLH` registers and switch to register operational mode:  
`UARTi.UART_LCR` = 0x00
6. Force DTR output to active:  
`UARTi.UART_MCR[0]` DTR = 1
7. Optional: Enable the THR interrupt:  
`UARTi.UART_IER[1]` THR\_IT = 1
8. Set the frame length to 60 bytes:  
`UARTi.UART_TXFLL` = 0x3C
9. Optional: Transmit eight additional starts of frame (MIR mode requires two starts):  
`UARTi.UART_EBLR` = 0x08
10. SIP is sent at the end of transmission:  
`UARTi.UART_ACREG[3]` = 1
11. Load the `UART_THR` register with the data to be transmitted.

### 19.4.2.3 FIR Mode

#### 19.4.2.3.1 Receive

The following programming model explains how to program the module to receive the IrDA frame with no parity, baud rate = 4 Mbps, FIFOs enabled, 8-bit word length.

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to the `UART_DLL` and `UART_DLH` registers (`UART_LCR[7]` DIV\_EN bit = 1):  
`UARTi.UART_LCR` = 0x80 (Data format is unaffected by the use and settings of the `UARTi.UART_LCR` register in IrDA mode.)
3. FIFO clear and enable:  
`UARTi.UART_FCR` = 0x7 (TX/RX FIFO trigger: `UART_FCR[7:6]` and `UART_FCR[5:4]`)  
`UARTi.UART_LCR[7]` = 0
4. Set FIR mode:  
`UARTi.UART_MDR1[2:0]` MODE\_SELECT = 0x5
5. Set frame length:  
`UARTi.UART_RXFLL` = 0xA (Data + CRC + STOP)
6. Disable access to the `UARTi.UART_DLL` registers and `UARTi.UART_DLH` and switch to register operational mode:  
`UARTi.UART_LCR[7]` DIV\_EN = 0x0
7. Optional: Enable the RHR interrupt:  
`UARTi.UART_IER[0]` RHR\_IT = 1

#### 19.4.2.3.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 4-byte frame with no parity, baud rate = 4 Mbps, FIFOs enabled, and 8-bit word length.

1. Disable the UART before accessing the `UARTi.UART_DLL` and `UARTi.UART_DLH` registers:  
Set the `UARTi.UART_MDR1[2:0]` MODE\_SELECT bit field to 0x7.
2. Grant access to EFR\_REG:  
`UARTi.UART_LCR` = 0xBF
3. Enable the enhanced features (EFR\_REG[4] ENAHNCED\_EN = 0x1):  
`UARTi.UART_EFR` = 0x10
4. FIFO clear and enable:  
`UARTi.UART_FCR` = 0x7 (TX/RX FIFO trigger: `UART_FCR[7:6]` and `UART_FCR[5:4]`).  
`UARTi.UART_LCR[7]` = 0
5. Set FIR mode and enable auto-SIP mode:  
`UARTi.UART_MDR1` = 0x45
6. Set frame length:  
`UARTi.UART_TXFLL` = 0x4  
`UARTi.UART_TXFLH` = 0x0  
`UARTi.UART_RXFLL` = 0xA (Data + CRC + STOP)  
`UARTi.UART_RXFLH` = 0x0
7. Force DTR output to active:  
`UARTi.UART_MCR[0]` DTR = 0x1
8. Optional: Enable the THR interrupt:  
`UARTi.UART_IER[1]` THR\_IT = 0x1
9. Optional: Transmit eight additional starts of frame (MIR mode requires two starts):  
`UARTi.UART_EBLR` = 0x08
10. SIP is sent at the end of transmission:  
`UARTi.UART_ACREG[3]` = 1
11. Load the `UART_THR` register with the data to be transmitted.

## 19.5 UART Registers

### 19.5.1 UART Registers

Each register is selected using a combination of address and, for some, LCR register bit settings as shown in Table 19-29. The following registers are accessible by the local host (LH) at address = module base address + address offset. The module base address is the module start address. Note that register address offsets depends upon the module address alignment at the system top level.

**Table 19-29. UART Registers**

| Address Offset | Registers              |                        |                        |                        |                          |                          |
|----------------|------------------------|------------------------|------------------------|------------------------|--------------------------|--------------------------|
|                | LCR[7] = 0             |                        | LCR[7:0] ≠ BFh         |                        | LCR[7:0] = BFh           |                          |
|                | Read                   | Write                  | Read                   | Write                  | Read                     | Write                    |
| 0h             | RHR                    | THR                    | DLL                    | DLL                    | DLL                      | DLL                      |
| 4h             | IER <sup>(1)</sup>     | IER <sup>(1)</sup>     | DLH                    | DLH                    | DLH                      | DLH                      |
| 8h             | IIR                    | FCR <sup>(2)</sup>     | IIR                    | FCR <sup>(2)</sup>     | EFR                      | EFR                      |
| Ch             | LCR                    | LCR                    | LCR                    | LCR                    | LCR                      | LCR                      |
| 10h            | MCR <sup>(2)</sup>     | MCR <sup>(2)</sup>     | MCR <sup>(2)</sup>     | MCR <sup>(2)</sup>     | XON1/ADDR1               | XON1/ADDR1               |
| 14h            | LSR                    | -                      | LSR                    | -                      | XON2/ADDR2               | XON2/ADDR2               |
| 18h            | MSR/TCR <sup>(3)</sup> | TCR <sup>(3)</sup>     | MSR/TCR <sup>(3)</sup> | TCR <sup>(3)</sup>     | XOFF1/TCR <sup>(3)</sup> | XOFF1/TCR <sup>(3)</sup> |
| 1Ch            | SPR/TLR <sup>(3)</sup> | SPR/TLR <sup>(3)</sup> | SPR/TLR <sup>(3)</sup> | SPR/TLR <sup>(3)</sup> | XOFF2/TLR <sup>(3)</sup> | XOFF2/TLR <sup>(3)</sup> |
| 20h            | MDR1                   | MDR1                   | MDR1                   | MDR1                   | MDR1                     | MDR1                     |
| 24h            | MDR2                   | MDR2                   | MDR2                   | MDR2                   | MDR2                     | MDR2                     |
| 28h            | SFLSR                  | TXFLL                  | SFLSR                  | TXFLL                  | SFLSR                    | TXFLL                    |
| 2Ch            | RESUME                 | TXFLH                  | RESUME                 | TXFLH                  | RESUME                   | TXFLH                    |
| 30h            | SFREGL                 | RXFLL                  | SFREGL                 | RXFLL                  | SFREGL                   | RXFLL                    |
| 34h            | SFREGH                 | RXFLH                  | SFREGH                 | RXFLH                  | SFREGH                   | RXFLH                    |
| 38h            | BLR                    | BLR                    | UASR                   | -                      | UASR                     | -                        |
| 3Ch            | ACREG                  | ACREG                  | -                      | -                      | -                        | -                        |
| 40h            | SCR                    | SCR                    | SCR                    | SCR                    | SCR                      | SCR                      |
| 44h            | SSR                    | SSR[2]                 | SSR                    | SSR[2]                 | SSR                      | SSR[2]                   |
| 48h            | EBLR                   | EBLR                   | -                      | -                      | -                        | -                        |
| 50h            | MVR                    | -                      | MVR                    | -                      | MVR                      | -                        |
| 54h            | SYSC                   | SYSC                   | SYSC                   | SYSC                   | SYSC                     | SYSC                     |
| 58h            | SYSS                   | -                      | SYSS                   | -                      | SYSS                     | -                        |
| 5Ch            | WER                    | WER                    | WER                    | WER                    | WER                      | WER                      |
| 60h            | CFPS                   | CFPS                   | CFPS                   | CFPS                   | CFPS                     | CFPS                     |
| 64h            | RXFIFO_LVL             | RXFIFO_LVL             | RXFIFO_LVL             | RXFIFO_LVL             | RXFIFO_LVL               | RXFIFO_LVL               |
| 68h            | TXFIFO_LVL             | TXFIFO_LVL             | TXFIFO_LVL             | TXFIFO_LVL             | TXFIFO_LVL               | TXFIFO_LVL               |
| 6Ch            | IER2                   | IER2                   | IER2                   | IER2                   | IER2                     | IER2                     |
| 70h            | ISR2                   | ISR2                   | ISR2                   | ISR2                   | ISR2                     | ISR2                     |
| 74h            | FREQ_SEL               | FREQ_SEL               | FREQ_SEL               | FREQ_SEL               | FREQ_SEL                 | FREQ_SEL                 |
| 78h            | -                      | -                      | -                      | -                      | -                        | -                        |
| 7Ch            | -                      | -                      | -                      | -                      | -                        | -                        |
| 80h            | MDR3                   | MDR3                   | MDR3                   | MDR3                   | MDR3                     | MDR3                     |
| 84h            | TXDMA                  | TXDMA                  | TXDMA                  | TXDMA                  | TXDMA                    | TXDMA                    |

<sup>(1)</sup> In UART modes, IER[7:4] can only be written when EFR[4] = 1; in IrDA/CIR modes, EFR[4] has no impact on the access to IER[7:4].

<sup>(2)</sup> MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.

<sup>(3)</sup> Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.



### 19.5.1.1 Receiver Holding Register (RHR)

The receiver section consists of the receiver holding register and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character.

---

**NOTE:** If an overflow occurs, the data in the RHR is not overwritten.

---

The receiver holding register (RHR) register is shown in [Figure 19-34](#) and described in [Table 19-30](#).

**Figure 19-34. Receiver Holding Register (RHR)**

| 15       | 8   | 7   | 0         |
|----------|-----|-----|-----------|
| Reserved |     | RHR |           |
|          | R-0 |     | R-unknown |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-30. Receiver Holding Register (RHR) Field Descriptions**

| Bit  | Field    | Value | Description               |
|------|----------|-------|---------------------------|
| 15-8 | Reserved | 0     | Reserved.                 |
| 7-0  | RHR      | 0-FFh | Receive holding register. |

### 19.5.1.2 Transmit Holding Register (THR)

The transmitter section consists of the transmit holding register and the transmit shift register. The transmit holding register is a 64-byte FIFO. The MPU writes data to the THR. The data is placed in the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location zero of the FIFO is used to store the data.

The transmit holding register (THR) is shown in [Figure 19-35](#) and described in [Table 19-31](#).

**Figure 19-35. Transmit Holding Register (THR)**

| 15       | 8   | 7   | 0         |
|----------|-----|-----|-----------|
| Reserved |     | THR |           |
|          | R-0 |     | W-unknown |

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

**Table 19-31. Transmit Holding Register (THR) Field Descriptions**

| Bit  | Field    | Value | Description                |
|------|----------|-------|----------------------------|
| 15-8 | Reserved | 0     | Reserved.                  |
| 7-0  | THR      | 0-FFh | Transmit holding register. |

### 19.5.1.3 Interrupt Enable Register (IER) - UART Mode

The interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are seven types of interrupt in this mode: receiver error, RHR interrupt, THR interrupt, XOFF received and CTS/RTS change of state from low to high. Each interrupt can be enabled/disabled individually. There is also a sleep mode enable bit. The UART interrupt enable register (IER) is shown in [Figure 19-36](#) and described in [Table 19-32](#).

**Figure 19-36. UART Interrupt Enable Register (IER)**

| 15    | Reserved |        |           |            |           |       |       |  | 8 |
|-------|----------|--------|-----------|------------|-----------|-------|-------|--|---|
| R-0   |          |        |           |            |           |       |       |  |   |
| 7     | 6        | 5      | 4         | 3          | 2         | 1     | 0     |  |   |
| CTSIT | RTSIT    | XOFFIT | SLEEPMODE | MODEMSTSIT | LINESTSIT | THRIT | RHRIT |  |   |
| R/W-0 | R/W-0    | R/W-0  | R/W-0     | R/W-0      | R/W-0     | R/W-0 | R/W-0 |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-32. UART Interrupt Enable Register (IER) Field Descriptions**

| Bit  | Field      | Value | Description  |
|------|------------|-------|--|
| 15-8 | Reserved   | 0     | Reserved.  |
| 7    | CTSIT      | 0     | Can be written only when EFR[4] = 1.                                   |
|      |            |       | Disables the CTS interrupt.  |
|      |            |       | Enables the CTS interrupt.   |
| 6    | RTSIT      | 0     | Can be written only when EFR[4] = 1.                                   |
|      |            |       | Disables the RTS interrupt.  |
|      |            |       | Enables the RTS interrupt.   |
| 5    | XOFFIT     | 0     | Can be written only when EFR[4] = 1.                                   |
|      |            |       | Disables the XOFF interrupt.   |
|      |            |       | Enables the XOFF interrupt.  |
| 4    | SLEEPMODE  | 0     | Can be only written when EFR[4] = 1.                                   |
|      |            |       | Disables sleep mode.   |
|      |            |       | Enables sleep mode (stop baud rate clock when the module is inactive). |
| 3    | MODEMSTSIT | 0     | Disables the modem status register interrupt.                          |
|      |            |       | Enables the modem status register interrupt                            |
| 2    | LINESTSIT  | 0     | Disables the receiver line status interrupt.                           |
|      |            |       | Enables the receiver line status interrupt.                            |
| 1    | THRIT      | 0     | Disables the THR interrupt.  |
|      |            |       | Enables the THR interrupt.   |
| 0    | RHRIT      | 0     | Disables the RHR interrupt and time out interrupt.                     |
|      |            |       | Enables the RHR interrupt and time out interrupt.                      |

#### 19.5.1.4 Interrupt Enable Register (IER) - IrDA Mode

The IrDA interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 8 types of interrupt in these modes, received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. The IrDA interrupt enable register (IER) is shown in [Figure 19-37](#) and described in [Table 19-33](#).

---

**NOTE:** The TXSTATUSIT interrupt reflects two possible conditions. The MDR2[0] bit should be read to determine the status in the event of this interrupt.

---

**Figure 19-37. IrDA Interrupt Enable Register (IER)**

| 15    | Reserved |            |              |             |              |       |       |       | 8   |
|-------|----------|------------|--------------|-------------|--------------|-------|-------|-------|-----|
|       |          |            |              |             |              |       |       |       | R-0 |
| 7     | 6        | 5          | 4            | 3           | 2            | 1     | 0     |       |     |
| EOFIT | LINESIT  | TXSTATUSIT | STSIFOTRIGIT | RXOVERRUNIT | LASTRXBYTEIT | THRIT | RHRIT |       |     |
| R/W-0 | R/W-0    | R/W-0      | R/W-0        | R/W-0       | R/W-0        | R/W-0 | R/W-0 | R/W-0 |     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-33. IrDA Interrupt Enable Register (IER) Field Descriptions**

| Bit  | Field        | Value | Description   |
|------|--------------|-------|---|
| 15-8 | Reserved     | 0     | Reserved.   |
| 7    | EOFIT        | 0     | Disables the received EOF interrupt.                  |
|      |              | 1     | Enables the received EOF interrupt.                   |
| 6    | LINESIT      | 0     | Disables the receiver line status interrupt.          |
|      |              | 1     | Enables the receiver line status interrupt.           |
| 5    | TXSTATUSIT   | 0     | Disables the TX status interrupt.                     |
|      |              | 1     | Enables the TX status interrupt.                      |
| 4    | STSIFOTRIGIT | 0     | Disables status FIFO trigger level interrupt.         |
|      |              | 1     | Enables status FIFO trigger level interrupt.          |
| 3    | RXOVERRUNIT  | 0     | Disables the RX overrun interrupt.                    |
|      |              | 1     | Enables the RX overrun interrupt.                     |
| 2    | LASTRXBYTEIT | 0     | Disables the last byte of frame in RX FIFO interrupt. |
|      |              | 1     | Enables the last byte of frame in RX FIFO interrupt.  |
| 1    | THRIT        | 0     | Disables the THR interrupt.                           |
|      |              | 1     | Enables the THR interrupt.                            |
| 0    | RHRIT        | 0     | Disables the RHR interrupt.                           |
|      |              | 1     | Enables the RHR interrupt.                            |

### 19.5.1.5 Interrupt Enable Register (IER) - CIR Mode

The CIR interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 5 types of interrupt in these modes, TX status, RX overrun, RX stop interrupt, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. The CIR interrupt enable register (IER) is shown in [Figure 19-38](#) and described in [Table 19-34](#).

---

**NOTE:** In CIR mode, the TXSTATUSIT bit has only one meaning corresponding to the case MDR2[0] = 0.

The RXSTOPIT interrupt is generated based on the value set in the BOF Length register (EBLR).

---

**Figure 19-38. CIR Interrupt Enable Register (IER)**

|                 |                     |                 |   |                      |                   |                |                |  |   |
|-----------------|---------------------|-----------------|---|----------------------|-------------------|----------------|----------------|--|---|
| 15              | Reserved            |                 |   |                      |                   |                |                |  | 8 |
| R-0             |                     |                 |   |                      |                   |                |                |  |   |
| 7               | 6                   | 5               | 4 | 3                    | 2                 | 1              | 0              |  |   |
| Reserved<br>R-0 | TXSTATUSIT<br>R/W-0 | Reserved<br>R-0 |   | RXOVERRUNIT<br>R/W-0 | RXSTOPIT<br>R/W-0 | THRIT<br>R/W-0 | RHRIT<br>R/W-0 |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-34. CIR Interrupt Enable Register (IER) Field Descriptions**

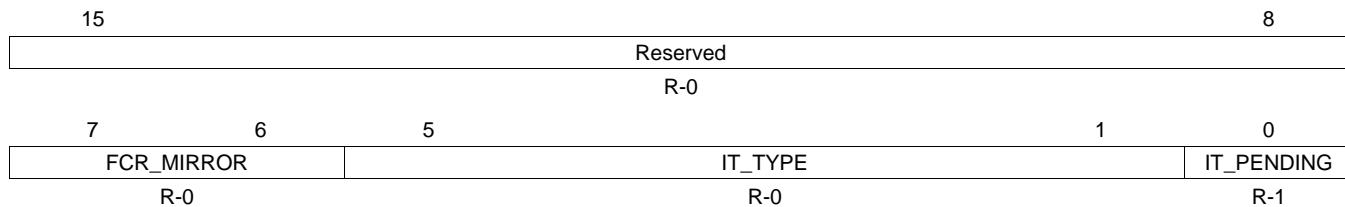
| Bit  | Field       | Value | Description                        |
|------|-------------|-------|------------------------------------|
| 15-6 | Reserved    | 0     | Reserved.                          |
| 5    | TXSTATUSIT  | 0     | Disables the TX status interrupt.  |
|      |             | 1     | Enables the TX status interrupt.   |
| 4    | Reserved    | 0     | Reserved.                          |
| 3    | RXOVERRUNIT | 0     | Disables the RX overrun interrupt. |
|      |             | 1     | Enables the RX overrun interrupt.  |
| 2    | RXSTOPIT    | 0     | Disables the RX stop interrupt.    |
|      |             | 1     | Enables the RX stop interrupt.     |
| 1    | THRIT       | 0     | Disables the THR interrupt.        |
|      |             | 1     | Enables the THR interrupt.         |
| 0    | RHRIT       | 0     | Disables the RHR interrupt.        |
|      |             | 1     | Enables the RHR interrupt.         |

### 19.5.1.6 Interrupt Identification Register (IIR) - UART Mode

The UART interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The UART interrupt identification register (IIR) is shown in [Figure 19-39](#) and described in [Table 19-35](#).

**NOTE:** An interrupt source can be flagged only if enabled in the IER register.

**Figure 19-39. UART Interrupt Identification Register (IIR)**



LEGEND: R = Read only; -n = value after reset

**Table 19-35. UART Interrupt Identification Register (IIR) Field Descriptions**

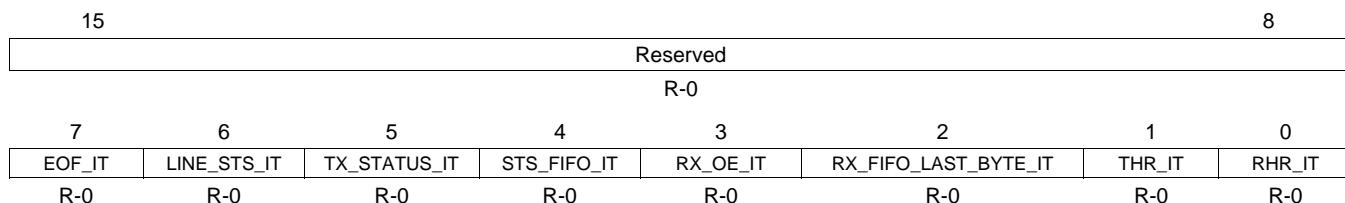
| Bit  | Field      | Value | Description  |
|------|------------|-------|--|
| 15-8 | Reserved   | 0     | Reserved.  |
| 7-6  | FCR_MIRROR | 0-3h  | Mirror the contents of FCR[0] on both bits.  |
| 5-1  | IT_TYPE    | 0-1Fh | Seven possible interrupts in UART mode; other combinations never occur:<br>0 Modem interrupt. Priority = 4.<br>1h THR interrupt. Priority = 3.<br>2h RHR interrupt. Priority = 2.<br>3h Receiver line status error. Priority = 1.<br>4h-5h Reserved<br>6h Rx timeout. Priority = 2.<br>7h Reserved<br>8h Xoff/special character. Priority = 5.<br>9h-Fh Reserved<br>10h CTS, RTS, DSR change state from active (low) to inactive (high). Priority = 6.<br>11h-1Fh Reserved |
| 0    | IT_PENDING | 0     | Interrupt pending.<br>An interrupt is pending.   |
|      |            | 1     | No interrupt is pending.   |

### 19.5.1.7 Interrupt Identification Register (IIR) - IrDA Mode

The IrDA interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The IrDA interrupt identification register (IIR) is shown in [Figure 19-40](#) and described in [Table 19-36](#).

**NOTE:** An interrupt source can be flagged only if enabled in the IER register.

**Figure 19-40. IrDA Interrupt Identification Register (IIR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-36. IrDA Interrupt Identification Register (IIR) Field Descriptions**

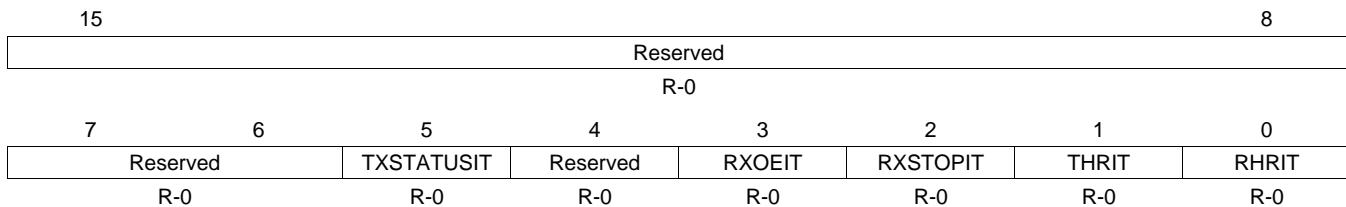
| Bit  | Field                | Value  | Description  |
|------|----------------------|--------|--|
| 15-8 | Reserved             | 0      | Reserved.  |
| 7    | EOF_IT               | 0<br>1 | Received EOF interrupt inactive.<br>Received EOF interrupt active.                                   |
| 6    | LINE_STS_IT          | 0<br>1 | Receiver line status interrupt inactive.<br>Receiver line status interrupt active.                   |
| 5    | TX_STATUS_IT         | 0<br>1 | TX status interrupt inactive.<br>TX status interrupt active.   |
| 4    | STS_FIFO_IT          | 0<br>1 | Status FIFO trigger level interrupt inactive.<br>Status FIFO trigger level interrupt active.         |
| 3    | RX_OE_IT             | 0<br>1 | RX overrun interrupt inactive.<br>RX overrun interrupt active.                                       |
| 2    | RX_FIFO_LAST_BYTE_IT | 0<br>1 | Last byte of frame in RX FIFO interrupt inactive.<br>Last byte of frame in RX FIFO interrupt active. |
| 1    | THR_IT               | 0<br>1 | THR interrupt inactive.<br>THR interrupt active.   |
| 0    | RHR_IT               | 0<br>1 | RHR interrupt inactive.<br>RHR interrupt active.   |

### 19.5.1.8 Interrupt Identification Register (IIR) - CIR Mode

The CIR interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. The CIR interrupt identification register (IIR) is shown in [Figure 19-41](#) and described in [Table 19-37](#).

**NOTE:** An interrupt source can be flagged only if enabled in the IER register.

**Figure 19-41. CIR Interrupt Identification Register (IIR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

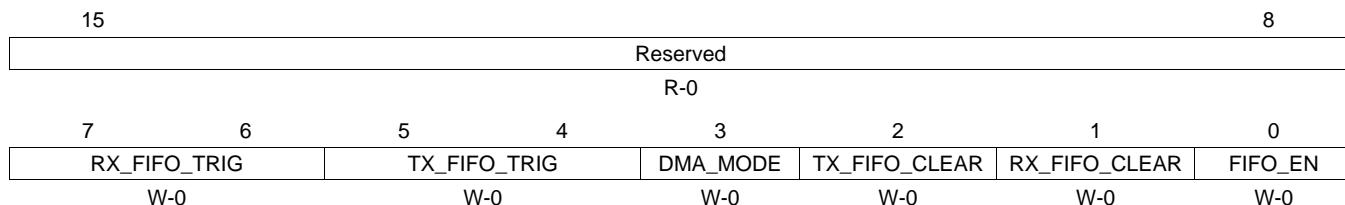
**Table 19-37. CIR Interrupt Identification Register (IIR) Field Descriptions**

| Bit  | Field      | Value  | Description  |
|------|------------|--------|--|
| 15-6 | Reserved   | 0      | Reserved.  |
| 5    | TXSTATUSIT | 0<br>1 | TX status interrupt inactive<br>TX status interrupt active             |
| 4    | Reserved   | 0      | Reserved.  |
| 3    | RXOEIT     | 0<br>1 | RX overrun interrupt inactive<br>RX overrun interrupt active           |
| 2    | RXSTOPIT   | 0<br>1 | Receive stop interrupt is inactive<br>Receive stop interrupt is active |
| 1    | THRIT      | 0<br>1 | THR interrupt inactive<br>THR interrupt active                         |
| 0    | RHRIT      | 0<br>1 | RHR interrupt inactive<br>RHR interrupt active                         |

### 19.5.1.9 FIFO Control Register (FCR)

The FIFO Control Register (FCR) is shown in [Figure 19-42](#) and described in [Table 19-38](#).

**Figure 19-42. FIFO Control Register (FCR)**



LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 19-38. FIFO Control Register (FCR) Field Descriptions**

| Bit  | Field         | Value | Description   |
|------|---------------|-------|---|
| 15-8 | Reserved      | 0     | Reserved.   |
| 7-6  | RX_FIFO_TRIG  | 0-3h  | Sets the trigger level for the RX FIFO:<br><br>If SCR[7] = 0 and TLR[7:4] ≠ 0000, RX_FIFO_TRIG is not considered.<br>If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with the granularity 1.<br>If SCR[7] = 0 and TLR[7:4] = 0000:<br><br>0 8 characters<br>1h 16 characters<br>2h 56 characters<br>3h 60 characters  |
| 5-4  | TX_FIFO_TRIG  | 0-3h  | Can be written only if EFR[4] = 1.<br><br>Sets the trigger level for the TX FIFO:<br><br>If SCR[6] = 0 and TLR[3:0] ≠ 0000, TX_FIFO_TRIG is not considered.<br>If SCR[6] = 1, TX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with a granularity of 1.<br>If SCR[6] = 0 and TLR[3:0] = 0000:<br><br>0 8 characters<br>1h 16 characters<br>2h 32 characters<br>3h 56 characters |
| 3    | DMA_MODE      |       | Can be changed only when the baud clock is not running (DLL and DLH cleared to 0).<br><br>If SCR[0] = 0, this register is considered.<br><br>0 DMA_MODE 0 (No DMA).<br>1 DMA_MODE 1 (UART_NDMA_REQ[0] in TX, UART_NDMA_REQ[1] in RX).   |
| 2    | TX_FIFO_CLEAR | 0     | No change.  |
|      |               | 1     | Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.   |
| 1    | RX_FIFO_CLEAR | 0     | No change.  |
|      |               | 1     | Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.  |
| 0    | FIFO_EN       |       | Can be changed only when the baud clock is not running (DLL and DLH cleared to 0).<br><br>0 Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs.<br>1 Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.  |

### 19.5.1.10 Line Control Register (LCR)

The line control register (LCR) is shown in [Figure 19-43](#) and described in [Table 19-39](#).

---

**NOTE:** As soon as LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1.

---

**Figure 19-43. Line Control Register (LCR)**

| Line Control Register (LCR) |          |              |              |           |         |             |  |   |
|-----------------------------|----------|--------------|--------------|-----------|---------|-------------|--|---|
| 15                          | Reserved |              |              |           |         |             |  | 8 |
| R-0                         |          |              |              |           |         |             |  |   |
| DIV_EN                      | BREAK_EN | PARITY_TYPE2 | PARITY_TYPE1 | PARITY_EN | NB_STOP | CHAR_LENGTH |  |   |
| R/W-0                       | R/W-0    | R/W-0        | R/W-0        | R/W-0     | R/W-0   | R/W-0       |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-39. Line Control Register (LCR) Field Descriptions**

| Bit  | Field        | Value | Description   |
|------|--------------|-------|---|
| 15-8 | Reserved     | 0     | Reserved.   |
| 7    | DIV_EN       | 0     | Divisor latch enable.   |
|      |              |       | Normal operating condition.   |
|      |              |       | Divisor latch enable. Allows access to DLL and DLH.   |
| 6    | BREAK_EN     | 0     | Break control bit.  |
|      |              |       | <b>Note:</b> When LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1. |
|      |              |       | Normal operating condition.   |
| 5    | PARITY_TYPE2 | 0     | If LCR[3] = 1:<br>If LCR[5] = 0, LCR[4] selects the forced parity format.   |
|      |              |       | If LCR[5] = 1 and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data.                 |
|      |              |       | If LCR[5] = 1 and LCR[4] = 1, the parity bit is forced to 0 in the transmitted and received data.                 |
| 4    | PARITY_TYPE1 | 0     | If LCR[3] = 1:<br>Odd parity is generated.  |
|      |              |       | Even parity is generated.   |
|      |              |       |   |
| 3    | PARITY_EN    | 0     | Parity bit.<br>No parity.   |
|      |              |       | A parity bit is generated during transmission, and the receiver checks for received parity.                       |
|      |              |       |   |
| 2    | NB_STOP      | 0     | Specifies the number of stop bits.<br>1 stop bit (word length = 5, 6, 7, 8).                                      |
|      |              |       | 1.5 stop bits (word length = 5) or 2 stop bits (word length = 6, 7, 8).   |
|      |              |       |   |
| 1-0  | CHAR_LENGTH  | 0-3h  | Specifies the word length to be transmitted or received.  |
|      |              | 0     | 5 bits  |
|      |              | 1h    | 6 bits  |
|      |              | 2h    | 7 bits  |
|      |              | 3h    | 8 bit   |

### 19.5.1.11 Modem Control Register (MCR)

Bits 3-0 control the interface with the modem, data set, or peripheral device that is emulating the modem. The modem control register (MCR) is shown in [Figure 19-44](#) and described in [Table 19-40](#).

**Figure 19-44. Modem Control Register (MCR)**

|          |          |       |            |         |         |     |     |  |   |
|----------|----------|-------|------------|---------|---------|-----|-----|--|---|
| 15       | Reserved |       |            |         |         |     |     |  | 8 |
| R-0      |          |       |            |         |         |     |     |  |   |
| 7        | 6        | 5     | 4          | 3       | 2       | 1   | 0   |  |   |
| Reserved | TCRTLR   | XONEN | LOOPBACKEN | CDSTSCH | RISTSCH | RTS | DTR |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-40. Modem Control Register (MCR) Field Descriptions**

| Bit  | Field      | Value | Description  |
|------|------------|-------|--|
| 15-7 | Reserved   | 0     | Reserved.  |
| 6    | TCRTLR     | 0     | Can be written only when EFR[4] = 1.   |
|      |            |       | No action.   |
|      |            |       | Enables access to the TCR and TLR registers.   |
| 5    | XONEN      | 0     | Can be written only when EFR[4] = 1.   |
|      |            |       | Disable XON any function.  |
|      |            |       | Enable XON any function.   |
| 4    | LOOPBACKEN | 0     | Loopback mode enable.  |
|      |            |       | Normal operating mode.   |
|      |            |       | Enable local loopback mode (internal). In this mode, the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally. |
| 3    | CDSTSCH    | 0     | In loopback mode, forces <b>DCD</b> input high and IRQ outputs to INACTIVE state.  |
|      |            |       | In loopback mode, forces <b>DCD</b> input low and IRQ outputs to INACTIVE state.   |
| 2    | RISTSCH    | 0     | In loopback mode, forces <b>RI</b> input inactive (high).  |
|      |            |       | In loopback mode, forces <b>RI</b> input active (low).   |
| 1    | RTS        | 0     | In loopback mode, controls MSR[4]. If auto-RTS is enabled, the <b>RTS</b> output is controlled by hardware flow control.   |
|      |            |       | Force <b>RTS</b> output to inactive (high).  |
|      |            |       | Force <b>RTS</b> output to active (low).   |
| 0    | DTR        | 0     | Force <b>DTR</b> output (used in loopback mode) to inactive (high).  |
|      |            |       | Force <b>DTR</b> output (used in loopback mode) to active (low).   |

### 19.5.1.12 Line Status Register (LSR) - UART Mode

When the UART line status register (LSR) is read, LSR[4:2] reflect the error bits (BI, FE, PE) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies errors in a character. Reading RHR updates BI, FE, and PE. LSR [7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the RX FIFO. The UART line status register (LSR) is shown in [Figure 19-45](#) and described in [Table 19-41](#).

**NOTE:** Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR. Reading LSR clears OE if set.

**Figure 19-45. UART Line Status Register (LSR)**

| 15        | Reserved |         |      |      |      |      |         |  | 8 |
|-----------|----------|---------|------|------|------|------|---------|--|---|
| R-0       |          |         |      |      |      |      |         |  |   |
| 7         | 6        | 5       | 4    | 3    | 2    | 1    | 0       |  |   |
| RXFIFOSTS | TXSRE    | TXFIFOE | RXBI | RXFE | RXPE | RXOE | RXFIFOE |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

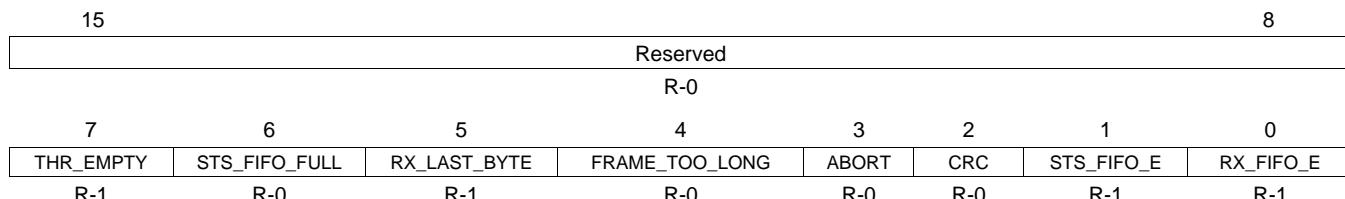
**Table 19-41. UART Line Status Register (LSR) Field Descriptions**

| Bit  | Field     | Value | Description   |
|------|-----------|-------|---|
| 15-8 | Reserved  | 0     | Reserved.   |
| 7    | RXFIFOSTS | 0     | Normal operation.   |
|      |           | 1     | At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no errors are present in the RX FIFO.                             |
| 6    | TXSRE     | 0     | Transmitter hold (TX FIFO) and shift registers are not empty.   |
|      |           | 1     | Transmitter hold (TX FIFO) and shift registers are empty.   |
| 5    | TXFIFOE   | 0     | Transmit hold register (TX FIFO) is not empty.  |
|      |           | 1     | Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.   |
| 4    | RXBI      | 0     | No break condition.   |
|      |           | 1     | A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character + 1 bit time frame).                           |
| 3    | RXFE      | 0     | No framing error in data being read from RX FIFO.   |
|      |           | 1     | Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).   |
| 2    | RXPE      | 0     | No parity error in data being read from RX FIFO.  |
|      |           | 1     | Parity error in data being read from RX FIFO.   |
| 1    | RXOE      | 0     | No overrun error.   |
|      |           | 1     | Overrun error occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full. |
| 0    | RXFIFOE   | 0     | No data in the receive FIFO.  |
|      |           | 1     | At least one data character in the RX FIFO.   |

### 19.5.1.13 Line Status Register (LSR) - IrDA Mode

When the IrDA line status register (LSR) is read, LSR[4:2] reflect the error bits (FL, CRC, ABORT) of the frame at the top of the status FIFO (next frame status to be read). The IrDA line status register (LSR) is shown in [Figure 19-46](#) and described in [Table 19-42](#).

**Figure 19-46. IrDA Line Status Register (LSR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

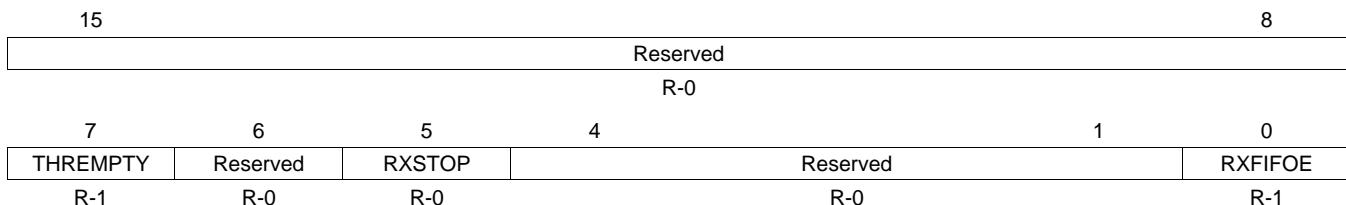
**Table 19-42. IrDA Line Status Register (LSR) Field Descriptions**

| Bit  | Field          | Value | Description   |
|------|----------------|-------|---|
| 15-8 | Reserved       | 0     | Reserved.   |
| 7    | THR_EMPTY      | 0     | Transmit holding register (TX FIFO) is not empty.   |
|      |                | 1     | Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.   |
| 6    | STS_FIFO_FULL  | 0     | Status FIFO is not full.  |
|      |                | 1     | Status FIFO is full.  |
| 5    | RX_LAST_BYTE   | 0     | The RX FIFO (RHR) does not contain the last byte of the frame to be read.   |
|      |                | 1     | The RX FIFO (RHR) contains the last byte of the frame to be read. This bit is set to 1 only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.  |
| 4    | FRAME_TOO_LONG | 0     | No frame-too-long error in frame.   |
|      |                | 1     | Frame-too-long error in the frame at the top of the status FIFO (next character to be read). This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) is received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected. |
| 3    | ABORT          | 0     | No abort pattern error in frame.  |
|      |                | 1     | Abort pattern received. SIR and MIR: abort pattern. FIR: Illegal symbol.  |
| 2    | CRC            | 0     | No CRC error in frame.  |
|      |                | 1     | CRC error in the frame at the top of the status FIFO (next character to be read).   |
| 1    | STS_FIFO_E     | 0     | Status FIFO is not empty.   |
|      |                | 1     | Status FIFO is empty.   |
| 0    | RX_FIFO_E      | 0     | At least one data character in the RX FIFO.   |
|      |                | 1     | No data in the receive FIFO.  |

### 19.5.1.14 Line Status Register (LSR) - CIR Mode

The CIR line status register (LSR) is shown in [Figure 19-47](#) and described in [Table 19-43](#).

**Figure 19-47. CIR Line Status Register (LSR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

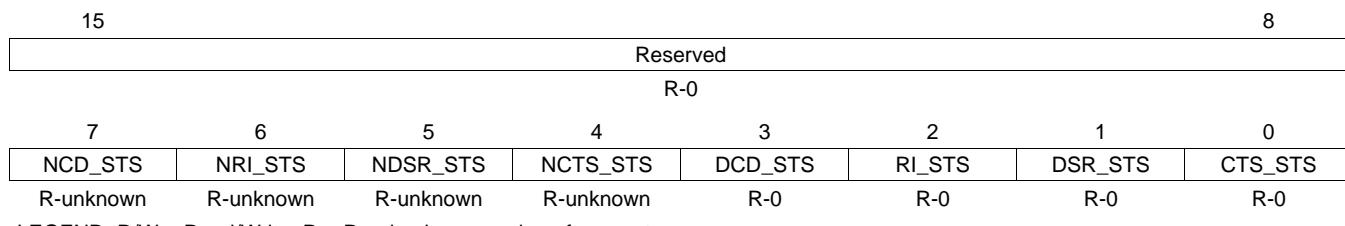
**Table 19-43. CIR Line Status Register (LSR) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 15-8 | Reserved  | 0     | Reserved.  |
| 7    | THREEMPTY | 0     | Transmit holding register (TX FIFO) is not empty.  |
|      |           | 1     | Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.  |
| 6    | Reserved  | 0     | Reserved.  |
| 5    | RXSTOP    | 0     | The RXSTOP is generated based on the value set in the BOF Length register (EBLR).<br>Reception is on going or waiting for a new frame. |
|      |           | 1     | Reception is completed. It is cleared on a single read of the LSR register.  |
| 4-1  | Reserved  | 0     | Reserved   |
| 0    | RXFIFOE   | 0     | At least one data character in the RX FIFO.  |
|      |           | 1     | No data in the receive FIFO.   |

### 19.5.1.15 Modem Status Register (MSR)

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the Local Host. It also indicates when a control input from the modem changes state. The modem status register (MSR) is shown in [Figure 19-48](#) and described in [Table 19-44](#).

**Figure 19-48. Modem Status Register (MSR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-44. Modem Status Register (MSR) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved.  |
| 7    | NCD_STS  |       | This bit is the complement of the $\overline{DCD}$ input. In loopback mode, it is equivalent to MCR[3].              |
| 6    | NRI_STS  |       | This bit is the complement of the $\overline{RI}$ input. In loopback mode, it is equivalent to MCR[2].               |
| 5    | NDSR_STS |       | This bit is the complement of the $\overline{DSR}$ input. In loopback mode, it is equivalent to MCR[0].              |
| 4    | NCTS_STS |       | This bit is the complement of the $\overline{CTS}$ input. In loopback mode, it is equivalent to MCR[1].              |
| 3    | DCD_STS  | 0     | No change.   |
|      |          | 1     | Indicates that $\overline{DCD}$ input (or MCR[3] in loopback mode) has changed. Cleared on a read.                   |
| 2    | RI_STS   | 0     | No change.   |
|      |          | 1     | Indicates that $\overline{RI}$ input (or MCR[2] in loopback mode) changed state from low to high. Cleared on a read. |
| 1    | DSR_STS  | 0     | No change.   |
|      |          | 1     | Indicates that $\overline{DSR}$ input (or MCR[0] in loopback mode) changed state. Cleared on a read.                 |
| 0    | CTS_STS  | 0     | No change.   |
|      |          | 1     | Indicates that $\overline{CTS}$ input (or MCR[1] in loopback mode) changed state. Cleared on a read.                 |

### 19.5.1.16 Transmission Control Register (TCR)

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware flow control. The transmission control register (TCR) is shown in [Figure 19-49](#) and described in [Table 19-45](#).

**NOTE:**

- Trigger levels from 0-60 bytes are available with a granularity of 4.
- Trigger level =  $4 \times [4\text{-bit register value}]$
- You must ensure that  $\text{TCR}[3:0] > \text{TCR}[7:4]$ , whenever auto-RTS or software flow control is enabled to avoid a misoperation of the device. In FIFO interrupt mode with flow control, you have to also ensure that the trigger level to HALT transmission is greater or equal to receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, FIFO operation stalls.
- In FIFO DMA mode with flow control, this concept does not exist because the DMA request is sent each time a byte is received.

**Figure 19-49. Transmission Control Register (TCR)**

| 15       | 8 | 7               | 4 | 3              | 0 |
|----------|---|-----------------|---|----------------|---|
| Reserved |   | RXFIFOTRIGSTART |   | RXFIFOTRIGHALT |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-45. Transmission Control Register (TCR) Field Descriptions**

| Bit  | Field           | Value | Description  |
|------|-----------------|-------|--|
| 15-8 | Reserved        | 0     | Reserved.  |
| 7-4  | RXFIFOTRIGSTART | 0-Fh  | RX FIFO trigger level to RESTORE transmission (0 to 60). |
| 3-0  | RXFIFOTRIGHALT  | 0-Fh  | RX FIFO trigger level to HALT transmission (0 to 60).    |

### 19.5.1.17 Scratchpad Register (SPR)

The scratchpad register (SPR) is a read/write register that does not control the module. It is a scratchpad register used to hold temporary data. The scratchpad register (SPR) is shown in [Figure 19-50](#) and described in [Table 19-46](#).

**Figure 19-50. Scratchpad Register (SPR)**

| 15       | 8 | 7        | 0 |
|----------|---|----------|---|
| Reserved |   | SPR_WORD |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-46. Scratchpad Register (SPR) Field Descriptions**

| Bit  | Field    | Value | Description          |
|------|----------|-------|----------------------|
| 15-8 | Reserved | 0     | Reserved.            |
| 7-0  | SPR_WORD | 0-FFh | Scratchpad register. |

### 19.5.1.18 Trigger Level Register (TLR)

This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation. The trigger level register (TLR) is shown in [Figure 19-51](#) and described in [Table 19-47](#).

**Figure 19-51. Trigger Level Register (TLR)**

| 15       | 8   | 7 | 4                | 3                | 0     |
|----------|-----|---|------------------|------------------|-------|
| Reserved |     |   | RX_FIFO_TRIG_DMA | TX_FIFO_TRIG_DMA |       |
|          | R-0 |   | R/W-0            |                  | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-47. Trigger Level Register (TLR) Field Descriptions**

| Bit  | Field            | Value | Description  |
|------|------------------|-------|--|
| 15-8 | Reserved         | 0     | Reserved.  |
| 7-4  | RX_FIFO_TRIG_DMA | 0-Fh  | Receive FIFO trigger level. See <a href="#">Table 19-48</a> .  |
| 3-0  | TX_FIFO_TRIG_DMA | 0-Fh  | Transmit FIFO trigger level. See <a href="#">Table 19-49</a> . |

**Table 19-48. RX FIFO Trigger Level Setting Summary**

| SCR[7] | TLR[7:4]  | Description   |
|--------|-----------|---|
| 0      | 0         | Defined by FCR[7:6] (either 8, 16, 56, 60 characters).  |
| 0      | ≠ 0000    | Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters).   |
| 1      | any value | Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters with a granularity of 1 character). <b>Note:</b> the combination of TLR[7:4] = 0000 and FCR[7:6] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior. |

**Table 19-49. TX FIFO Trigger Level Setting Summary**

| SCR[6] | TLR[3:0]  | Description   |
|--------|-----------|---|
| 0      | 0         | Defined by FCR[5:4] (either 8, 16, 32, 56 characters).  |
| 0      | ≠ 0000    | Defined by TLR[3:0] (from 4 to 60 characters with a granularity of 4 characters).   |
| 1      | any value | Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 characters with a granularity of 1 character). <b>Note:</b> the combination of TLR[3:0] = 0000 and FCR[5:4] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior. |

### 19.5.1.19 Mode Definition Register 1 (MDR1)

The mode of operation is programmed by writing to MDR1[2:0]; therefore, the mode definition register 1 (MDR1) must be programmed on startup after configuration of the configuration registers (DLL, DLH, and LCR). The value of MDR1[2:0] must not be changed again during normal operation. The mode definition register 1 (MDR1) is shown in [Figure 19-52](#) and described in [Table 19-50](#).

**NOTE:** If the module is disabled by setting the MODESELECT field to 7h, interrupt requests can still be generated unless disabled through the interrupt enable register (IER). In this case, UART mode interrupts are visible. Reading the interrupt identification register (IIR) shows the UART mode interrupt flags.

**Figure 19-52. Mode Definition Register 1 (MDR1)**

|       |              |         |       |         |         |            |   |
|-------|--------------|---------|-------|---------|---------|------------|---|
| 15    | Reserved     |         |       |         |         |            | 8 |
| R-0   |              |         |       |         |         |            |   |
| 7     | FRAMEENDMODE | SIPMODE | SCT   | SETTXIR | IRSLEEP | MODESELECT | 0 |
| R/W-0 | R/W-0        | R/W-0   | R/W-0 | R/W-0   | R/W-0   | R/W-7h     |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-50. Mode Definition Register 1 (MDR1) Field Descriptions**

| Bit  | Field        | Value  | Description   |
|------|--------------|--|---|
| 15-8 | Reserved     | 0  | Reserved.   |
| 7    | FRAMEENDMODE | 0<br>1   | IrDA mode only.<br>Frame-length method.<br>Set EOT bit method.  |
| 6    | SIPMODE      | 0<br>1   | MIR/FIR modes only.<br>Manual SIP mode: SIP is generated with the control of ACREG[3].<br>Automatic SIP mode: SIP is generated after each transmission.   |
| 5    | SCT          | 0<br>1   | Store and control the transmission.<br>Starts the infrared transmission when a value is written to the THR register.<br>Starts the infrared transmission with the control of ACREG[2]. <b>Note:</b> Before starting any transmission, there must be no reception ongoing. |
| 4    | SETTXIR      | 0<br>1   | Used to configure the infrared transceiver.<br>If MDR2[7] = 0, no action; if MDR2[7] = 1, TXIR pin output is forced low.<br>TXIR pin output is forced high (not dependant of MDR2[7] value).  |
| 3    | IRSLEEP      | 0<br>1   | IrDA/CIR sleep mode.<br>IrDA/CIR sleep mode disabled.<br>IrDA/CIR sleep mode enabled.   |
| 2-0  | MODESELECT   | 0-7h<br>0<br>1<br>2h<br>3h<br>4h<br>5h<br>6h<br>7h | UART/IrDA/CIR mode selection.<br>UART 16x mode.<br>SIR mode.<br>UART 16x auto-baud.<br>UART 13x mode.<br>MIR mode.<br>FIR mode.<br>CIR mode.<br>Disable (default state).  |

### 19.5.1.20 Mode Definition Register 2 (MDR2)

The MDR2[0] bit describes the status of the TX status interrupt in IIR[5]. The IRTXUNDERRUN bit must be read after a TX status interrupt occurs. The MDR2[2:1] bits set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0]. The mode definition register 2 (MDR2) is shown in [Figure 19-53](#) and described in [Table 19-51](#).

---

**NOTE:** The MDR2[6] bit gives the flexibility to invert the RX pin inside the UART module to ensure that the protocol at the input of the transceiver module has the same polarity at module level. By default, the RX pin is inverted because most of transceiver invert the IR receive pin.

---

**Figure 19-53. Mode Definition Register 2 (MDR2)**

|       |            |            |              |           |            |              |  |   |
|-------|------------|------------|--------------|-----------|------------|--------------|--|---|
| 15    | Reserved   |            |              |           |            |              |  | 8 |
| R-0   |            |            |              |           |            |              |  |   |
| 7     | SETTXIRALT | IRRXINVERT | CIRPULSEMODE | UARTPULSE | STSIFOTRIG | IRTXUNDERRUN |  | 0 |
| R/W-0 | R/W-0      | R/W-0      | R/W-0        | R/W-0     | R/W-0      | R-0          |  |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-51. Mode Definition Register 2 (MDR2) Field Descriptions**

| Bit  | Field        | Value                       | Description  |
|------|--------------|-----------------------------|--|
| 15-8 | Reserved     | 0                           | Reserved.  |
| 7    | SETTXIRALT   | 0<br>1                      | Provides alternate functionality for MDR1[4].<br>Normal mode<br>Alternate mode for SETTXIR   |
| 6    | IRRXINVERT   | 0<br>1                      | Only for IR mode (IrDA and CIR). Invert RX pin in the module before the voting or sampling system logic of the infrared block. This does not affect the RX path in UART modem modes.<br>Inversion is performed.<br>No inversion is performed.  |
| 5-4  | CIRPULSEMODE | 0-3h<br>0<br>1h<br>2h<br>3h | CIR pulse modulation definition. Defines high level of the pulse width associated with a digit:<br>Pulse width of 3 from 12 cycles.<br>Pulse width of 4 from 12 cycles.<br>Pulse width of 5 from 12 cycles.<br>Pulse width of 6 from 12 cycles.  |
| 3    | UARTPULSE    | 0<br>1                      | UART mode only. Used to allow pulse shaping in UART mode.<br>Normal UART mode.<br>UART mode with pulse shaping.  |
| 2-1  | STSIFOTRIG   | 0-3h<br>0<br>1h<br>2h<br>3h | Only for IrDA mode. Frame status FIFO threshold select:<br>1 entry<br>4 entries<br>7 entries<br>8 entries  |
| 0    | IRTXUNDERRUN | 0<br>1                      | IrDA transmission status interrupt. When the TX status interrupt (IIR[5]) occurs, the meaning of the interrupt is:<br>The last bit of the frame was transmitted successfully without error.<br>An underrun occurred. The last bit of the frame was transmitted but with an underrun error. The bit is reset to 0 when the RESUME register is read. |

### 19.5.1.21 Status FIFO Line Status Register (SFLSR)

Reading the status FIFO line status register (SFLSR) effectively reads frame status information from the status FIFO. This register does not physically exist. Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first). The status FIFO line status register (SFLSR) is shown in [Figure 19-54](#) and described in [Table 19-52](#).

---

**NOTE:** Top of RX FIFO = Next frame to be read from RX FIFO.

---

**Figure 19-54. Status FIFO Line Status Register (SFLSR)**

|          |          |          |                      |              |           |          |     |     |
|----------|----------|----------|----------------------|--------------|-----------|----------|-----|-----|
| 15       | Reserved |          |                      |              |           |          |     | 8   |
| R-0      |          |          |                      |              |           |          |     |     |
| 7        | 5        | 4        | 3                    | 2            | 1         | 0        |     |     |
| Reserved |          | OE_ERROR | FRAME_TOO_LONG_ERROR | ABORT_DETECT | CRC_ERROR | Reserved | R-0 | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-52. Status FIFO Line Status Register (SFLSR) Field Descriptions**

| Bit  | Field                | Value | Description   |
|------|----------------------|-------|---|
| 15-5 | Reserved             | 0     | Reserved.   |
| 4    | OE_ERROR             | 0     | No error  |
|      |                      | 1     | Overrun error in RX FIFO when frame at top of RX FIFO was received. |
| 3    | FRAME_TOO_LONG_ERROR | 0     | No error  |
|      |                      | 1     | Frame-length too long error in frame at top of RX FIFO.             |
| 2    | ABORT_DETECT         | 0     | No error  |
|      |                      | 1     | Abort pattern detected in frame at top of RX FIFO.                  |
| 1    | CRC_ERROR            | 0     | No error  |
|      |                      | 1     | CRC error in frame at top of RX FIFO.                               |
| 0    | Reserved             | 0     | Reserved.   |

### 19.5.1.22 RESUME Register

The RESUME register is used to clear internal flags, which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and always reads as 00. The RESUME register is shown in [Figure 19-55](#) and described in [Table 19-53](#).

**Figure 19-55. RESUME Register**

|          |   |   |        |
|----------|---|---|--------|
| 15       | 8 | 7 | 0      |
| Reserved |   |   | RESUME |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-53. RESUME Register Field Descriptions**

| Bit  | Field    | Value | Description                         |
|------|----------|-------|-------------------------------------|
| 15-8 | Reserved | 0     | Reserved.                           |
| 7-0  | RESUME   | 0-FFh | Dummy read to restart the TX or RX. |

### 19.5.1.23 Status FIFO Register Low (SFREGL)

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR. The status FIFO register low (SFREGL) is shown in Figure 19-56 and described in Table 19-54.

**Figure 19-56. Status FIFO Register Low (SFREGL)**

| 15       | 8 | 7         | 0 |
|----------|---|-----------|---|
| Reserved |   | SFREGL    |   |
| R-0      |   | R-unknown |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-54. Status FIFO Register Low (SFREGL) Field Descriptions**

| Bit  | Field    | Value | Description                   |
|------|----------|-------|-------------------------------|
| 15-8 | Reserved | 0     | Reserved.                     |
| 7-0  | SFREGL   | 0-FFh | LSB part of the frame length. |

### 19.5.1.24 Status FIFO Register High (SFREGH)

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR. The status FIFO register high (SFREGH) is shown in Figure 19-57 and described in Table 19-55.

**Figure 19-57. Status FIFO Register High (SFREGH)**

| 15       | 4 | 3         | 0 |
|----------|---|-----------|---|
| Reserved |   | SFREGH    |   |
| R-0      |   | R-unknown |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-55. Status FIFO Register High (SFREGH) Field Descriptions**

| Bit  | Field    | Value | Description                   |
|------|----------|-------|-------------------------------|
| 15-4 | Reserved | 0     | Reserved.                     |
| 3-0  | SFREGH   | 0-Fh  | MSB part of the frame length. |

### 19.5.1.25 BOF Control Register (BLR)

The BLR[6] bit is used to select whether C0h or FFh start patterns are to be used, when multiple start flags are required in SIR mode. If only one start flag is required, this is always C0h. If  $n$  start flags are required, either  $(n - 1)$  C0h or  $(n - 1)$  FFh flags are sent, followed by a single C0h flag (immediately preceding the first data byte). The BOF control register (BLR) is shown in [Figure 19-58](#) and described in [Table 19-56](#).

**Figure 19-58. BOF Control Register (BLR)**

| 15       | 8 | 7              | 6        | 5        | 0   |
|----------|---|----------------|----------|----------|-----|
| Reserved |   | STS FIFO RESET | XBOFTYPE | Reserved | R-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-56. BOF Control Register (BLR) Field Descriptions**

| Bit  | Field          | Value | Description                                    |
|------|----------------|-------|--|
| 15-8 | Reserved       | 0     | Reserved.                                      |
| 7    | STS FIFO RESET |       | Status FIFO reset. This bit is self-clearing.. |
| 6    | XBOFTYPE       | 0     | SIR xBOF select.<br>FFh start pattern is used. |
|      |                | 1     | C0h start pattern is used.                     |
| 0    | Reserved       | 0     | Reserved.                                      |

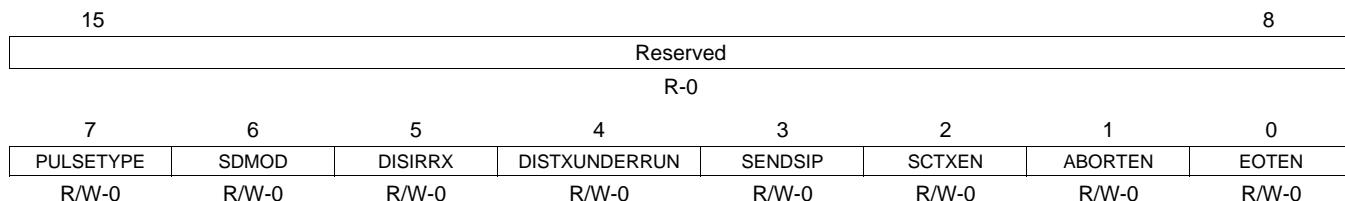
### 19.5.1.26 Auxiliary Control Register (ACREG)

The auxiliary control register (ACREG) is shown in [Figure 19-59](#) and described in [Table 19-57](#).

**NOTE:**

- If transmit FIFO is not empty and MDR1[5] = 1, IrDA starts a new transfer with data of previous frame as soon as abort frame has been sent. Therefore, TX FIFO must be reset before sending an abort frame.
- It is recommended to disable TX FIFO underrun capability by masking corresponding underrun interrupt. When disabling underrun by setting ACREG[4] = 1, unknown data is sent over TX line.

**Figure 19-59. Auxiliary Control Register (ACREG)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-57. Auxiliary Control Register (ACREG) Field Descriptions**

| Bit  | Field         | Value  | Description  |
|------|---------------|--------|--|
| 15-8 | Reserved      | 0      | Reserved.  |
| 7    | PULSETYPE     | 0<br>1 | SIR pulse-width select:<br>3/16 of baud-rate pulse width<br>1.6 $\mu$ s  |
| 6    | SDMOD         | 0<br>1 | Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers.<br>SD pin is set to high.<br>SD pin is set to low.   |
| 5    | DISIRRX       | 0<br>1 | Disable RX input.<br>Normal operation (RX input automatically disabled during transmit, but enabled outside of transmit operation).<br>Disables RX input (permanent state; independent of transmit).   |
| 4    | DISTXUNDERRUN | 0<br>1 | Disable TX underrun.<br>Long stop bits cannot be transmitted. TX underrun is enabled.<br>Long stop bits can be transmitted.  |
| 3    | SENDSIP       | 0<br>1 | MIR/FIR modes only. Send serial infrared interaction pulse (SIP).<br>If this bit is set during an MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission.<br>No action.<br>Send SIP pulse. |
| 2    | SCTXEN        |        | Store and control TX start. When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state-machine starts frame transmission. This bit is self-clearing.   |
| 1    | ABORTEN       |        | Frame abort. The LH can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame.   |
| 0    | EOTEN         |        | EOT (end-of-transmission) bit. The LH writes 1 to this bit just before it writes the last byte to the TX FIFO in the set-EOT bit frame-closing method. This bit is automatically cleared when the LH writes to the THR (TX FIFO).                                |

### 19.5.1.27 Supplementary Control Register (SCR)

The supplementary control register (SCR) is shown in [Figure 19-60](#) and described in [Table 19-58](#).

**NOTE:** Bit 4 enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

**Figure 19-60. Supplementary Control Register (SCR)**

| 15    | Reserved |       |       |       |       |       |       |          | 8          |
|-------|----------|-------|-------|-------|-------|-------|-------|----------|------------|
| R-0   |          |       |       |       |       |       |       |          |            |
| 7     | 6        | 5     | 4     | 3     | 2     | 1     | 0     |          |            |
| R/W-0 | R/W-0    | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | DMAMODE2 | DMAMODECTL |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-58. Supplementary Control Register (SCR) Field Descriptions**

| Bit  | Field                | Value                       | Description  |
|------|----------------------|-----------------------------|--|
| 15-8 | Reserved             | 0                           | Reserved.  |
| 7    | RXTRIGGRANU1         | 0<br>1                      | Disables the granularity of 1 for trigger RX level.<br>Enables the granularity of 1 for trigger RX level.  |
| 6    | TXTRIGGRANU1         | 0<br>1                      | Disables the granularity of 1 for trigger TX level.<br>Enables the granularity of 1 for trigger TX level.  |
| 5    | DSRIT                | 0<br>1                      | Disables DSR interrupt.<br>Enables DSR interrupt.  |
| 4    | RXCTSDSRWAKEUPENABLE | 0<br>1                      | RX CTS wake-up enable.<br>Disables the WAKE UP interrupt and clears SSR[1].<br>Waits for a falling edge of RX, CTS, or DSR pins to generate an interrupt.  |
| 3    | TXEMPTYCTLIT         | 0<br>1                      | Normal mode for THR interrupt.<br>THR interrupt is generated when TX FIFO and TX shift register are empty.   |
| 2-1  | DMAMODE2             | 0-3h<br>0<br>1h<br>2h<br>3h | Specifies the DMA mode valid if SCR[0] = 1:<br>DMA mode 0 (no DMA).<br>DMA mode 1 (UARTnDMAREQ[0] in TX, UARTnDMAREQ[1] in RX)<br>DMA mode 2 (UARTnDMAREQ[0] in RX)<br>DMA mode 3 (UARTnDMAREQ[0] in TX) |
| 0    | DMAMODECTL           | 0<br>1                      | The DMAMODE is set with FCR[3].<br>The DMAMODE is set with SCR[2:1].   |

### 19.5.1.28 Supplementary Status Register (SSR)

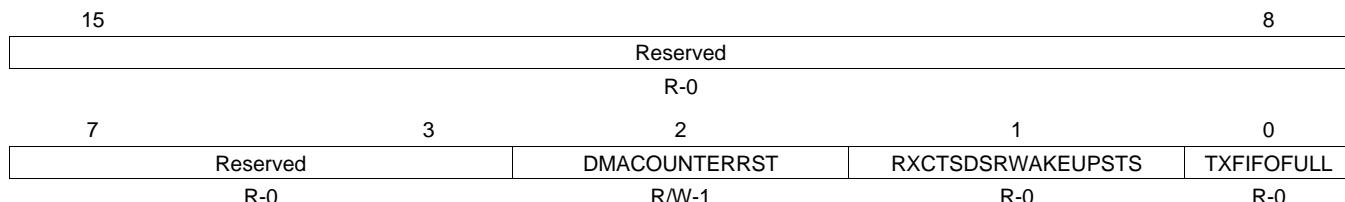
The supplementary status register (SSR) is shown in [Figure 19-61](#) and described in [Table 19-59](#).

---

**NOTE:** Bit 1 is reset only when SCR[4] is reset to 0.

---

**Figure 19-61. Supplementary Status Register (SSR)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-59. Supplementary Status Register (SSR) Field Descriptions**

| Bit  | Field             | Value | Description   |
|------|-------------------|-------|---|
| 15-3 | Reserved          | 0     | Reserved.   |
| 2    | DMACOUNTERRST     | 0     | The DMA counter will not be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]).                   |
|      |                   | 1     | The DMA counter will be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]).                       |
| 1    | RXCTSDSRWAKEUPSTS | 0     | Pin falling edge detection: Reset only when SCR[4] is reset to 0.<br>No falling-edge event on RX, CTS, and DSR. |
|      |                   | 1     | A falling edge occurred on RX, CTS, or DSR.   |
|      |                   | 0     | TX FIFO is not full.  |
|      |                   | 1     | TX FIFO is full.  |
| 0    | TXFIFOFULL        | 0     |   |

### 19.5.1.29 BOF Length Register (EBLR)

In IrDA SIR operation, the BOF length register (EBLR) specifies the number of BOF + xBOFs to transmit. The value set into this register must consider the BOF character; therefore, to send only one BOF with no XBOF, this register must be set to 1. To send one BOF with  $n$  XBOFs, this register must be set to  $n + 1$ . Furthermore, the value 0 sends 1 BOF plus 255 XBOFs.

In IrDA MIR mode, the BOF length register (EBLR) specifies the number of additional start flags (MIR protocol mandates a minimum of 2 start flags).

In CIR mode, the BOF length register (EBLR) specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]). All the received zeros are stored in the RX FIFO. When the register is cleared to 0, this feature is deactivated and always in reception state, which is disabled by setting the ACREG[5] bit to 1.

The BOF length register (EBLR) is shown in [Figure 19-62](#) and described in [Table 19-60](#).

---

**NOTE:** If the RX\_STOP interrupt occurs before a byte boundary, the remaining bits of the last byte are filled with zeros and then passed into the RX FIFO.

---

**Figure 19-62. BOF Length Register (EBLR)**

| 15       | 8 | 7     | 0 |
|----------|---|-------|---|
| Reserved |   | EBLR  |   |
| R-0      |   | R/W-0 |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-60. BOF Length Register (EBLR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 15-8 | Reserved | 0     | Reserved.   |
| 7-0  | EBLR     |       | IrDA mode: This register allows definition of up to 176 xBOFs, the maximum required by IrDA specification.<br>CIR mode: This register specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]).<br>0 Feature disabled.<br>1h Generate RXSTOP interrupt after receiving 1 zero bit.<br>...<br>FFh Generate RXSTOP interrupt after receiving 255 zero bits. |

### **19.5.1.30 Module Version Register (MVR)**

The reset value is fixed by hardware and corresponds to the RTL revision of this module. A reset has no effect on the value returned. The module version register (MVR) is shown in [Figure 19-63](#) and described in [Table 19-61](#).

**Figure 19-63. Module Version Register (MVR)**

| 15       | 8 | 7 | 4         | 3 | 0         |
|----------|---|---|-----------|---|-----------|
| Reserved |   |   | MAJORREV  |   | MINORREV  |
| R-0      |   |   | R-unknown |   | R-unknown |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-61. Module Version Register (MVR) Field Descriptions**

| Bit  | Field    | Value | Description                          |
|------|----------|-------|--------------------------------------|
| 15-8 | Reserved | 0     | Reserved.                            |
| 7-4  | MAJORREV | 0-Fh  | Major revision number of the module. |
| 3-0  | MINORREV | 0-Fh  | Minor revision number of the module. |

### 19.5.1.31 System Configuration Register (SYSC)

The AUTOIDLE bit controls a power-saving technique to reduce the logic power consumption of the module interface; that is, when the feature is enabled, the interface clock is gated off until the module interface is accessed. When the SOFTRESET bit is set high, it causes a full device reset. The system configuration register (SYSC) is shown in [Figure 19-64](#) and described in [Table 19-62](#).

**Figure 19-64. System Configuration Register (SYSC)**

| 15       | 5 | 4        | 3         | 2         | 1        | 0 |
|----------|---|----------|-----------|-----------|----------|---|
| Reserved |   | IDLEMODE | ENAWAKEUP | SOFTRESET | AUTOIDLE |   |
| R-0      |   | R/W-0    | R/W-0     | R/W-0     | R/W-0    |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-62. System Configuration Register (SYSC) Field Descriptions**

| Bit  | Field     | Value | Description   |
|------|-----------|-------|---|
| 15-5 | Reserved  | 0     | Reserved.   |
| 4-3  | IDLEMODE  | 0-3h  | Power management req/ack control.<br>0 Force idle: Idle request is acknowledged unconditionally.<br>1h No-idle: Idle request is never acknowledged.<br>2h Smart idle: Acknowledgement to an idle request is given based in the internal activity of the module.<br>3h Smart idle Wakeup: Acknowledgement to an idle request is given based in the internal activity of the module. The module is allowed to generate wakeup request. Only available on UART0. |
| 2    | ENAWAKEUP | 0     | Wakeup control.<br>0 Wakeup is disabled.<br>1 Wakeup capability is enabled.   |
| 1    | SOFTRESET | 0     | Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Read returns 0.<br>0 Normal mode.<br>1 Module is reset.   |
| 0    | AUTOIDLE  | 0     | Internal interface clock-gating strategy.<br>0 Clock is running.<br>1 Reserved.   |

### 19.5.1.32 System Status Register (SYSS)

The system status register (SYSS) is shown in [Figure 19-65](#) and described in [Table 19-63](#).

**Figure 19-65. System Status Register (SYSS)**

| 15       | 1 | 0         |
|----------|---|-----------|
| Reserved |   | RESETDONE |
| R-0      |   | R-0       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-63. System Status Register (SYSS) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 15-1 | Reserved  | 0     | Reserved.  |
| 0    | RESETDONE | 0     | Internal reset monitoring.<br>0 Internal module reset is ongoing.<br>1 Reset complete. |

### 19.5.1.33 Wake-Up Enable Register (WER)

The wake-up enable register (WER) is used to mask and unmask a UART event that subsequently notifies the system. An event is any activity in the logic that can cause an interrupt and/or an activity that requires the system to wake up. Even if wakeup is disabled for certain events, if these events are also an interrupt to the UART, the UART still registers the interrupt as such. The wake-up enable register (WER) is shown in [Figure 19-66](#) and described in [Table 19-64](#).

**Figure 19-66. Wake-Up Enable Register (WER)**

| 15         | Reserved      |               |             |              |             |              |              |       | 8   |
|------------|---------------|---------------|-------------|--------------|-------------|--------------|--------------|-------|-----|
| Reserved   |               |               |             |              |             |              |              |       | R-0 |
| 7          | 6             | 5             | 4           | 3            | 2           | 1            | 0            |       |     |
| TXWAKEUPEN | RLS_INTERRUPT | RHR_INTERRUPT | RX_ACTIVITY | DCD_ACTIVITY | RI_ACTIVITY | DSR_ACTIVITY | CTS_ACTIVITY |       |     |
| R/W-1      | R/W-1         | R/W-1         | R/W-1       | R/W-1        | R/W-1       | R/W-1        | R/W-1        | R/W-1 |     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-64. Wake-Up Enable Register (WER) Field Descriptions**

| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 15-8 | Reserved      | 0     | Reserved.  |
| 7    | TXWAKEUPEN    | 0     | Wake-up interrupt.<br>Event is not allowed to wake up the system.                    |
|      |               | 1     | Event can wake up the system: Event can be: THRIT or TXDMA request and/or TXSATUSIT. |
| 6    | RLS_INTERRUPT | 0     | Receiver line status interrupt.<br>Event is not allowed to wake up the system.       |
|      |               | 1     | Event can wake up the system.  |
| 5    | RHR_INTERRUPT | 0     | RHR interrupt.<br>Event is not allowed to wake up the system.                        |
|      |               | 1     | Event can wake up the system.  |
| 4    | RX_ACTIVITY   | 0     | Event is not allowed to wake up the system.  |
|      |               | 1     | Event can wake up the system.  |
| 3    | DCD_ACTIVITY  | 0     | Event is not allowed to wake up the system.  |
|      |               | 1     | Event can wake up the system.  |
| 2    | RI_ACTIVITY   | 0     | Event is not allowed to wake up the system.  |
|      |               | 1     | Event can wake up the system.  |
| 1    | DSR_ACTIVITY  | 0     | Event is not allowed to wake up the system.  |
|      |               | 1     | Event can wake up the system.  |
| 0    | CTS_ACTIVITY  | 0     | Event is not allowed to wake up the system.  |
|      |               | 1     | Event can wake up the system.  |

### 19.5.1.34 Carrier Frequency Prescaler Register (CFPS)

Since the consumer IR (CIR) works at modulation rates of 30–56.8 kHz, the 48 MHz clock must be prescaled before the clock can drive the IR logic. The carrier frequency prescaler register (CFPS) sets the divisor rate to give a range to accommodate the remote control requirements in BAUD multiples of 12x. The value of the CFPS at reset is 105 decimal (69h), which equates to a 38.1 kHz output from starting conditions. The 48 MHz carrier is prescaled by the CFPS that is then divided by the 12x BAUD multiple. The carrier frequency prescaler register (CFPS) is shown in [Figure 19-67](#) and described in [Table 19-65](#).

**Figure 19-67. Carrier Frequency Prescaler Register (CFPS)**

| 15       | 8 | 7       | 0 |
|----------|---|---------|---|
| Reserved |   | CFPS    |   |
| R-0      |   | R/W-69h |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-65. Carrier Frequency Prescaler Register (CFPS) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved.  |
| 7-0  | CFPS     | 0-FFh | System clock frequency prescaler at (12x multiple). CFPS = 0 is not supported. Examples for CFPS values: |

| Target Frequency (kHz) | CFPS (decimal) | Actual Frequency (kHz) |
|------------------------|----------------|------------------------|
| 30                     | 133            | 30.08                  |
| 32.75                  | 122            | 32.79                  |
| 36                     | 111            | 36.04                  |
| 36.7                   | 109            | 36.69                  |
| 38                     | 105            | 38.1                   |
| 40                     | 100            | 40                     |
| 56.8                   | 70             | 57.14                  |

### 19.5.1.35 Divisor Latches Low Register (DLL)

The divisor latches low register (DLL) with the DLH register stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most-significant part of the divisor, DLL stores the least-significant part of the divisor. The DLL register is shown in [Figure 19-68](#) and described in [Table 19-66](#).

---

**NOTE:** DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

---

**Figure 19-68. Divisor Latches Low Register (DLL)**

|          |   |           |   |
|----------|---|-----------|---|
| 15       | 8 | 7         | 0 |
| Reserved |   | CLOCK_LSB |   |
| R-0      |   | R/W-0     |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-66. Divisor Latches Low Register (DLL) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 15-8 | Reserved  | 0     | Reserved.  |
| 7-0  | CLOCK_LSB | 0-FFh | Divisor latches low. Stores the 8 LSB divisor value. |

### 19.5.1.36 Divisor Latches High Register (DLH)

The divisor latches high register (DLH) with the DLL register stores the 14-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most-significant part of the divisor, DLL stores the least-significant part of the divisor. The DLH register is shown in [Figure 19-69](#) and described in [Table 19-67](#).

---

**NOTE:** DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

---

**Figure 19-69. Divisor Latches High Register (DLH)**

|          |   |           |   |
|----------|---|-----------|---|
| 15       | 6 | 5         | 0 |
| Reserved |   | CLOCK_MSB |   |
| R-0      |   | R/W-0     |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-67. Divisor Latches High Register (DLH) Field Descriptions**

| Bit  | Field     | Value | Description   |
|------|-----------|-------|---|
| 15-6 | Reserved  | 0     | Reserved.   |
| 5-0  | CLOCK_MSB | 0-3Fh | Divisor latches high. Stores the 6 MSB divisor value. |

### 19.5.1.37 Enhanced Feature Register (EFR)

The enhanced feature register (EFR) enables or disables enhanced features. Most enhanced functions apply only to UART modes, but EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in IrDA modes. The enhanced feature register (EFR) is shown in [Figure 19-70](#) and described in [Table 19-68](#).

**Figure 19-70. Enhanced Feature Register (EFR)**

|           |           |                   |            |   |               |   |
|-----------|-----------|-------------------|------------|---|---------------|---|
| 15        | Reserved  |                   |            |   |               | 8 |
| R-0       |           |                   |            |   |               |   |
| 7         | 6         | 5                 | 4          | 3 | 0             |   |
| AUTOCTSEN | AUTORTSEN | SPECIALCHARDETECT | ENHANCEDEN |   | SWFLOWCONTROL |   |

R/W-0      R/W-0      R/W-0      R/W-0      R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-68. Enhanced Feature Register (EFR) Field Descriptions**

| Bit  | Field             | Value | Description   |
|------|-------------------|-------|---|
| 15-8 | Reserved          | 0     | Reserved.   |
| 7    | AUTOCTSEN         | 0     | Auto-CTS enable bit (UART mode only).<br>Normal operation.  |
|      |                   | 1     | Auto-CTS flow control is enabled; transmission is halted when the CTS pin is high (inactive).   |
| 6    | AUTORTSEN         | 0     | Auto-RTS enable bit (UART mode only).<br>Normal operation.  |
|      |                   | 1     | Auto-RTS flow control is enabled; RTS pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached.  |
| 5    | SPECIALCHARDETECT | 0     | Special character detect (UART mode only).<br>Normal operation.   |
|      |                   | 1     | Special character detect enable. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to RX FIFO and the IIR[4] bit is set to 1 to indicate that a special character was detected.  |
| 4    | ENHANCEDEN        | 0     | Enhanced functions write enable bit.<br>Disables writing to IER[7:4], FCR[5:4], and MCR[7:5].   |
|      |                   | 1     | Enables writing to IER[7:4], FCR[5:4], and MCR[7:5].  |
| 3-0  | SWFLOWCONTROL     | 0-Fh  | Combinations of software flow control can be selected by programming this bit. XON1 and XON2 should be set to different values if the software flow control is enabled. See <a href="#">Table 19-69</a> . In IrDA mode, EFR[1:0] selects the IR address to check. See <a href="#">IR Address Checking</a> . |

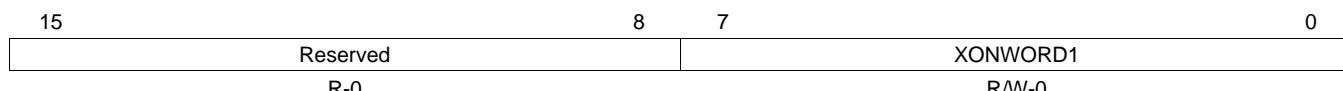
**Table 19-69. EFR[3:0] Software Flow Control Options**

| EFR[3:0] |       |       |       |   |
|----------|-------|-------|-------|---|
| Bit 3    | Bit 2 | Bit 1 | Bit 0 | TX/RX Software Flow Control                               |
| 0        | 0     | X     | X     | No transmit flow control                                  |
| 1        | 0     | X     | X     | Transmit XON1, XOFF1                                      |
| 0        | 1     | X     | X     | Transmit XON2, XOFF2                                      |
| 1        | 1     | X     | X     | Transmit XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup>          |
| X        | X     | 0     | 0     | No receive flow control                                   |
| X        | X     | 1     | 0     | Receiver compares XON1, XOFF1                             |
| X        | X     | 0     | 1     | Receiver compares XON2, XOFF2                             |
| X        | X     | 1     | 1     | Receiver compares XON1, XON2: XOFF1, XOFF2 <sup>(1)</sup> |

<sup>(1)</sup> The XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.

### 19.5.1.38 XON1/ADDR1 Register

In UART mode, XON1 character; in IrDA mode, ADDR1 address 1. The XON1/ADDR1 register is shown in [Figure 19-71](#) and described in [Table 19-70](#).

**Figure 19-71. XON1/ADDR1 Register**

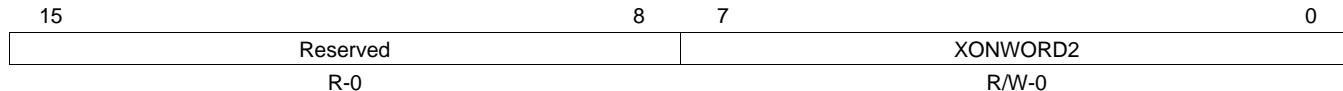
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-70. XON1/ADDR1 Register Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved.  |
| 7-0  | XONWORD1 | 0-FFh | Stores the 8-bit XON1 character in UART modes and ADDR1 address 1 in IrDA modes. |

### 19.5.1.39 XON2/ADDR2 Register

In UART mode, XON2 character; in IrDA mode, ADDR2 address 2. The XON2/ADDR2 register is shown in [Figure 19-72](#) and described in [Table 19-71](#).

**Figure 19-72. XON2/ADDR2 Register**

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-71. XON2/ADDR2 Register Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved.  |
| 7-0  | XONWORD2 | 0-FFh | Stores the 8-bit XON2 character in UART modes and ADDR2 address 2 in IrDA modes. |

#### 19.5.1.40 XOFF1 Register

In UART mode, XOFF1 character. The XOFF1 register is shown in [Figure 19-73](#) and described in [Table 19-72](#).

**Figure 19-73. XOFF1 Register**

| 15       | 8 | 7 | 0         |
|----------|---|---|-----------|
| Reserved |   |   | XOFFWORD1 |
| R-0      |   |   | R/W-0     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-72. XOFF1 Register Field Descriptions**

| Bit  | Field     | Value | Description                                     |
|------|-----------|-------|---|
| 15-8 | Reserved  | 0     | Reserved.                                       |
| 7-0  | XOFFWORD1 | 0-FFh | Stores the 8-bit XOFF1 character in UART modes. |

#### 19.5.1.41 XOFF2 Register

In UART mode, XOFF2 character. The XOFF2 register is shown in [Figure 19-74](#) and described in [Table 19-73](#).

**Figure 19-74. XOFF2 Register**

| 15       | 8 | 7 | 0         |
|----------|---|---|-----------|
| Reserved |   |   | XOFFWORD2 |
| R-0      |   |   | R/W-0     |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-73. XOFF2 Register Field Descriptions**

| Bit  | Field     | Value | Description                                     |
|------|-----------|-------|---|
| 15-8 | Reserved  | 0     | Reserved.                                       |
| 7-0  | XOFFWORD2 | 0-FFh | Stores the 8-bit XOFF2 character in UART modes. |

### 19.5.1.42 Transmit Frame Length Low Register (TXFLL)

The transmit frame length low register (TXFLL) and the TXFLH register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used. The transmit frame length low register (TXFLL) is shown in [Figure 19-75](#) and described in [Table 19-74](#).

**Figure 19-75. Transmit Frame Length Low Register (TXFLL)**

| 15       | 8 | 7 | 0     |
|----------|---|---|-------|
| Reserved |   |   | TXFLL |
| R-0      |   |   | W-0   |

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

**Table 19-74. Transmit Frame Length Low Register (TXFLL) Field Descriptions**

| Bit  | Field    | Value | Description                                    |
|------|----------|-------|--|
| 15-8 | Reserved | 0     | Reserved.                                      |
| 7-0  | TXFLL    | 0-FFh | LSB register used to specify the frame length. |

### 19.5.1.43 Transmit Frame Length High Register (TXFLH)

The transmit frame length high register (TXFLH) and the TXFLL register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used. The transmit frame length high register (TXFLH) is shown in [Figure 19-76](#) and described in [Table 19-75](#).

**Figure 19-76. Transmit Frame Length High Register (TXFLH)**

| 15       | 5 | 4 | 0     |
|----------|---|---|-------|
| Reserved |   |   | TXFLH |
| R-0      |   |   | W-0   |

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

**Table 19-75. Transmit Frame Length High Register (TXFLH) Field Descriptions**

| Bit  | Field    | Value | Description                                    |
|------|----------|-------|--|
| 15-5 | Reserved | 0     | Reserved.                                      |
| 4-0  | TXFLH    | 0-1Fh | MSB register used to specify the frame length. |

#### 19.5.1.44 Received Frame Length Low Register (RXFLL)

The received frame length low register (RXFLL) and the RXFLH register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is  $n$  bytes, program RXFLL and RXFLH to be  $n + 3$  in SIR or MIR modes and  $n + 6$  in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag). The received frame length low register (RXFLL) is shown in [Figure 19-77](#) and described in [Table 19-76](#).

**Figure 19-77. Received Frame Length Low Register (RXFLL)**

| 15       | 8 | 7     | 0 |
|----------|---|-------|---|
| Reserved |   | RXFLL |   |
| R-0      |   | W-0   |   |

LEGEND: R/W = Read/Write; W = Write only; -n = value after reset

**Table 19-76. Received Frame Length Low Register (RXFLL) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 15-8 | Reserved | 0     | Reserved.   |
| 7-0  | RXFLL    | 0-FFh | LSB register used to specify the frame length in reception. |

#### 19.5.1.45 Received Frame Length High Register (RXFLH)

The received frame length high register (RXFLH) and the RXFLL register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is  $n$  bytes, program RXFLL and RXFLH to be  $n + 3$  in SIR or MIR modes and  $n + 6$  in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag). The received frame length high register (RXFLH) is shown in [Figure 19-78](#) and described in [Table 19-77](#).

**Figure 19-78. Received Frame Length High Register (RXFLH)**

| 15       | 4 | 3     | 0 |
|----------|---|-------|---|
| Reserved |   | RXFLH |   |
| R-0      |   | W-0   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-77. Received Frame Length High Register (RXFLH) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 15-4 | Reserved | 0     | Reserved.   |
| 3-0  | RXFLH    | 0-Fh  | MSB register used to specify the frame length in reception. |

### 19.5.1.46 UART Autobauding Status Register (UASR)

The UART autobauding status register (UASR) returns the speed, the number of bits by characters, and the type of parity in UART autobauding mode. In autobauding mode, the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition. The UART autobauding status register (UASR) is shown in [Figure 19-79](#) and described in [Table 19-78](#).

**NOTE:**

- This register is used to set up transmission according to characteristics of previous reception, instead of LCR, DLL, and DLH registers when UART is in autobauding mode.
- To reset the autobauding hardware (to start a new “AT” detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to 7h (reset state), then set to 2h (UART in autobaud mode) or cleared to 0 (UART in standard mode).

**Usage limitation:**

- Only 7 and 8 bits character (5 and 6 bits not supported).
- 7 bits character with space parity not supported.
- Baud rate between 1200 and 115 200 bp/s (10 possibilities).

**Figure 19-79. UART Autobauding Status Register (UASR)**

| 15       | 8 | 7          | 6         | 5 | 4     | 0 |
|----------|---|------------|-----------|---|-------|---|
| Reserved |   | PARITYTYPE | BITBYCHAR |   | SPEED |   |
| R-0      |   | R-0        | R-0       |   | R-0   |   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-78. UART Autobauding Status Register (UASR) Field Descriptions**

| Bit  | Field      | Value  | Description                                  |
|------|------------|--------|--|
| 15-8 | Reserved   | 0      | Reserved.                                    |
| 7-6  | PARITYTYPE | 0-3h   | Type of the parity in UART autobauding mode. |
|      |            | 0      | No parity identified                         |
|      |            | 1h     | Parity space                                 |
|      |            | 2h     | Even parity                                  |
|      |            | 3h     | Odd parity                                   |
| 5    | BITBYCHAR  |        | Number of bits by characters.                |
|      |            | 0      | 7-bit character identified.                  |
|      |            | 1      | 8-bit character identified.                  |
| 4-0  | SPEED      | 0-1Fh  | Speed  |
|      |            | 0      | No speed identified.                         |
|      |            | 1h     | 115 200 baud                                 |
|      |            | 2h     | 57 600 baud                                  |
|      |            | 3h     | 38 400 baud                                  |
|      |            | 4h     | 28 800 baud                                  |
|      |            | 5h     | 19 200 baud                                  |
|      |            | 6h     | 14 400 baud                                  |
|      |            | 7h     | 9600 baud                                    |
|      |            | 8h     | 4800 baud                                    |
|      |            | 9h     | 2400 baud                                    |
|      |            | Ah     | 1200 baud                                    |
|      |            | Bh-1Fh | Reserved                                     |

#### 19.5.1.47 Received FIFO Level (RXFIFO\_LVL) Register

The Received FIFO Level register (RXFIFO\_LVL) is shown in [Figure 19-80](#) and described in [Table 19-79](#).

**Figure 19-80. RXFIFO\_LVL Register**

| 31       | 8 | 7 | 0          |
|----------|---|---|------------|
| Reserved |   |   | RXFIFO_LVL |
| R-0      |   |   | R-0        |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-79. RXFIFO\_LVL Register Field Descriptions**

| Bit  | Field      | Value | Description          |
|------|------------|-------|----------------------|
| 31-8 | Reserved   | 0     | Reserved.            |
| 7-0  | RXFIFO_LVL | 0     | Level of the RX FIFO |

#### 19.5.1.48 Transmit FIFO Level (TXFIFO\_LVL) Register

The Transmit FIFO Level register (TXFIFO\_LVL) is shown in [Figure 19-81](#) and described in [Table 19-80](#).

**Figure 19-81. TXFIFO\_LVL Register**

| 31       | 8 | 7 | 0          |
|----------|---|---|------------|
| Reserved |   |   | TXFIFO_LVL |
| R-0      |   |   | R-0        |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-80. TXFIFO\_LVL Register Field Descriptions**

| Bit  | Field      | Value | Description          |
|------|------------|-------|----------------------|
| 31-8 | Reserved   | 0     | Reserved.            |
| 7-0  | TXFIFO_LVL | 0     | Level of the TX FIFO |

### 19.5.1.49 IER2 Register

The IER2 register enables RX/TX FIFOs empty corresponding interrupts. The IER2 register (IER2) is shown in [Figure 19-82](#) and described in [Table 19-81](#).

**Figure 19-82. IER2 Register**

| 31 | Reserved | 2 | 1     | 0     |
|----|----------|---|-------|-------|
|    | R-0      |   | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-81. IER2 Register Field Descriptions**

| Bit  | Field               | Value  | Description  |
|------|---------------------|--------|--|
| 31-2 | Reserved            | 0      | Reserved.  |
| 1    | EN_TXFIFO_EM<br>PTY | 0<br>1 | Disables EN_TXFIFO_EMPTY interrupt.<br>Enables EN_TXFIFO_EMPTY interrupt.                                  |
| 0    | EN_RXFIFO_EM<br>PTY | 0<br>1 | Number of bits by characters.<br>Disables EN_RXFIFO_EMPTY interrupt.<br>Enables EN_RXFIFO_EMPTY interrupt. |

### **19.5.1.50 ISR2 Register**

The ISR2 register displays the status of RX/TX FIFOs empty corresponding interrupts. The ISR2 register (ISR2) is shown in [Figure 19-79](#) and described in [Table 19-78](#).

**Figure 19-83. ISR2 Register**

| 31 | Reserved | 1                    | 0                    |
|----|----------|----------------------|----------------------|
|    | R-0      | TXFIFO_EMPT<br>Y_STS | RXFIFO_EMPT<br>Y_STS |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-82. ISR2 Register Field Descriptions**

| <b>Bit</b> | <b>Field</b>     | <b>Value</b> | <b>Description</b>   |
|------------|------------------|--------------|--|
| 31-2       | Reserved         | 0            | Reserved.  |
| 1          | TXFIFO_EMPTY_STS | 0<br>1       | TXFIFO_EMPTY interrupt not pending.<br>TXFIFO_EMPTY interrupt pending. |
| 0          | RXFIFO_EMPTY_STS | 0<br>1       | RXFIFO_EMPTY interrupt not pending.<br>RXFIFO_EMPTY interrupt pending. |

### 19.5.1.51 FREQ\_SEL Register

The FREQ\_SEL register (FREQ\_SEL) is shown in [Figure 19-84](#) and described in [Table 19-83](#).

**Figure 19-84. FREQ\_SEL Register**

| 31       | 8 | 7 | 0        |
|----------|---|---|----------|
| Reserved |   |   | FREQ_SEL |
| R-0      |   |   | R/W-1A   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-83. FREQ\_SEL Register Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-8 | Reserved | 0     | Reserved.   |
| 7-0  | FREQ_SEL | 1A    | Sets the sample per bit if non default frequency is used. MDR3[1] must be set to 1 after this value is set. Must be equal or higher then 6. |

### 19.5.1.52 Mode Definition Register 3 (MDR3) Register

The DISABLE\_CIR\_RX\_DEMOD register bit will force the CIR receiver to bypass demodulation of received data if set. See the CIR Mode Block Components. The NONDEFAULT\_FREQ register bit allows the user to set sample per bit by writing it into FREQ\_SEL register. Set it if non-default (48 MHz) fclk frequency is used to achieve a less than 2% error rate. Changing this bit (to any value) will automatically disable the device by setting MDR[2:0] to "111".

The Mode Definition Register 3 (MDR3) register is shown in [Figure 19-79](#) and described in [Table 19-78](#).

**Figure 19-85. Mode Definition Register 3 (MDR3) Register**

| 31       | 3                    | 2               | 1                    | 0     |
|----------|----------------------|-----------------|----------------------|-------|
| Reserved | SET_DMA_TX_THRESHOLD | NONDEFAULT_FREQ | DISABLE_CIR_RX_DEMOD |       |
| R-0      |                      | R/W-0           |                      | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-84. Mode Definition Register 3 (MDR3) Register Field Descriptions**

| Bit  | Field                | Value | Description   |
|------|----------------------|-------|---|
| 31-3 | Reserved             | 0     | Reserved.   |
| 2    | SET_DMA_TX_THRESHOLD | 0     | Disable use of TX DMA Threshold register. Use 64-TX trigger as DMA threshold. |
|      |                      | 1     | Enable to set different TX DMA threshold in the TX DMA Threshold register.    |
| 1    | NONDEFAULT_FREQ      | 0     | Disables using NONDEFAULT fclk frequencies                                    |
|      |                      | 1     | Enables using NONDEFAULT fclk frequencies (set FREQ_SEL and DLH/DLL)          |
| 0    | DISABLE_CIR_RX_DEMOD | 0     | Enables CIR RX demodulation   |
|      |                      | 1     | Disables CIR RX demodulation  |

### 19.5.1.53 TX\_DMA\_THRESHOLD Register

TX DMA threshold value

The TX\_DMA\_THRESHOLD register is shown in [Figure 19-79](#) and described in [Table 19-78](#).

**Figure 19-86. TX\_DMA\_THRESHOLD Register**

| 31 | Reserved | 6 | 5 | 0     |
|----|----------|---|---|-------|
|    | R-0      |   |   | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 19-85. TX\_DMA\_THRESHOLD Register Field Descriptions**

| Bit  | Field            | Value | Description   |
|------|------------------|-------|---|
| 31-6 | Reserved         | 0     | Reserved.   |
| 5-0  | TX_DMA_THRESHOLD | 0     | Used to manually set the TX DMA threshold level. <a href="#">UART_MDR3[2] SET_TX_DMA_THRESHOLD</a> must be 1 and must be value + tx_trigger_level = 64 (TX FIFO size). If not, 64-tx_trigger_level will be used without modifying the value of this register. |

**Timers**

This chapter describes the timers for the device.

| Topic                         | Page        |
|-------------------------------|-------------|
| <b>20.1 DMTimer .....</b>     | <b>3538</b> |
| <b>20.2 DMTimer 1ms .....</b> | <b>3569</b> |
| <b>20.3 RTC_SS .....</b>      | <b>3605</b> |
| <b>20.4 WATCHDOG .....</b>    | <b>3653</b> |

## 20.1 DMTimer

### 20.1.1 Introduction

#### 20.1.1.1 Overview

The timer module contains a free running upward counter with auto reload capability on overflow. The timer counter can be read and written in real-time (while counting). The timer module includes compare logic to allow an interrupt event on a programmable counter matching value.

A dedicated output signal can be pulsed or toggled on overflow and match event. This output offers a timing stamp trigger signal or PWM (pulse-width modulation) signal sources. A dedicated output signal can be used for general purpose PORPGPOCFG (directly driven by bit 14 of the TCLR register). A dedicated input signal is used to trigger automatic timer counter capture and interrupt event, on programmable input signal transition type. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged in one module interrupt line and one wake-up line. Each internal interrupt sources can be independently enabled/disabled.

This module is controllable through the OCP peripheral bus.

As two clock domains are managed inside this module, resynchronization is done by special logic between the OCP clock domain and the Timer clock domain. At reset, synchronization logic allows utilization of all ratios between the OCP clock and the Timer clock. A drawback of this mode is that full-resynchronization path is used with access latency performance impact in terms of OCP clock cycles. In order to improve module access latency, and under restricted conditions on clocks ratios, write-posted mode can be used by setting the POSTED bit of the System Control Register (TSCR). Under this mode, write posted mode is enabled, meaning that OCP write command is granted before write process complete in the timer clock domain. This mode allows software to do concurrent writes on Dual Mode timer registers and to observe write process completion (synchronization) at the software level by reading independent write posted status bits in the Write Posted Status Register (TWPS).

#### 20.1.1.2 Features

The timer consists of the following features:

- Counter timer with compare and capture modes
- Auto-reload mode
- Start-stop mode
- Programmable divider clock source
- 16-32 bit addressing
- “On the fly” read/write registers
- Interrupts generated on overflow, compare and capture
- Interrupt enable
- Wake-up enable (only for Timer0)
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- Dedicated output signal for general purpose use PORPGPOCFG
- OCP interface compatible

The Timer resolution and interrupt period are dependent on the selected input clock and clock prescale value. Example resolutions for common clock values are shown in [Table 20-1](#).

**Table 20-1. Timer Resolution and Maximum Range**

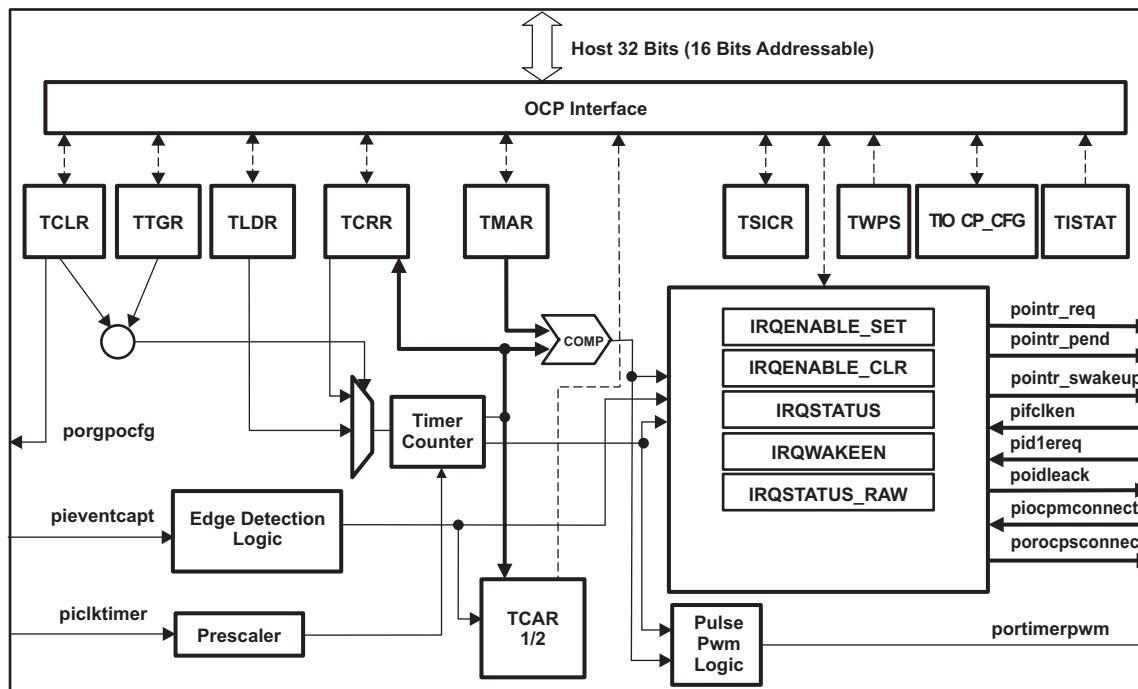
| Clock      | Prescaler | Resolution | Interrupt Period Range |
|------------|-----------|------------|------------------------|
| 32.768 KHz | 1 (min)   | 31.25 us   | 31.25 us to ~36h 35m   |
|            | 256 (max) | 8 ms       | 8 ms to ~391d 22h 48m  |

**Table 20-1. Timer Resolution and Maximum Range (continued)**

| Clock    | Prescaler | Resolution | Interrupt Period Range |
|----------|-----------|------------|------------------------|
| 27 MHz   | 1 (min)   | ~37 ns     | ~37 ns to ~159s        |
|          | 256 (max) | ~9.48 us   | ~19.7 us to ~11h 18m   |
| 38.4 MHz | 1 (min)   | ~26 ns     | ~26 ns to ~112 s       |
|          | 256 (max) | ~6.7 us    | ~13.3 us to ~7h 5m     |

### 20.1.1.3 Functional Block Diagram

Figure 20-1 shows a block diagram of the timer.

**Figure 20-1. Timer Block Diagram**


## 20.1.2 Integration

### 20.1.2.1 Timer Connectivity Attributes

**Table 20-2. Timer[0] Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Power domain        | Wakeup domain  |
| Clock Domain        | PD_WKUP_L4_WKUP_GCLK (Interface/OCP)<br>PD_WKUP_TIMER0_GCLK (Func) |
| Reset Signals       | WKUP_DOM_RST_N   |
| Idle/Wakeup Signals | Idle<br>Slave Wakeup   |
| Interrupt Requests  | 1 to MPU Subsystem (TINT0), 1 to WakeM3                            |
| DMA Requests        | None   |
| Physical Address    | L4 Wakeup slave port   |

**Table 20-3. Timer[2–7] Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L4LS_GCLK (OCP)<br>Functional Clocks:<br>PD_PER_TIMER2_GCLK (Timer 2)<br>PD_PER_TIMER3_GCLK (Timer 3)<br>PD_PER_TIMER4_GCLK (Timer 4)<br>PD_PER_TIMER5_GCLK (Timer 5)<br>PD_PER_TIMER6_GCLK (Timer 6)<br>PD_PER_TIMER7_GCLK (Timer 7) |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle (No wakeup capabilities)  |
| Interrupt Requests  | 1 per timer module to MPU Subsystem (TINT2 - TINT7)<br>Also to TSC_ADC event capture mux for Timer 4 – Timer7  |
| DMA Requests        | Interrupt requests are redirected as DMA requests: 1 per instance (TINTx)  |
| Physical Address    | L4 Peripheral slave port   |

### 20.1.2.2 Timer Clock and Reset Management

Each DMTimer[2–7] functional clock is selected within the PRCM using the associated CLKSEL\_TIMERx\_CLK register from 3 possible sources:

- The 24-MHz (typ) system clock (CLK\_M\_OSC)
- The PER PLL generated 32.768 KHz clock (CLK\_32KHZ)
- The TCLKIN external timer input clock.

The DMTimer 0 functional clock is fixed to use the internal 32KHz RC Clock (CLK\_RC32K).

### 20.1.2.3 Timer Clock Signals

**Table 20-4. Timer Clock Signals**

| Clock Signal                    | Max Freq              | Reference / Source  | Comments  |
|---------------------------------|-----------------------|---|---|
| <b>Timer[0] Clock Signals</b>   |                       |   |   |
| PICLKOPC<br>Interface clock     | 100 MHz               | CORE_CLKOUTM4 / 2   | pd_wkup_l4_wkup_gclk<br>from PRCM   |
| PICLKIMER<br>Functional clock   | 25 MHz <sup>(1)</sup> | CLK_RC32K   | pd_wkup_timer0_gclk<br>from PRCM  |
| <b>Timer[2–7] Clock Signals</b> |                       |   |   |
| PICLKOPC<br>Interface clock     | 100 MHz               | CORE_CLKOUTM4 / 2   | pd_per_l4ls_gclk<br>from PRCM   |
| PICLKIMER<br>Functional clock   | 25 MHz <sup>(1)</sup> | CLK_M_OSC<br>CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375)<br>TCLKIN | pd_per_timer2_gclk<br>pd_per_timer3_gclk<br>pd_per_timer4_gclk<br>pd_per_timer5_gclk<br>pd_per_timer6_gclk<br>pd_per_timer7_gclk<br>from PRCM |

<sup>(1)</sup> PICLKIMER must be less than PICLKOPC/4.

#### 20.1.2.4 Timer Pin List

The timer PIEVENTCAPT input and PORTIMERPWM output signals are muxed onto a single TIMER I/O pad. The pad direction (and hence the pin function) are controlled from within the DMTimer module using the PORGPOCFG signal as an output enable.

**Table 20-5. Timer Pin List**

| Pin    | Type | Description                         |
|--------|------|-------------------------------------|
| TCLKIN | I    | External timer clock source         |
| TIMER4 | I/O  | Timer 4 trigger input or PWM output |
| TIMER5 | I/O  | Timer 5 trigger input or PWM output |
| TIMER6 | I/O  | Timer 6 trigger input or PWM output |
| TIMER7 | I/O  | Timer 7 trigger input or PWM output |

## 20.1.3 Functional Description

### 20.1.3.1 Functional Description

The general-purpose timer is an upward counter. It supports 3 functional modes:

- Timer mode
- Capture mode
- Compare mode

By default, after core reset, the capture and compare modes are disabled.

#### 20.1.3.1.1 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the Timer Control Register (TCLR ST bit). The Timer Counter Register (TCRR) can be loaded when stopped or on the fly (while counting). TCRR can be loaded directly by a TCRR Write access with the new timer value. TCRR can also be loaded with the value held in the Timer Load Register TLDR by a trigger register (TTGR) Write access. The TCRR loading is done regardless the TTGR written value. The timer counter register TCRR value can be read when stopped or captured on the fly by a TCRR Read access. The timer is stopped and the counter value is cleared to "0" when the module's reset is asserted. The timer is maintained in stop after reset is released. When the timer is stopped TCRR is frozen and it can be restarted from the frozen value unless TCRR has been reloaded with a new value.

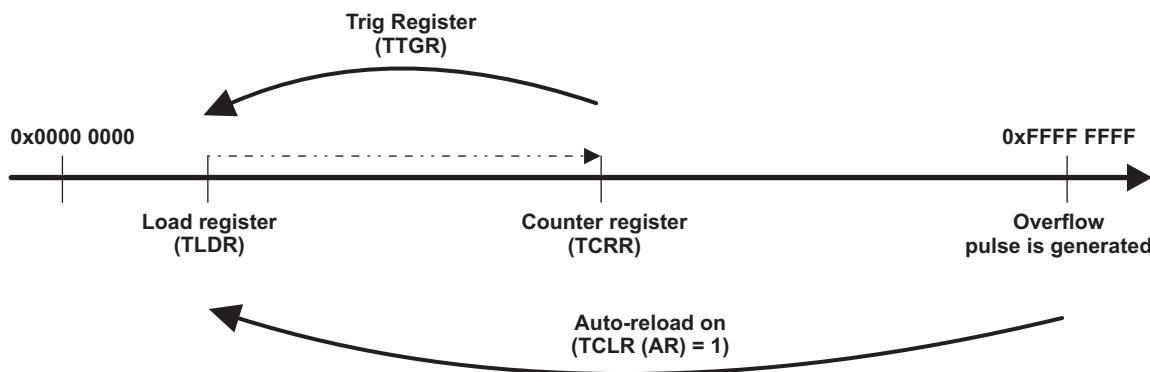
In the one shot mode (TCLR AR bit = 0), the counter is stopped after counting overflow (counter value remains at zero).

When the auto-reload mode is enabled (TCLR AR bit = 1), the TCRR is reloaded with the Timer Load Register (TLDR) value after a counting overflow.

It is not recommended to put the overflow value (FFFF FFFFh) in TLDR because it can lead to undesired results.

An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer Interrupt Enable Register (IRQENABLE\_SET\_OVF\_IT\_FLAG bit = 1). A dedicated output pin (PORTIMERPWM) is programmed through TCLR (TRG and PT bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs.

**Figure 20-2. TCRR Timing Value**



#### 20.1.3.1.2 Capture Mode Functionality

The timer value in TCRR can be captured and saved in TCAR1 or TCAR2 function of the mode selected in TCLR through the field CAPT\_MODE when a transition is detected on the module input pin (PIEVENTCAPT). The edge detection circuitry monitors transitions on the input pin (PIEVENTCAPT).

Rising transition, falling transition or both can be selected in TCLR (TCM bit) to trig the timer counter capture. The module sets the IRQSTATUS (TCAR\_IT\_FLAG bit) when an active transition is detected and at the same time the counter value TCRR is stored in one of the timer capture registers TCAR1 or TCAR2 as follows:

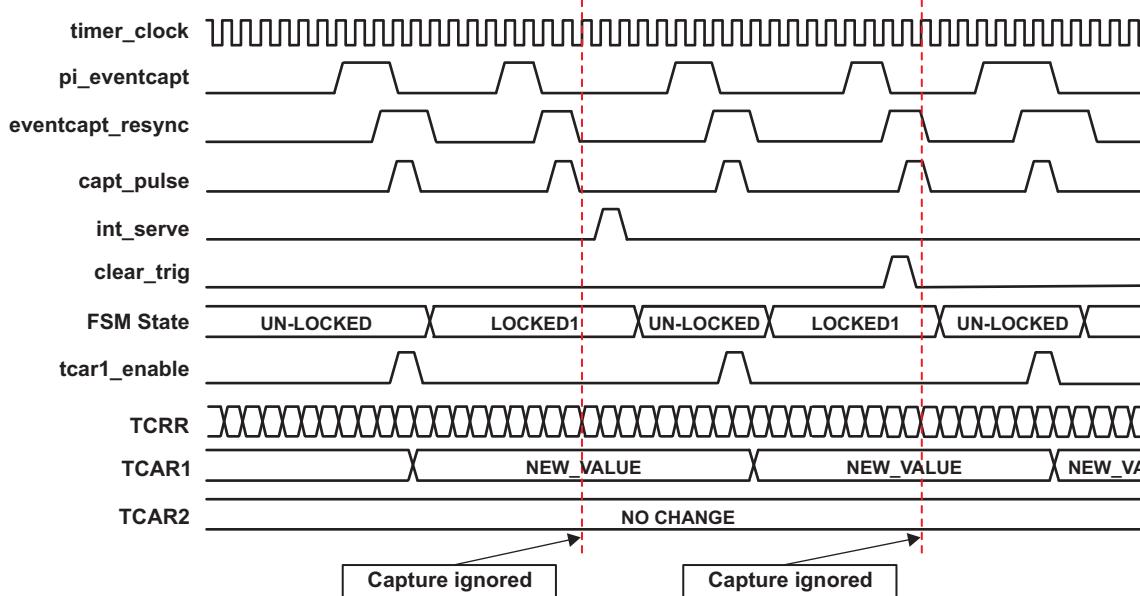
- If TCLR's CAPT\_MODE field is 0 then, on the first enabled capture event, the value of the counter register is saved in TCAR1 register and all the next events are ignored (no update on TCAR1 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a 1 in it.
- If TCLR's CAPT\_MODE field is 1 then, on the first enabled captured event, the counter value is saved in TCAR1 register and, on the second enabled capture event, the value of the counter register is saved in TCAR2 register. All the other events are ignored (no update on TCAR1/2 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a 1 in it. This mechanism is useful for period calculation of a clock if that clock is connected to the PIEVENTCAPT input pin.

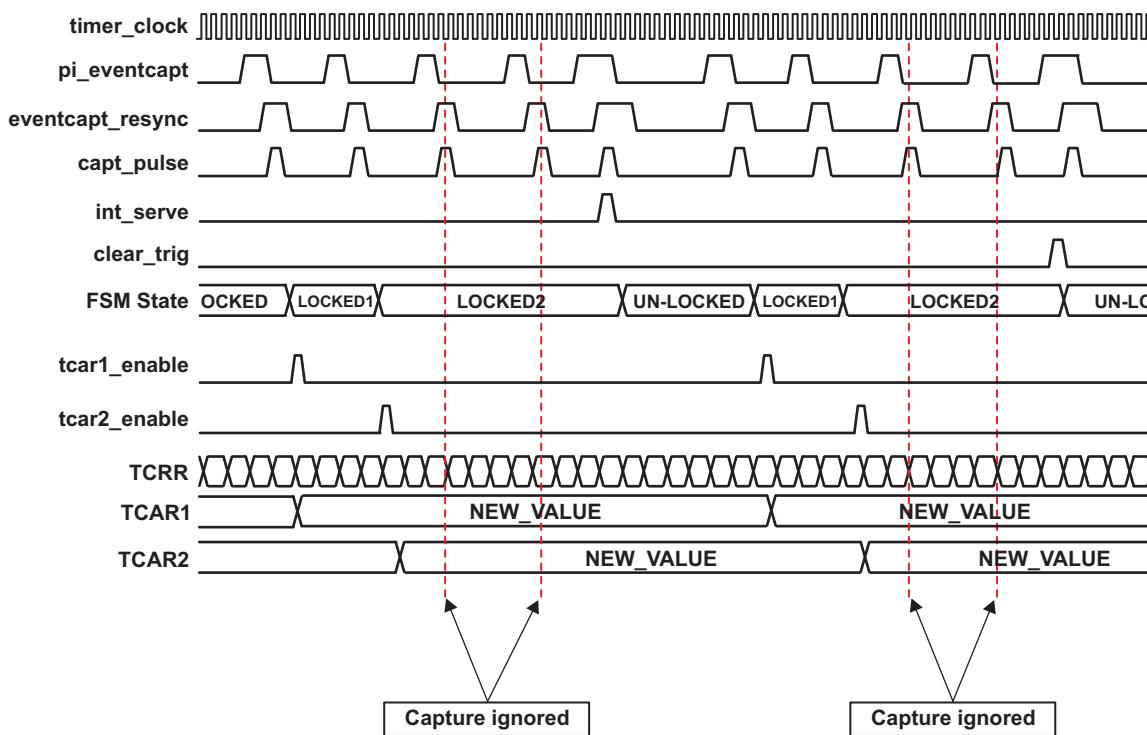
The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is served - TCAR\_IT\_FLAG bit of IRQSTATUS (previously 1) is cleared. The timer functional clock (input to prescaler) is used to sample the input pin (PIEVENTCAPT). Input negative or positive pulse can be detected when pulse time is above functional clock period. An interrupt can be issued on transition detection if the capture interrupt enable bit is set in the Timer Interrupt Enable Register IRQENABLE\_SET (TCAR\_IT\_FLAG bit).

In [Figure 20-3](#), the TCM value is 01 and CAPT\_MODE is 0 - only rising edge of the PIEVENTCAPT will trigger a capture in TCAR and only TCAR1 will update.

In [Figure 20-4](#), the TCM value is 01 and CAPT\_MODE is 1 - only rising edge of the PIEVENTCAPT will trigger a capture in TCAR1 on first enabled event and TCAR2 will update on the second enabled event.

**Figure 20-3. Capture Wave Example for CAPT\_MODE = 0**



**Figure 20-4. Capture Wave Example for CAPT\_MODE = 1**


#### 20.1.3.1.3 Compare Mode Functionality

When Compare Enable TCLR (CE bit) is set to 1, the timer value (TCRR) is permanently compared to the value held in timer match register (TMAR). TMAR value can be loaded at any time (timer counting or stop). When the TCRR and the TMAR values match, an interrupt can be issued if the IRQENABLE\_SET (MAT\_EN\_FLAG bit) is set. The right programming way is to write a compare value in TMAR register before setting TCLR (CE bit) to avoid any unwanted interrupt due to a reset value matching effect.

The dedicated output pin (PORTIMERPWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (TIMER clock duration) or to invert the current value (toggle mode) when an overflow and a match occur.

#### 20.1.3.1.4 Prescaler Functionality

A prescaler counter can be used to divide the timer counter input clock frequency. The prescaler is enabled when TCLR bit 5 is set (PRE). The 2<sup>n</sup> division ratio value (PTV) can be configured in the TCLR register. The prescaler counter is reset when the timer counter is stopped or reloaded on the fly.

**Table 20-6. Prescaler Functionality**

| Contexts                       | Prescaler Counter | Timer Counter |
|--------------------------------|-------------------|---------------|
| Overflow (when Auto-reload on) | Reset             | TLDR          |
| TCRR Write                     | Reset             | TCRR          |
| TTGR Write                     | Reset             | TLDR          |
| Stop                           | Reset             | Frozen        |

#### 20.1.3.1.5 Pulse-Width Modulation

The timer can be configured to provide a programmable pulse-width modulation (PORTIMERPWM) output. The PORTIMERPWM output pin can be configured to toggle on specified event. TCLR (TRG bits) determines on which register value the PORTIMERPWM pin toggles. Either overflow or match can be used to toggle the PORTIMERPWM pin, when a compare condition occurs.

In case of overflow and match mode, the match event will be ignored from the moment the mode was set-up until the first overflow event occurs (see [Figure 20-4](#)).

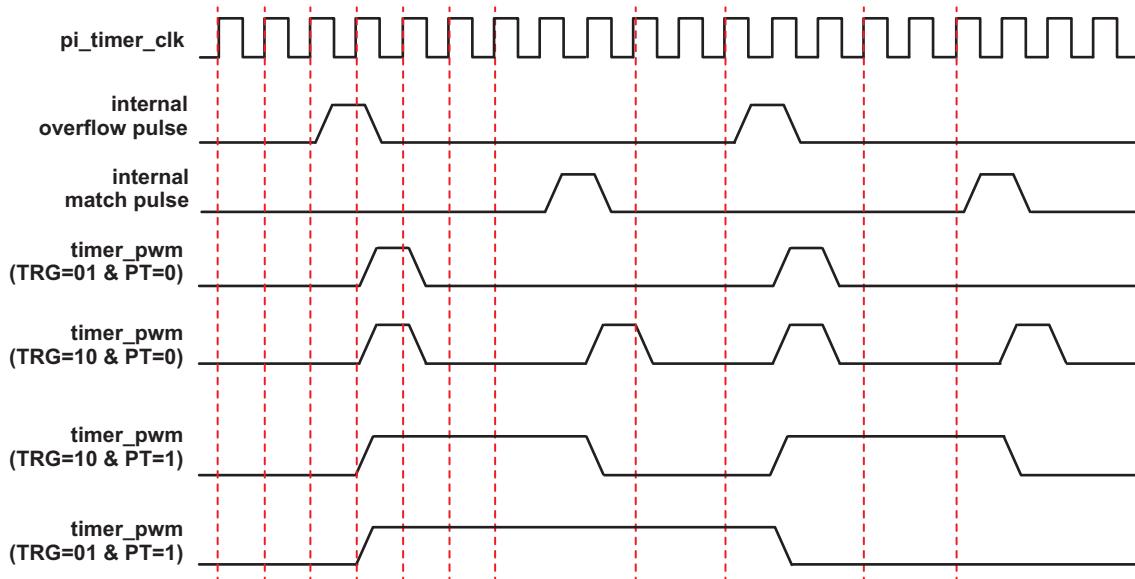
The TCLR (SCPWM bit) can be programmed to set or clear the PORTIMERPWM output signal while the counter is stopped or the triggering is off only. This allows fixing a deterministic state of the output pin when modulation is stopped. The modulation is synchronously stopped when TRG bit is cleared and overflow occurred.

In the following timing diagram, the internal overflow pulse is set each time (FFFF FFFFh – TLDR + 1) value is reached, and the internal match pulse is set when the counter reaches TMAR register value. According to TCLR (TRG and PT bits) programming value, the timer provides pulse or PWM on the output pin (PORTIMERPWM).

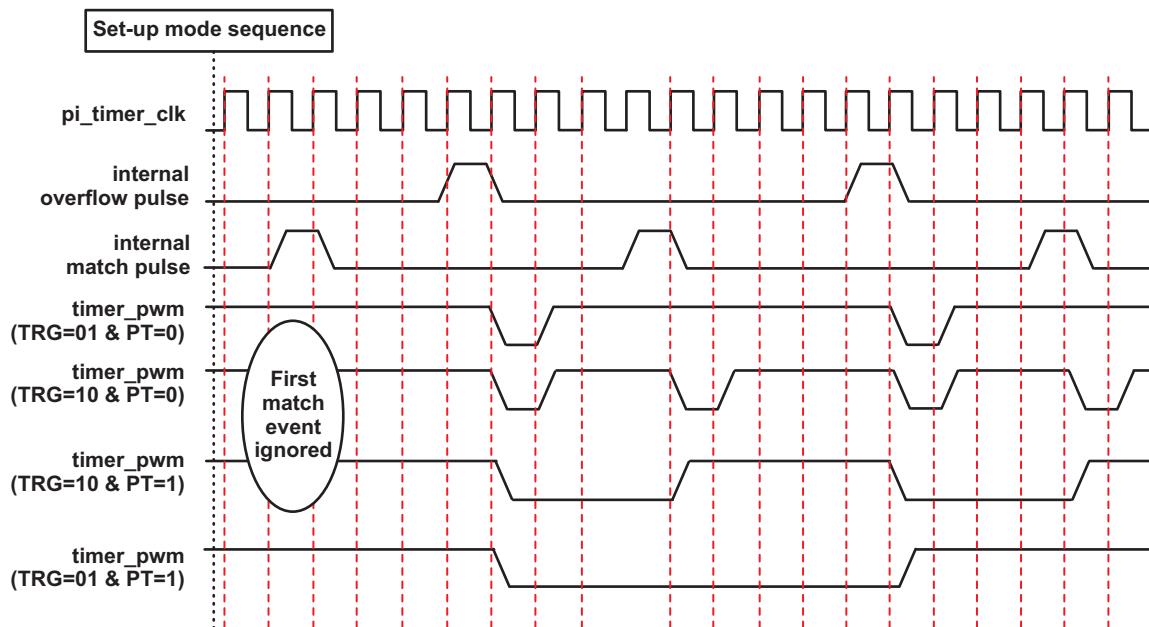
The TLDR and TMAR registers must keep values smaller than the overflow value (FFFF FFFFh) with at least 2 units. In case the PWM trigger events are both overflow and match, the difference between the values kept in TMAR register and the value in TLDR must be at least 2 units. When match event is used the compare mode TCLR (CE) must be set.

In [Figure 20-5](#), TCLR (SCPWM bit) is cleared to 0. In [Figure 20-6](#), TCLR (SCPWM bit) is set to 1.

**Figure 20-5. Timing Diagram of Pulse-Width Modulation with SCPWM = 0**



**Figure 20-6. Timing Diagram of Pulse-Width Modulation with SCPWM = 1**



### 20.1.3.1.6 Timer Counting Rate

The timer counter is composed of a prescaler stage and a timer counter. Prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage. The ratio can be managed by accessing the ratio definition field of the control register (PTV and PRE of TCLR). See [Table 20-7](#).

The timer rate is defined by:

- The value of the prescaler fields (PRE and PTV of TCLR register)
- The value loaded into the Timer Load Register (TLDR).

**Table 20-7. Prescaler Clock Ratios Value**

| PRE | PTV | Divisor (PS) |
|-----|-----|--------------|
| 0   | X   | 1            |
| 1   | 0   | 2            |
| 1   | 1   | 4            |
| 1   | 2   | 8            |
| 1   | 3   | 16           |
| 1   | 4   | 32           |
| 1   | 5   | 64           |
| 1   | 6   | 128          |
| 1   | 7   | 256          |

The timer rate equation is as follows:

$$(FFFF\ FFFFh - TLDR + 1) \times \text{timer Clock period} \times \text{Clock Divider (PS)}$$

With timer Clock period = 1/ timer Clock frequency and PS = 2(PTV + 1).

As an example, if we consider a timer clock input of 32 kHz, with a PRE field equal to 0, the timer output period is:

**Table 20-8. Value and Corresponding Interrupt Period**

| TLDR       | Interrupt period |
|------------|------------------|
| 0000 0000h | 37 h             |
| FFFF 0000h | 2 s              |
| FFFF FFF0h | 500 us           |
| FFFF FF0Eh | 62.5 us          |

### 20.1.3.1.7 Dual Mode Timer Under Emulation

During emulation mode (when PINSUSPENDN signal is active), the timer can/cannot continue running according to the value of the EmuFree bit of the timer OCP configuration register (TIOCP\_CFG).

If EmuFree is 1, timer execution is not stopped and, regardless of the value of PINSUSPENDN signal, and the interrupt assertion is still generated when overflow is reached.

If EmuFree is 0, counters (prescaler/timer) are frozen and an increment start occurs again as soon as PINSUSPENDN becomes inactive. The asynchronous input pin is internally synchronized on 2 TIMER clock rising edges.

### 20.1.3.2 Accessing Registers

All registers are 32-bit wide, accessible via OCP interface with 16-bit or 32-bit OCP access (Read/Write). The 32-bit registers write update in 16 bits access must be LSB16 first and the second write access must be MSB16. For the write operation, the module allows skipping the MSB access if the user does not need to update the 16 MSB bits of the register, but only for the OCP registers (TIDR, TIOCP\_CFG, IRQSTATUS\_RAW, IRQSTATUS, IRQENABLE\_SET, IRQENABLE\_CLR, IRQWAKEEN and TSICR). The write operation on any functional register (TCLR, TCRR, TLDR, TTGR and TMAR) must be complete (the MSB must be written even if the MSB data is not used).

#### 20.1.3.2.1 Programming the Timer Registers

The TLDR, TCRR, TCLR, TIOCP\_CFG, IRQSTATUS, IRQENABLE\_SET, IRQENABLE\_CLR, IRQWAKEEN, TTGR, TSICR and TMAR registers write is done synchronously with OCP clock, by the host, using the OCP bus protocol.

#### 20.1.3.2.2 Reading the Timer Registers

The counter register (TCRR) is a 32-bit “atomic datum” and 16-bit capture is done on the 16-bit LSB first to allow atomic LSB16 + MSB16 capture. Atomic capture is also performed for the TCAR1 and TCAR2 registers as they may change due to internal processes.

#### 20.1.3.2.3 OCP Error Generation

The timer module responds with error indication in the following cases:

##### Error on write transactions

- Assert the PORSRESP = ERR signal in the same cycle as PORSCMDACCEPTED.
- Use the ERR code for PORSRESP during the response phase.

##### Error on read transactions

- Assert the PORSRESP = ERR signal in the same cycle as PORSCMDACCEPTED.
- Use the ERR code for PORSRESP during the response phase. PORSDATA in this case is not valid.

**Table 20-9. OCP Error Reporting**

| Error Type  | Response: SRESP = ERR |
|---|-----------------------|
| Unsupported PIOCPMCMD command                                   | Yes                   |
| Address error: Read or write to a non-existing internal address | No                    |
| Read to write-only registers and write to read-only registers   | No                    |
| Unaligned address (PIOCPMADDR ≠ 00) on read/write transaction   | Yes                   |
| Unsupported PIOCPMBYTEEN on read/write transaction              | Yes                   |

**NOTE:** Byte enable “0000” is a supported byte enable.

### 20.1.3.3 Posted Mode Selection

A choice between the two synchronization modes will be made taking into account the frequency ratio and the stall periods that can be supported by the system, without impacting the global performance.

The posted mode selection applies only to functional registers that require synchronization on/from timer clock domain. For write operation the registers affected by this posted/non-posted selection are: TCLR, TLDR, TCRR, TTGR and TMAR. For read operation the register affected by this posted/non-posted selection are: TCRR, TCAR1 and TCAR2.

The OCP clock domain synchronous registers TIDR, TIOCP\_CFG, TISTAT, IRQSTATUS, IRQSTATUS\_RAW, IRQENABLE\_SET, IRQENABLE\_CLR, IRQWAKEEN, TWPS and TSICR are not affected by the posted/non-posted mode selection; the write/read operation is effective and acknowledged (command accepted) after one OCP clock cycle from the command assertion.

The configuration posted/non-posted is made by setting the PIFREQRATIO when the module is integrated. The PIFREQRATIO signal should be tied to '1' when the freq (timer) < freq (OCP)/4 frequency, and tied to 0 when it is the opposite frequency ratio. The PIFREQRATIO represent the reset value of the TSICR (POSTED bit). The configuration can be changed (overwritten) by software, writing the TSICR (POSTED bit) register.

The following cases are possible:

- Posted Mode can be used when the functional frequency range is: freq (timer) < freq (OCP)/4.
- Non-Posted Mode can be used regardless of the frequency range. Recommended frequency is: freq (timer) >= freq (OCP)/4.

---

**NOTE:** The Non-Posted Mode can be also used when freq (timer) < freq (OCP)/4, but it is recommended to use the Posted Mode. Using Non-Posted Mode will delay the command accept and the transaction latency will be as described in the below chapters. Posted mode offers an OCP interface latency improvement and can be used only if the frequencies respect the freq (timer) < freq (OCP)/4 formula.

---

## 20.1.3.4 Write Registers Access

### 20.1.3.4.1 Write Posted

This mode can be used only if the functional frequency range is freq (timer) < freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is set to 1 in the timer control register.

This mode uses a posted-write scheme to update any internal register. The write transaction is immediately acknowledged on the OCP interface, although the effective write operation will occur later, due to a resynchronisation in the timer clock domain. This has the advantage of not stalling either the interconnect system, or the CPU that requested the write transaction. For each register, a status bit is provided, that is set if there is a pending write access to this register.

In this mode, it is mandatory that the CPU checks the status bit prior to any write access. In case a write is attempted to a register with a previous access pending, the previous access is discarded without notice (this can lead to unexpected results also).

There is one status bit per register, accessible in the Timer Write Posted Status Register. When the timer module operates in this mode, there is an automatic sampling of the current timer counter value, in an OCP-synchronized capture register. Consequently, any read access to the timer counter register does not add any re-synchronization latency; the current value is always available.

A register read following a write posted register (on the same register) is not insured to read the previous write value if the write posted process is not completed. Software synchronization should be used to avoid non-coherent read.

The drawback of this automatic update mechanism is that it assumes a given relationship between the OCP interface frequency and the timer functional frequency.

This posted period is defined as the interval between the posted write access request and the reset of the posted bit in TWPS register, and can be quantified:

$$T(\text{reset posted max.}) = 3 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

The time when the write accomplishes is:

$$T(\text{write accomplish}) = 1 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

#### 20.1.3.4.2 Write Non-Posted

This mode is functional regardless of the ratio between the OCP interface frequency and the functional clock frequency. Recommended functional frequency range is freq (timer)  $\geq$  freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is cleared to 0 in the timer control register.

This mode uses a non posted-write scheme to update any internal register. That means the write transaction will not be acknowledged on the OCP interface, until the effective write operation occurs, after the resynchronisation in the timer clock domain. The drawback is that both the interconnect system and the CPU are stalled during this period.

- The latency of the interrupt serving is increased, as the interconnect system and CPU are stalled.
- An interconnect logic, including time-out logic to detect erroneous transactions, can generate an unwanted system abort event.

The stall period is defined as the interval between the non-posted write access request and the rise of the command accept signal and can be quantified:

$$T(\text{stall max.}) = 3 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

The time when the write accomplishes is:

$$T(\text{write accomplish}) = 1 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

A register read following a write to the same register is always coherent.

#### 20.1.3.5 Read Registers Access

##### 20.1.3.5.1 Read Posted

This mode can be used only if the functional frequency range is freq (timer)  $<$  freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is set to 1 in the timer control register.

This mode uses a posted-read scheme, for reading any internal register. The read transaction is immediately acknowledged on the OCP interface, and the value to be read has been previously resynchronised. This has the advantage of not stalling either the interconnect system, or the CPU that requested the read transaction.

##### 20.1.3.5.2 Read Non-Posted

This mode is functional whatever the ratio between the OCP interface frequency and the functional clock frequency. Recommended functional frequency range is freq (timer)  $\geq$  freq (OCP)/4.

This mode is used if TSICR (POSTED bit) is cleared to 0 in the timer control register.

This mode uses a non posted-read scheme, for reading any internal register. The read transaction will not be acknowledged on the OCP interface, until the effective read operation occurs, after the resynchronisation in the timer clock domain. The drawback is that both the interconnect system and the CPU are stalled during this period.

- The latency of the interrupt serving is increased, as the interconnect system and the CPU are stalled.
- An interconnect system including time-out logic to detect erroneous transactions can generate an unwanted system abort event.

This mode only applies only to three registers: TCRR, TCAR1 and TCAR2, which need resynchronisation from functional to OCP clock domains.

The stall period is defined as the interval between the non-posted read access request and the rise of the command accept signal and can be quantified:

$$T(\text{stall max.}) = 3 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

The time when the value is sampled is:

$$T(\text{read sample}) = 1 \text{ OCP clock} + 2.5 \text{ TIMER clock}$$

## 20.1.4 Use Cases

### 20.1.5 TIMER Registers

Table 20-10 lists the memory-mapped registers for the TIMER. All register offset addresses not listed in Table 20-10 should be considered as reserved locations and the register contents should not be modified.

**Table 20-10. TIMER REGISTERS**

| Offset | Acronym | Register Name                                | Section                           |
|--------|---------|--|-----------------------------------|
| 00h    |         | Identification Register                      | <a href="#">Section 20.1.5.1</a>  |
| 10h    |         | Timer OCP Configuration Register             | <a href="#">Section 20.1.5.2</a>  |
| 24h    |         | Timer IRQSTATUS Raw Register                 | <a href="#">Section 20.1.5.3</a>  |
| 28h    |         | Timer IRQSTATUS Register                     | <a href="#">Section 20.1.5.4</a>  |
| 2Ch    |         | Timer IRQENABLE Set Register                 | <a href="#">Section 20.1.5.5</a>  |
| 30h    |         | Timer IRQENABLE Clear Register               | <a href="#">Section 20.1.5.6</a>  |
| 34h    |         | Timer IRQ Wakeup Enable Register             | <a href="#">Section 20.1.5.7</a>  |
| 38h    |         | Timer Control Register                       | <a href="#">Section 20.1.5.8</a>  |
| 3Ch    |         | Timer Counter Register                       | <a href="#">Section 20.1.5.9</a>  |
| 40h    |         | Timer Load Register                          | <a href="#">Section 20.1.5.10</a> |
| 44h    |         | Timer Trigger Register                       | <a href="#">Section 20.1.5.11</a> |
| 48h    |         | Timer Write Posted Status Register           | <a href="#">Section 20.1.5.12</a> |
| 4Ch    |         | Timer Match Register                         | <a href="#">Section 20.1.5.13</a> |
| 50h    |         | Timer Capture Register                       | <a href="#">Section 20.1.5.14</a> |
| 54h    |         | Timer Synchronous Interface Control Register | <a href="#">Section 20.1.5.15</a> |
| 58h    |         | Timer Capture Register                       | <a href="#">Section 20.1.5.16</a> |

### 20.1.5.1 Identification Register Register (offset = 00h) [reset = 40000100h]

Identification Register is shown in [Figure 20-7](#) and described in [Table 20-11](#).

This read only register contains the revision number of the module. A write to this register has no effect. This register is used by software to track features, bugs, and compatibility.

**Figure 20-7. Identification Register Register**

| 31     | 30     | 29     | 28       | 27      | 26     | 25   | 24 |
|--------|--------|--------|----------|---------|--------|------|----|
|        | SCHEME |        | Reserved |         |        | FUNC |    |
| R/W-1h |        | R-0h   |          |         | R/W-0h |      |    |
| 23     | 22     | 21     | 20       | 19      | 18     | 17   | 16 |
|        |        |        |          | FUNC    |        |      |    |
|        |        |        |          | R/W-0h  |        |      |    |
| 15     | 14     | 13     | 12       | 11      | 10     | 9    | 8  |
|        |        | R RTL  |          | X MAJOR |        |      |    |
|        |        | R/W-0h |          | R/W-1h  |        |      |    |
| 7      | 6      | 5      | 4        | 3       | 2      | 1    | 0  |
|        | CUSTOM |        |          | Y MINOR |        |      |    |
|        | R/W-0h |        |          | R/W-0h  |        |      |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-11. Identification Register Register Field Descriptions**

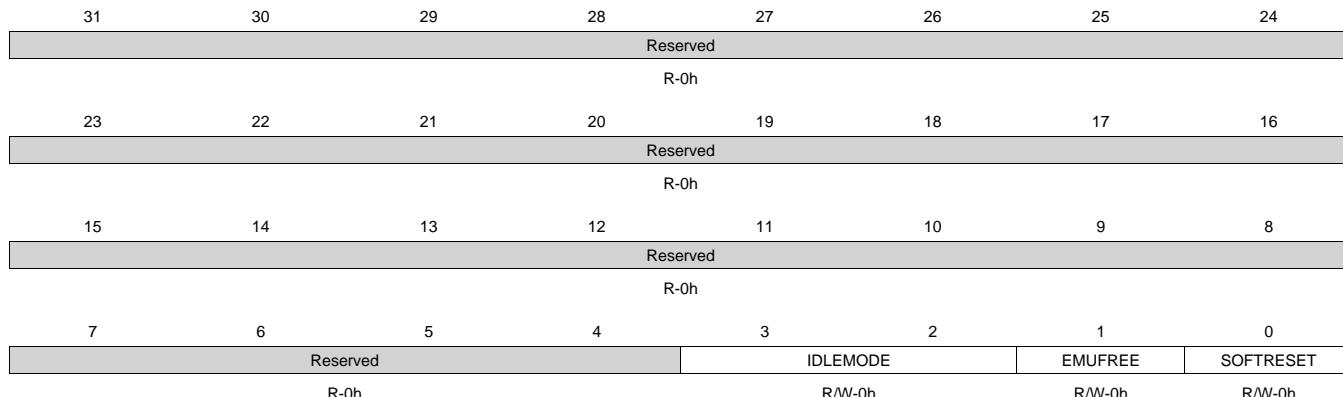
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-30 | SCHEME   | R/W  | 1h    | Used to distinguish between old scheme and current.     |
| 29-28 | Reserved | R    | 0h    |   |
| 27-16 | FUNC     | R/W  | 0h    | Function indicates a software compatible module family. |
| 15-11 | R RTL    | R/W  | 0h    | RTL Version (R).  |
| 10-8  | X MAJOR  | R/W  | 1h    | Major Revision (X).                                     |
| 7-6   | CUSTOM   | R/W  | 0h    | Indicates a special version for a particular device.    |
| 5-0   | Y MINOR  | R/W  | 0h    | Minor Revision (Y).                                     |

### 20.1.5.2 Timer OCP Configuration Register Register (offset = 10h) [reset = 0h]

Timer OCP Configuration Register is shown in [Figure 20-8](#) and described in [Table 20-12](#).

This register allows controlling various parameters of the OCP interface.

**Figure 20-8. Timer OCP Configuration Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-12. Timer OCP Configuration Register Register Field Descriptions**

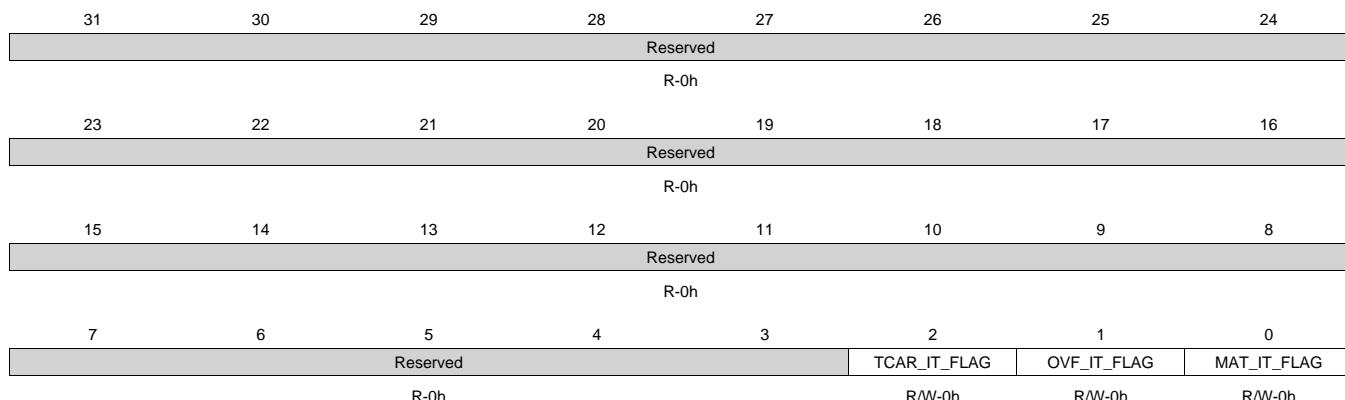
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-4 | Reserved  | R    | 0h    |  |
| 3-2  | IDLEMODE  | R/W  | 0h    | Power management, req/ack control<br>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only.<br>0x1 = No-idle mode: local target never enters idle state. Backup mode, for debug only.<br>0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events.<br>0x3 = Reserved. |
| 1    | EMUFREE   | R/W  | 0h    | Emulation mode<br>0x0 = The timer is frozen in emulation mode (PINSUSPENDN signal active).<br>0x1 = The timer runs free, regardless of PINSUSPENDN value.  |
| 0    | SOFTRESET | R/W  | 0h    | Software reset.<br>0x0(W) = No action.<br>0x0(R) = Reset done, no pending action.<br>0x1(W) = Initiate software reset.<br>0x1(R) = Reset ongoing.  |

### 20.1.5.3 Timer IRQSTATUS Raw Register Register (offset = 24h) [reset = 0h]

Timer IRQSTATUS Raw Register is shown in [Figure 20-9](#) and described in [Table 20-13](#).

Component interrupt request status. Check the corresponding secondary status register. Raw status is set even if event is not enabled. Write 1 to set the (raw) status, mostly for debug.

**Figure 20-9. Timer IRQSTATUS Raw Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-13. Timer IRQSTATUS Raw Register Register Field Descriptions**

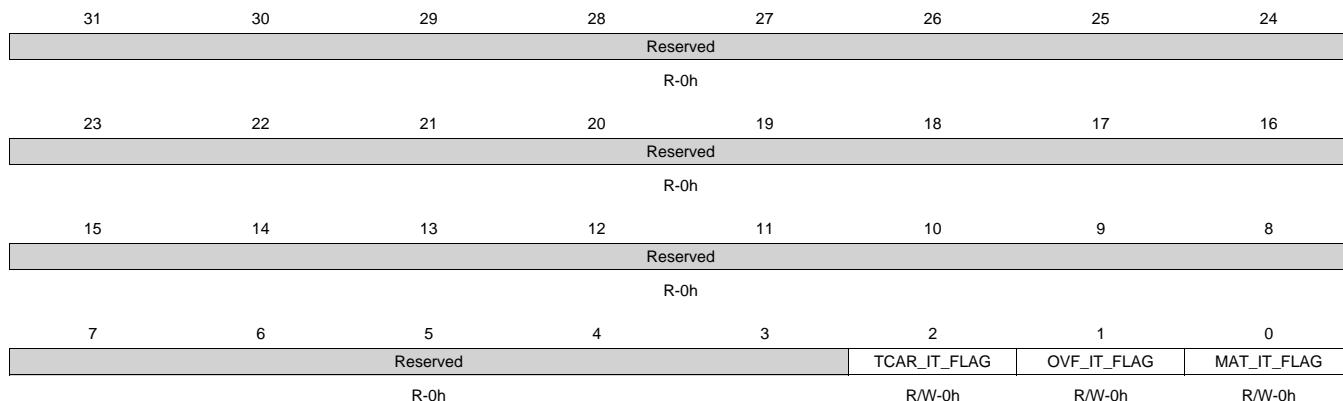
| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-3 | Reserved     | R    | 0h    |  |
| 2    | TCAR_IT_FLAG | R/W  | 0h    | IRQ status for Capture<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Trigger IRQ event by software<br>0x1(R) = IRQ event pending  |
| 1    | OVF_IT_FLAG  | R/W  | 0h    | IRQ status for Overflow<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Trigger IRQ event by software<br>0x1(R) = IRQ event pending |
| 0    | MAT_IT_FLAG  | R/W  | 0h    | IRQ status for Match<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Trigger IRQ event by software<br>0x1(R) = IRQ event pending    |

#### 20.1.5.4 Timer IRQSTATUS Register Register (offset = 28h) [reset = 0h]

Timer IRQSTATUS Register is shown in [Figure 20-10](#) and described in [Table 20-14](#).

Component interrupt request status. Check the corresponding secondary status register. Enabled status is not set unless event is enabled. Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, that is, even if not enabled).

**Figure 20-10. Timer IRQSTATUS Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-14. Timer IRQSTATUS Register Register Field Descriptions**

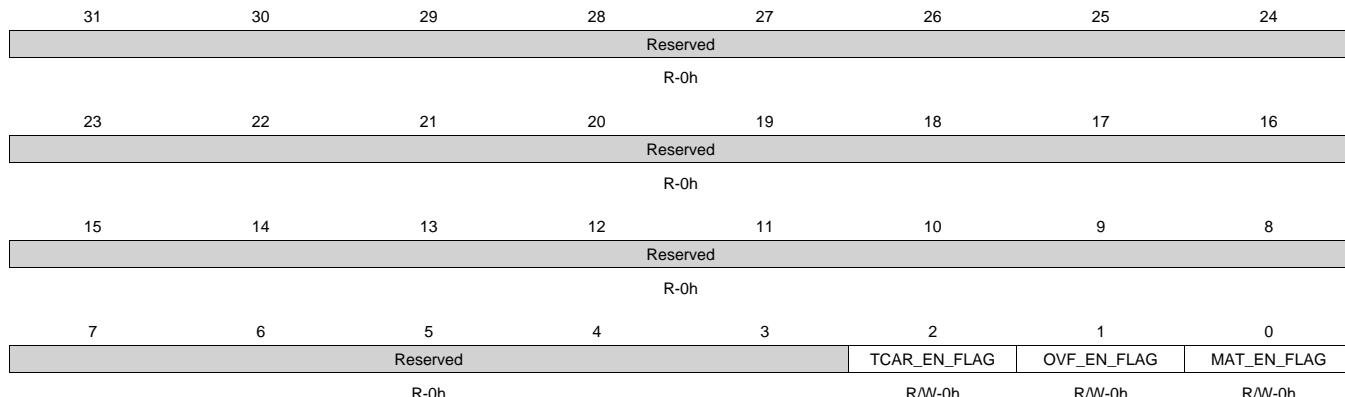
| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-3 | Reserved     | R    | 0h    |  |
| 2    | TCAR_IT_FLAG | R/W  | 0h    | IRQ status for Capture<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Clear pending event, if any<br>0x1(R) = IRQ event pending  |
| 1    | OVF_IT_FLAG  | R/W  | 0h    | IRQ status for Overflow<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Clear pending event, if any<br>0x1(R) = IRQ event pending |
| 0    | MAT_IT_FLAG  | R/W  | 0h    | IRQ status for Match<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Clear pending event, if any<br>0x1(R) = IRQ event pending    |

### 20.1.5.5 Timer IRQENABLE Set Register Register (offset = 2Ch) [reset = 0h]

Timer IRQENABLE Set Register is shown in [Figure 20-11](#) and described in [Table 20-15](#).

Component interrupt request enable. Write 1 to set (enable interrupt). Readout equal to corresponding \_CLR register.

**Figure 20-11. Timer IRQENABLE Set Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-15. Timer IRQENABLE Set Register Register Field Descriptions**

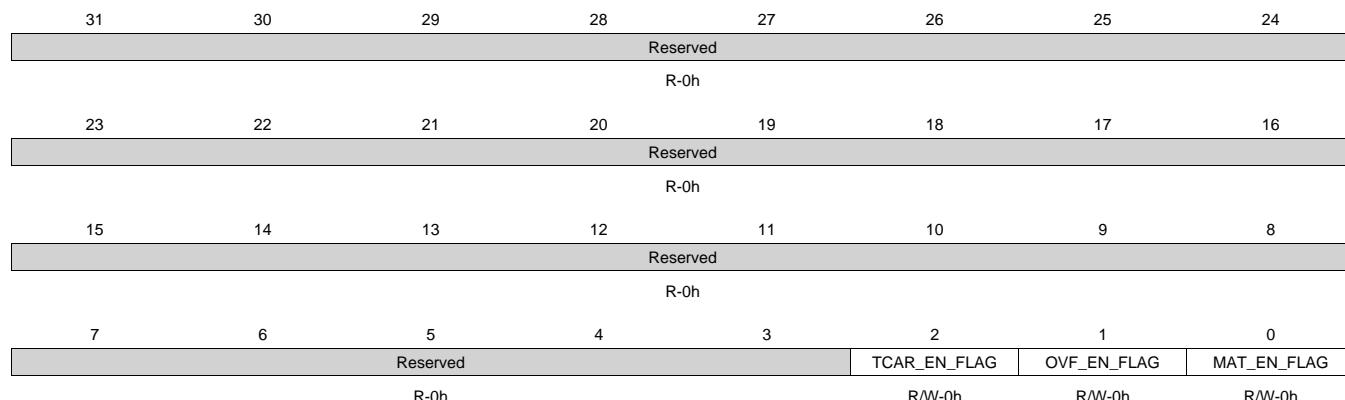
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-3 | Reserved     | R    | 0h    |   |
| 2    | TCAR_EN_FLAG | R/W  | 0h    | IRQ enable for Compare<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Set IRQ enable<br>0x1(R) = IRQ event is enabled  |
| 1    | OVF_EN_FLAG  | R/W  | 0h    | IRQ enable for Overflow<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Set IRQ enable<br>0x1(R) = IRQ event is enabled |
| 0    | MAT_EN_FLAG  | R/W  | 0h    | IRQ enable for Match<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Set IRQ enable<br>0x1(R) = IRQ event is enabled    |

### 20.1.5.6 Timer IRQENABLE Clear Register Register (offset = 30h) [reset = 0h]

Timer IRQENABLE Clear Register is shown in [Figure 20-12](#) and described in [Table 20-16](#).

Component interrupt request enable. Write 1 to clear (disable interrupt). Readout equal to corresponding \_SET register.

**Figure 20-12. Timer IRQENABLE Clear Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-16. Timer IRQENABLE Clear Register Register Field Descriptions**

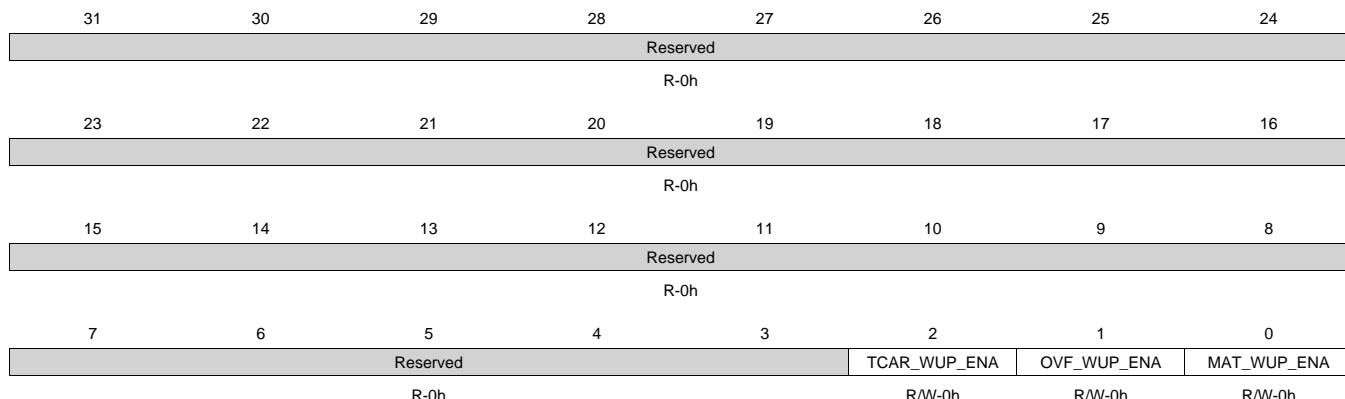
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-3 | Reserved     | R    | 0h    |   |
| 2    | TCAR_EN_FLAG | R/W  | 0h    | IRQ enable for Compare<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Clear IRQ enable<br>0x1(R) = IRQ event is enabled  |
| 1    | OVF_EN_FLAG  | R/W  | 0h    | IRQ enable for Overflow<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Clear IRQ enable<br>0x1(R) = IRQ event is enabled |
| 0    | MAT_EN_FLAG  | R/W  | 0h    | IRQ enable for Match<br>0x0(W) = No action<br>0x0(R) = IRQ event is disabled<br>0x1(W) = Clear IRQ enable<br>0x1(R) = IRQ event is enabled    |

### 20.1.5.7 Timer IRQ Wakeup Enable Register Register (offset = 34h) [reset = 0h]

Timer IRQ Wakeup Enable Register is shown in [Figure 20-13](#) and described in [Table 20-17](#).

Wakeup-enabled events taking place when module is idle will generate an asynchronous wakeup.

**Figure 20-13. Timer IRQ Wakeup Enable Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-17. Timer IRQ Wakeup Enable Register Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-3 | Reserved     | R    | 0h    |   |
| 2    | TCAR_WUP_ENA | R/W  | 0h    | Wakeup generation for Compare<br>0x0 = Wakeup disabled<br>0x1 = Wakeup enabled  |
| 1    | OVF_WUP_ENA  | R/W  | 0h    | Wakeup generation for Overflow<br>0x0 = Wakeup disabled<br>0x1 = Wakeup enabled |
| 0    | MAT_WUP_ENA  | R/W  | 0h    | Wakeup generation for Match<br>0x0 = Wakeup disabled<br>0x1 = Wakeup enabled    |

### 20.1.5.8 Timer Control Register Register (offset = 38h) [reset = 0h]

Timer Control Register is shown in [Figure 20-14](#) and described in [Table 20-18](#).

When the TCM field passed from (00) to any other combination then the TCAR\_IT\_FLAG and the edge detection logic are cleared. The ST bit of TCLR register may be updated from the OCP interface or reset due to one-shot overflow. The OCP interface update has the priority.

**Figure 20-14. Timer Control Register Register**

|          |         |           |        |        |        |        |    |
|----------|---------|-----------|--------|--------|--------|--------|----|
| 31       | 30      | 29        | 28     | 27     | 26     | 25     | 24 |
| Reserved |         |           |        |        |        |        |    |
| R-0h     |         |           |        |        |        |        |    |
| 23       | 22      | 21        | 20     | 19     | 18     | 17     | 16 |
| Reserved |         |           |        |        |        |        |    |
| R-0h     |         |           |        |        |        |        |    |
| 15       | 14      | 13        | 12     | 11     | 10     | 9      | 8  |
| Reserved | GPO_CFG | CAPT_MODE | PT     | TRG    | TCM    |        |    |
| R-0h     | R/W-0h  | R/W-0h    | R/W-0h | R/W-0h | R/W-0h |        |    |
| 7        | 6       | 5         | 4      | 3      | 2      | 1      | 0  |
| SCPWM    | CE      | PRE       |        | PTV    | AR     | ST     |    |
| R/W-0h   | R/W-0h  | R/W-0h    |        | R/W-0h | R/W-0h | R/W-0h |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-18. Timer Control Register Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-15 | Reserved  | R    | 0h    |  |
| 14    | GPO_CFG   | R/W  | 0h    | General purpose output this register drives directly the PORPGPOCFG output pin<br>0x0 = PORPGPOCFG drives 0<br>0x1 = PORPGPOCFG drives 1   |
| 13    | CAPT_MODE | R/W  | 0h    | Capture mode.<br>0x0 = Single capture<br>0x1 = Capture on second event   |
| 12    | PT        | R/W  | 0h    | Pulse or toggle mode on PORTIMERPWM output pin<br>0x0 = Pulse<br>0x1 = Toggle  |
| 11-10 | TRG       | R/W  | 0h    | Trigger output mode on PORTIMERPWM output pin<br>0x0 = No trigger<br>0x1 = Trigger on overflow<br>0x2 = Trigger on overflow and match<br>0x3 = Reserved  |
| 9-8   | TCM       | R/W  | 0h    | Transition Capture Mode on PIEVENTCAPT input pin<br>0x0 = No capture<br>0x1 = Capture on low to high transition<br>0x2 = Capture on high to low transition<br>0x3 = Capture on both edge transition  |
| 7     | SCPWM     | R/W  | 0h    | This bit should be set or clear while the timer is stopped or the trigger is off<br>0x0 = Clear the PORTIMERPWM output pin and select positive pulse for pulse mode<br>0x1 = Set the PORTIMERPWM output pin and select negative pulse for pulse mode |
| 6     | CE        | R/W  | 0h    | 0x0 = Compare mode is disabled<br>0x1 = Compare mode is enabled  |

**Table 20-18. Timer Control Register Register Field Descriptions (continued)**

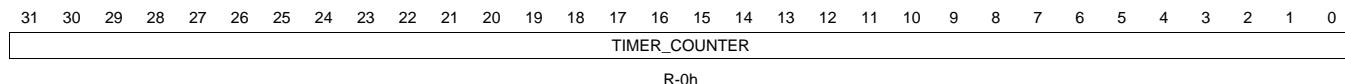
| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 5   | PRE   | R/W  | 0h    | Prescaler enable<br>0x0 = The TIMER clock input pin clocks the counter<br>0x1 = The divided input pin clocks the counter  |
| 4-2 | PTV   | R/W  | 0h    | Pre-scale clock Timer value   |
| 1   | AR    | R/W  | 0h    | 0x0 = One shot timer<br>0x1 = Auto-reload timer   |
| 0   | ST    | R/W  | 0h    | In the case of one-shot mode selected (AR = 0), this bit is automatically reset by internal logic when the counter is overflowed.<br>0x0(Read) = Stop timeOnly the counter is frozen<br>0x1 = Start timer |

### 20.1.5.9 Timer Counter Register Register (offset = 3Ch) [reset = 0h]

Timer Counter Register is shown in [Figure 20-15](#) and described in [Table 20-19](#).

The TCRR register is a 32-bit register, 16-bit addressable. The MCU can perform a 32-bit access or two 16-bit accesses to access the register. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCRR value is not read while it is being incremented. In 16-bit mode the following sequence must be followed to read the TCRR register properly: First, perform an OCP Read Transaction to Read the lower 16-bit of the TCRR register (offset = 28h). When the TCRR is read and synchronized, the lower 16-bit LSBs are driven onto the output OCP data bus and the upper 16-bit MSBs of the TCRR register are stored in a temporary register. Second, perform an OCP Read Transaction to read the upper 16-bit of the TCRR register (offset = 2Ah). During this Read, the value of the upper 16-bit MSBs that has been temporary register is forwarded onto the output OCP data bus. So, to read the value of TCRR correctly, the first OCP read access has to be to the lower 16-bit (offset = 28h), followed by OCP read access to the upper 16-bit (offset = 2Ah). As the TCRR is updated using more sources (shadow\_in\_tcrr, incremented value of tcrr, TLDR and 0 ), a priority order will be defined: The first priority is the OCP update. The second is the reload way (triggered through TTGR reg. or following an auto-reload overflow). The third is the one-shot overflow reset to 0. The last is the incremented value.

**Figure 20-15. Timer Counter Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-19. Timer Counter Register Register Field Descriptions**

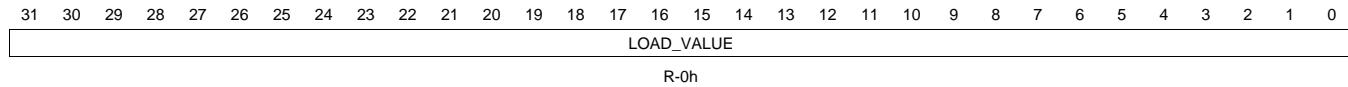
| Bit  | Field         | Type | Reset | Description            |
|------|---------------|------|-------|------------------------|
| 31-0 | TIMER_COUNTER | R    | 0h    | Value of TIMER counter |

### 20.1.5.10 Timer Load Register Register (offset = 40h) [reset = 0h]

Timer Load Register is shown in [Figure 20-16](#) and described in [Table 20-20](#).

LOAD\_VALUE must be different than the timer overflow value (FFFF FFFFh).

**Figure 20-16. Timer Load Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-20. Timer Load Register Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | LOAD_VALUE | R    | 0h    | Timer counter value loaded on overflow in auto-reload mode or on TTGR write access |

### 20.1.5.11 Timer Trigger Register Register (offset = 44h) [reset = FFFFFFFFh]

Timer Trigger Register is shown in [Figure 20-17](#) and described in [Table 20-21](#).

The read value of this register is always FFFF FFFFh.

**Figure 20-17. Timer Trigger Register Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TTGR_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-FFFFFFFh |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-21. Timer Trigger Register Register Field Descriptions**

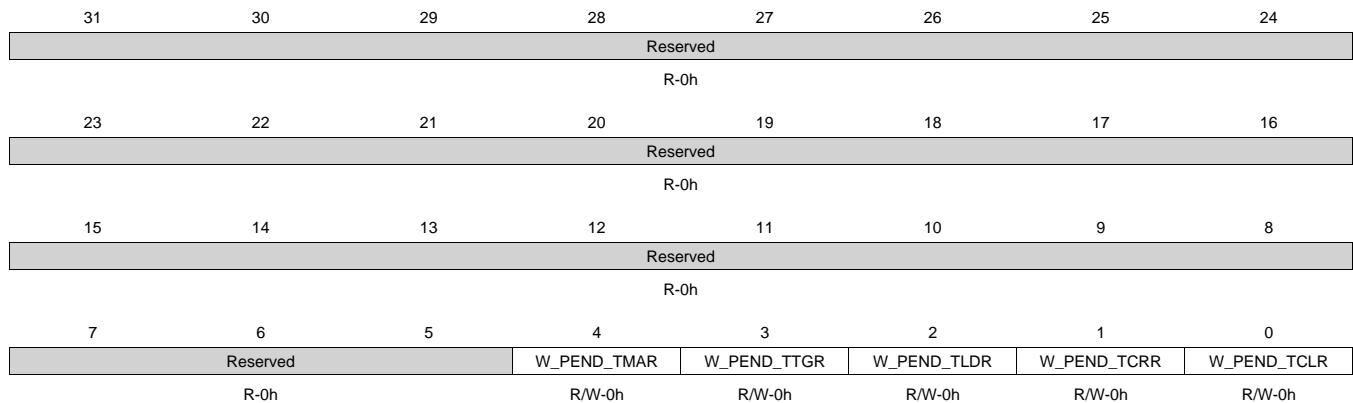
| Bit  | Field      | Type | Reset    | Description  |
|------|------------|------|----------|--|
| 31-0 | TTGR_VALUE | R    | FFFFFFFh | Writing in the TTGR register, TCRR will be loaded from TLDR and prescaler counter will be cleared.<br>Reload will be done regardless of the AR field value of TCLR register. |

### 20.1.5.12 Timer Write Posted Status Register Register (offset = 48h) [reset = 0h]

Timer Write Posted Status Register is shown in [Figure 20-18](#) and described in [Table 20-22](#).

In posted mode the software must read the pending write status bits (Timer Write Posted Status register bits [4:0]) to insure that following write access will not be discarded due to on going write synchronization process. These bits are automatically cleared by internal logic when the write to the corresponding register is acknowledged.

**Figure 20-18. Timer Write Posted Status Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-22. Timer Write Posted Status Register Register Field Descriptions**

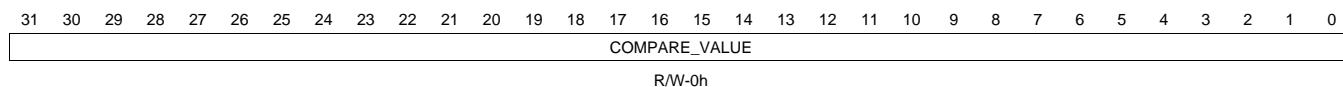
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-5 | Reserved    | R    | 0h    |  |
| 4    | W_PEND_TMAR | R/W  | 0h    | When equal to 1, a write is pending to the TMAR register |
| 3    | W_PEND_TTGR | R/W  | 0h    | When equal to 1, a write is pending to the TTGR register |
| 2    | W_PEND_TLDR | R/W  | 0h    | When equal to 1, a write is pending to the TLDR register |
| 1    | W_PEND_TCRR | R/W  | 0h    | When equal to 1, a write is pending to the TCRR register |
| 0    | W_PEND_TCLR | R/W  | 0h    | When equal to 1, a write is pending to the TCLR register |

### 20.1.5.13 Timer Match Register Register (offset = 4Ch) [reset = 0h]

Timer Match Register is shown in [Figure 20-19](#) and described in [Table 20-23](#).

The compare logic consists of a 32-bit wide, read/write data TMAR register and logic to compare counter s current value with the value stored in the TMAR register.

**Figure 20-19. Timer Match Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-23. Timer Match Register Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description                               |
|------|---------------|------|-------|---|
| 31-0 | COMPARE_VALUE | R/W  | 0h    | Value to be compared to the timer counter |

### 20.1.5.14 Timer Capture Register Register (offset = 50h) [reset = 0h]

Timer Capture Register is shown in [Figure 20-20](#) and described in [Table 20-24](#).

When the appropriate (rising, falling or both) transition is detected in the edge detection logic the current counter value is stored to the TCAR1 register. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCAR1 value is not read while it is being updated due to some capture event. In 16-bit mode the following sequence must be followed to read the TCAR1 register properly:

**Figure 20-20. Timer Capture Register Register**

|                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31             | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAPTURED_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-24. Timer Capture Register Register Field Descriptions**

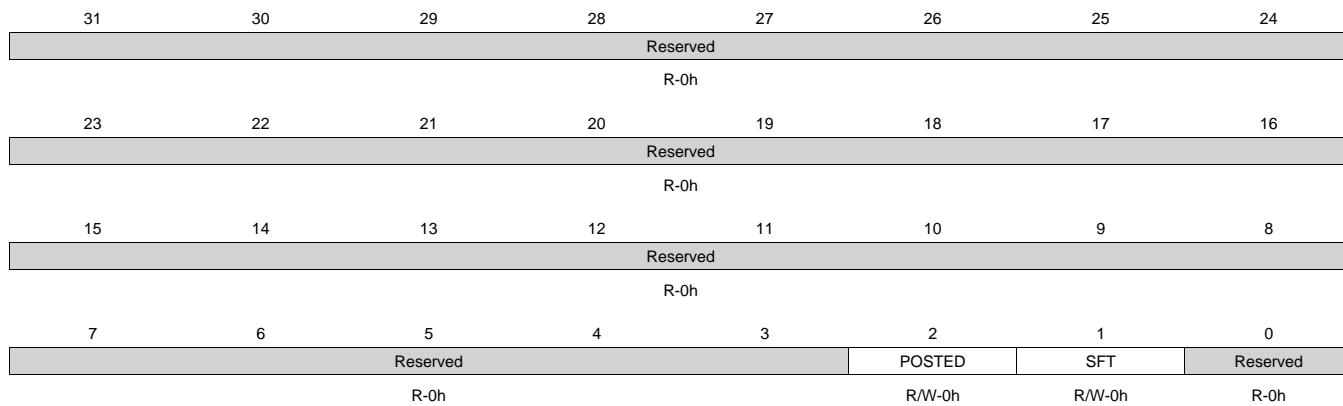
| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-0 | CAPTURED_VALUE | R/W  | 0h    | Timer counter value captured on an external event trigger |

### 20.1.5.15 Timer Synchronous Interface Control Register Register (offset = 54h) [reset = 0h]

Timer Synchronous Interface Control Register is shown in [Figure 20-21](#) and described in [Table 20-25](#).

Access to this register is not stalled even if the timer is in non-posted mode configuration. To abort any wrong behavior, software can permanently reset the functional part of the module. Also in case of a wrong hardware PIFREQRATIO tied the POSTED field can be reprogrammed on the fly, so deadlock situation cannot happen. Reset value of POSTED depends on hardware integration module at design time. Software must read POSTED field to get the hardware module configuration.

**Figure 20-21. Timer Synchronous Interface Control Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-25. Timer Synchronous Interface Control Register Register Field Descriptions**

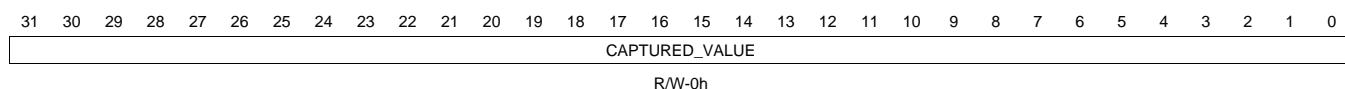
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-3 | Reserved | R    | 0h    |   |
| 2    | POSTED   | R/W  | 0h    | PIFREQRATIO<br>0x0 = Posted mode inactive: will delay the command accept output signal.<br>0x1 = Posted mode active (clocks ratio needs to fit freq (timer) less than freq (OCP)/4 frequency requirement) |
| 1    | SFT      | R/W  | 0h    | This bit resets all the function parts of the module.<br>During reads it always returns 0.<br>0x0 = Software reset is enabled<br>0x1 = Software reset is disabled   |
| 0    | Reserved | R    | 0h    |   |

### 20.1.5.16 Timer Capture Register Register (offset = 58h) [reset = 0h]

Timer Capture Register is shown in [Figure 20-22](#) and described in [Table 20-26](#).

When the appropriate (rising, falling or both) transition is detected in the edge detection logic and the capture on second event is activated from the control register (TCLR), the current counter value is stored to the TCAR2 register. Note that since the OCP clock is completely asynchronous with the timer clock, some synchronization is done in order to make sure that the TCAR2 value is not read while it is being updated due to some capture event. In 16-bit mode the following sequence must be followed to read the TCAR2 register properly: First, perform an OCP Read Transaction to Read the lower 16-bits of the TCAR2 register. Second, perform an OCP Read Transaction to read the upper 16-bits of the TCAR2 register.

**Figure 20-22. Timer Capture Register Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-26. Timer Capture Register Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-0 | CAPTURED_VALUE | R/W  | 0h    | Timer counter value captured on an external event trigger |

## 20.2 DMTimer 1ms

### 20.2.1 Introduction

This peripheral is a 32-bit timer offering:

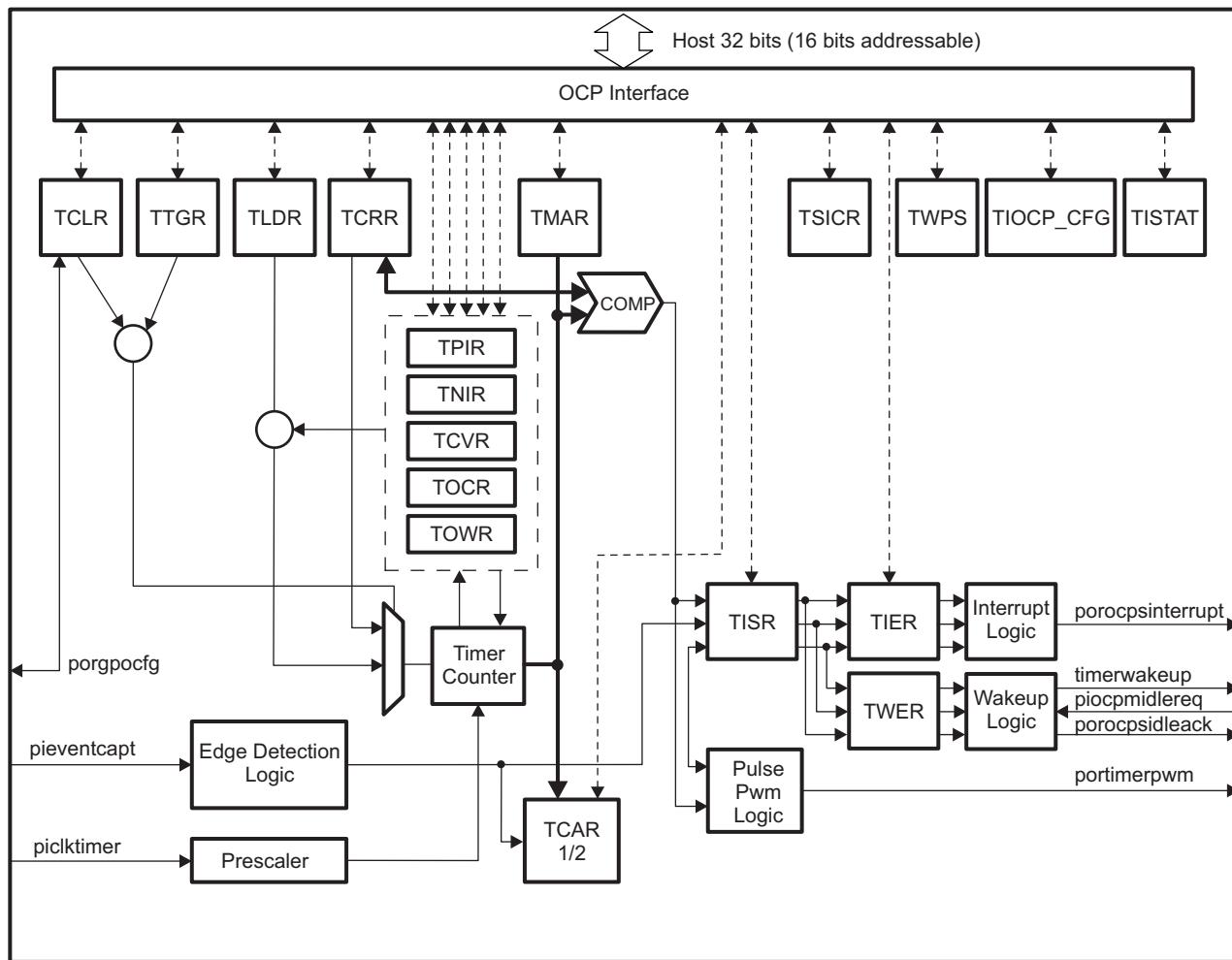
- Counter timer with compare and capture modes
- Auto-reload mode
- Start-stop mode
- Generate 1 ms tick with 32768-Hz functional clock
- Programmable divider clock source
- 16–32 bit addressing
- On-the-fly read/write registers
- Interrupts generated on overflow, compare and capture
- Interrupt enable
- Wake-up enable
- Write posted mode
- Dedicated input trigger for capture mode and dedicated output trigger/PWM signal
- Dedicated output signal for general purpose use PORGPOCFG
- OCP interface compatible

The timer module contains a free running upward counter with auto reload capability on overflow. The timer counter can be read and written on the fly (while counting). The timer module includes compare logic to allow interrupt event on programmable counter matching value.

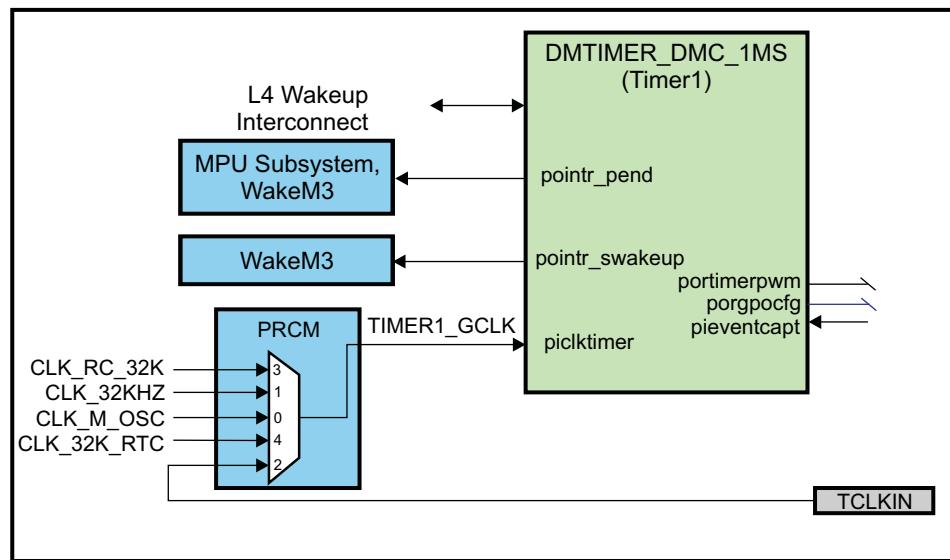
A dedicated output signal can be pulsed or toggled on overflow and match event. This output offers timing stamp trigger signal or PWM (pulse width modulation) signal sources. A dedicated output signal can be used for general purpose PORGPOCFG (directly driven by the bit 14 of the TCLR register). A dedicated input signal is used to trigger automatic timer counter capture and interrupt event, on programmable input signal transition type. A programmable clock divider (prescaler) allows reduction of the timer input clock frequency. All internal timer interrupt sources are merged in one module interrupt line and one wake-up line. Each internal interrupt sources can be independently enabled/disabled with a dedicated bit of TIER register for the interrupt features and a dedicated bit of TWER for the wake-up.

This module is controllable through the OCP peripheral bus.

As 2 clocks domains are managed inside this module, resynchronization is done by special logic between OCP clock domain and Timer clock domain. At reset, synchronization logic allows utilization of all ratios between OCP clock and Timer clock. Drawback of this mode is that full-resynchronization path is used with access latency performance impact in terms of OCP clock cycles. In order to improve module access latency, and under restricted conditions on clocks ratios (cf. 7.1 Write posted), write-posted mode can be used by setting POSTED bit of System Control register (TSICR). Under this mode, write posted mode is enabled, meaning that OCP write command is granted before write process complete in the timer clock domain. This mode allows software (SW) to do concurrent writes on Dual Mode timer registers and to observe write process completion (synchronization) at SW level by reading independent write posted status bits in the Write Posted Status Register (TWPS).

**Figure 20-23. Block Diagram**

## 20.2.2 Integration



**Figure 20-24. DMTimer 1 ms Integration**

### 20.2.2.1 Timer Connectivity Attributes

**Table 20-27. Timer1 Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Wakeup Domain  |
| Clock Domain        | PD_WKUP_L4_WKUP_GCLK (OCP)<br>PD_WKUP_TIMER1_GCLK (Func) |
| Reset Signals       | WKUP_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle / Slave Wakeup                                |
| Interrupt Requests  | 1 to MPU Subsystem (TINT1_1MS) and WakeM3                |
| DMA Requests        | None   |
| Physical Address    | L4 Wakeup slave port                                     |

### 20.2.2.2 Timer Clock and Reset Management

The DMTimer1 1ms timer functional clock can be selected from one of five sources using the CLKSEL\_TIMER1MS\_CLK register in the PRCM:

- The 24 MHz (typ) system clock (CLK\_M\_OSC)
- The PER PLL generated 32.768 KHz clock (CLK\_32KHZ)
- The TCLKIN external timer input clock
- The on-chip ~32.768 KHz oscillator (CLK\_RC32K)
- The external 32.768 KHz oscillator/clock (CLK\_32K\_RTC)

### 20.2.2.3 Timer Clock Signals

**Table 20-28. Timer Clock Signals**

| Clock Signal                      | Max Freq              | Reference / Source  | Comments                          |
|-----------------------------------|-----------------------|---|-----------------------------------|
| <b>Timer1 (1ms) Clock Signals</b> |                       |   |                                   |
| PICLKOPC<br>Interface clock       | 100 MHz               | CORE_CLKOUTM4 / 2   | pd_wkup_l4_wkup_gclk<br>from PRCM |
| PICLKTIMER<br>Functional clock    | 25 MHz <sup>(1)</sup> | CLK_M_OSC<br>CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375)<br>TCLKIN<br>CLK_RC32K<br>CLK_32K_RTC | pd_wkup_timer1_gclk<br>from PRCM  |

<sup>(1)</sup> PICLKTIMER must be less than PICLKOPC/4.

### 20.2.3 Functional Description

The general-purpose timer is an upward counter. It supports 3 functional modes:

- Timer mode
- Capture mode
- Compare mode

By default, after core reset, the capture and compare modes are disabled.

#### 20.2.3.1 Timer Mode Functionality

The timer is an upward counter that can be started and stopped at any time through the Timer Control Register (TCLR ST bit). The Timer Counter Register (TCRR) can be loaded when stopped or on the fly (while counting). TCRR can be loaded directly by a TCRR Write access with the new timer value. TCRR can also be loaded with the value held in the Timer Load Register TLDR by a trigger register (TTGR) Write access. The TCRR loading is done regardless the TTGR written value. The timer counter register TCRR value can be read when stopped or captured on the fly by a TCRR Read access. The timer is stopped and the counter value set to "0" when the module's reset is asserted. The timer is maintained in stop after reset is released. When the timer is stopped TCRR is frozen and it can be restarted from the frozen value unless TCRR has been reloaded with a new value.

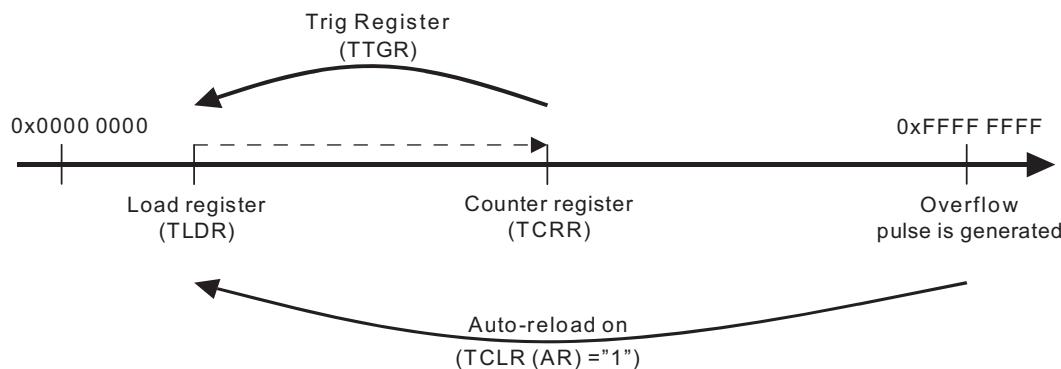
In the one shot mode (TCLR AR bit = "0"), the counter is stopped after counting overflow (counter value remains at zero).

When the auto-reload mode is enabled (TCLR AR bit = "1"), the TCRR is reloaded with the Timer Load Register (TLDR) value after a counting overflow.

It is not recommended to put the overflow value (0xFFFFFFFF) in TLDR because it can lead to undesired results.

An interrupt can be issued on overflow if the overflow interrupt enable bit is set in the timer Interrupt Enable Register (TIER OVF\_IT\_ENA bit = "1"). A dedicated output pin (PORTIMERPWM) is programmed through TCLR (TRG and PT bits) to generate one positive pulse (prescaler duration) or to invert the current value (toggle mode) when an overflow occurs.

**Figure 20-25. TCRR Timing Value**

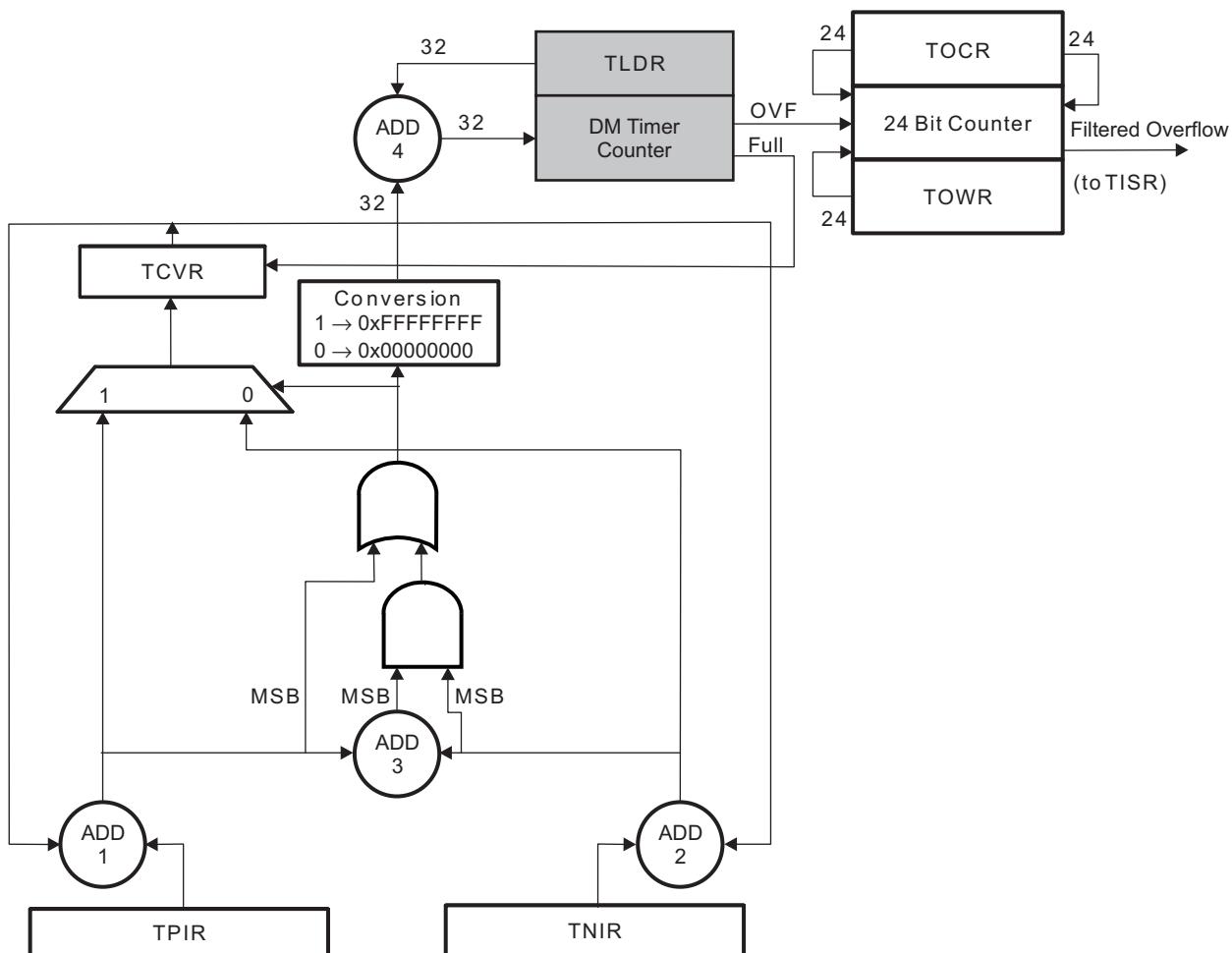


##### 20.2.3.1.1 1 ms Tick Generation

To minimize the error between a true 1ms tick and the tick generated by the 32768 Hz timer the sequencing of the sub-1ms periods and the over-1ms periods must be shuffled.

An additional block (1ms block) is used to correct this error.

In this implementation the increment sequencing is automatically managed by the timer to minimize the error. The value of the Timer Positive Increment register (TPIR) and Timer Negative Increment register (TNIR) only need to be defined by the user. Auto adaptation mechanism is used to simplify the programming model.

**Figure 20-26. 1ms Module Block Diagram**

The TPIR, TNIR, TCVR and adders Add1..3 are used to define whether next value loaded in the TCRR will be value of the TLDR (sub-period value) or the value of TLDR – 1 (over-period value).

The following table shows the value loaded in TCRR according to the sign of the result of Add1, Add2 and Add3. MSB = '0' means a positive value, MSB = '1' means a negative value.

**Table 20-29. Value Loaded in TCRR to Generate 1ms Tick**

| Add1 MSB | Add2 MSB | Add3 MSB | TCRR     |
|----------|----------|----------|----------|
| 0        | 0        | 0        | TLDR     |
| 0        | 0        | 1        | TLDR     |
| 0        | 1        | 0        | TLDR     |
| 0        | 1        | 1        | TLDR - 1 |
| 1        | 0        | 0        | N.A.     |
| 1        | 0        | 1        | N.A.     |
| 1        | 1        | 0        | TLDR - 1 |
| 1        | 1        | 1        | TLDR - 1 |

The values of TPIR and TNIR registers are calculated with formula:

$$\text{Positive Increment Value} = ((\text{INTEGER}[ \text{Fclk} * \text{Ttick} ] + 1) * 1e6) - (\text{Fclk} * \text{Ttick} * 1e6)$$

$$\text{Negative Increment Value} = (\text{INTEGER}[ \text{Fclk} * \text{Ttick} ] * 1e6) - (\text{Fclk} * \text{Ttick} * 1e6)$$

where:

Fclk – clock frequency (KHz)

Ttick – tick period (ms)

The Timer Overflow Counter Register (TOCR) and the Timer Overflow Wrapping Register (TOWR) are used for interrupt filtering. When timer overflows it increments the 24 bit TOCR register. When 24 bit TOCR register values matches the value in the 24 bit TOWR register and timer overflow is asserted, the TOCR is reset and an interrupt is generated to TISR.

With the Conversion block, in reset state (Positive Increment register, Negative Increment register and Counter Value register are all zeroed) the programming model and the behavior of the DMtimer\_dmc1ms remain unchanged.

For 1 ms tick with a 32768-Hz clock:

TPIR = 232000

TNIR = -768000

TLDR = 0xFFFFFE0

**NOTE:** Any value of the tick period can be generated with appropriate value of the TPIR, TNIR and TLDR registers.

By default the TPIR, TNIR, TCVR, TOCR, TOWR registers and the associated logic are in reset mode (all 0s) and have no action on the programming model of the DMtimer\_dmc1ms.

### 20.2.3.2 Capture Mode Functionality

The timer value in TCRR can be captured and saved in TCAR1 or TCAR2 function of the mode selected in TCLR through the field CAPT\_MODE when a transition is detected on the module input pin (PIEVENTCAPT). The edge detection circuitry monitors transitions on the input pin (PIEVENTCAPT).

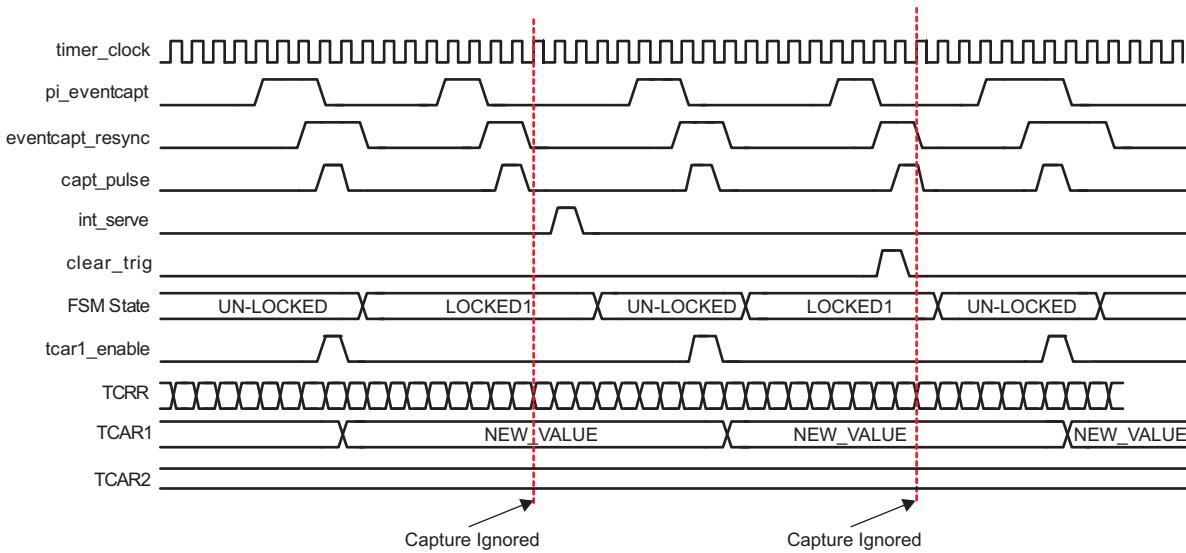
Rising transition, falling transition or both can be selected in TCLR (TCM bit) to trig the timer counter capture. The module sets the TISR (TCAR\_IT\_FLAG bit) when an active transition is detected and at the same time the counter value TCRR is stored in one of the timer capture registers TCAR1 or TCAR2 as follows:

- If TCLR's CAPT\_MODE field is "0" then, on the first enabled capture event, the value of the counter register is saved in TCAR1 register and all the next events are ignored (no update on TCAR1 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a "1" in it..
- If TCLR's CAPT\_MODE field is "1" then, on the first enabled captured event, the counter value is saved in TCAR1 register and, on the second enabled capture event, the value of the counter register is saved in TCAR2 register. If capture interrupt is enabled, the interrupt will be asserted on the second event capture. All the other events are ignored (no update on TCAR1/2 and no interrupt triggering) until the detection logic is reset or the interrupt status register is cleared on TCAR's position writing a "1" in it. This mechanism is useful for period calculation of a clock if that clock is connected to the PIEVENTCAPT input pin.

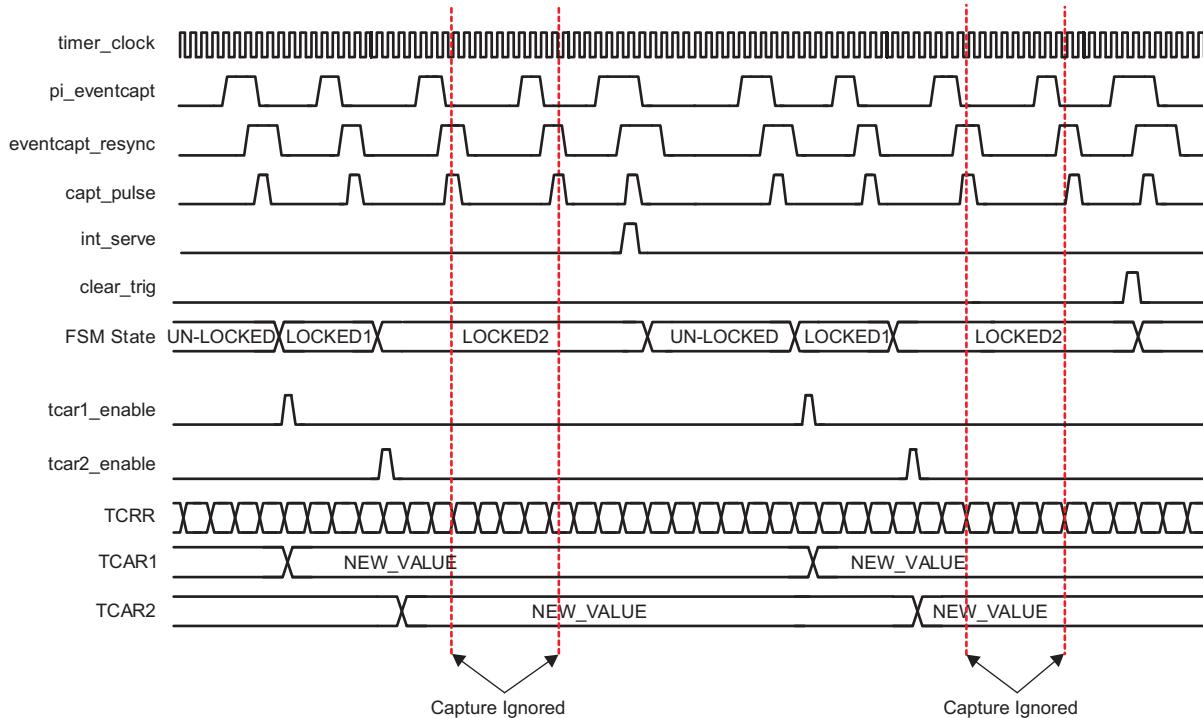
The edge detection logic is reset (a new capture is enabled) when the active capture interrupt is served - TCAR\_IT\_FLAG bit of TISR (previously '1') is cleared through a "1" written in it or when the edge detection mode bits TCLR (TCM bit) passed from the No Capture Mode detection to any other modes. The timer functional clock (input to prescaler) is used to sample the input pin (PIEVENTCAPT). Input negative or positive pulse can be detected when pulse time is above functional clock period. An interrupt can be issued on transition detection if the capture interrupt enable bit is set in the Timer Interrupt Enable Register TIER (TCAR\_IT\_ENA bit).

See the following examples:

In the next wave, the TCM value is "01" and CAPT\_MODE is "0"- only rising edge of the PIEVENTCAPT will trigger a capture in TCAR and only TCAR1 will update.

**Figure 20-27. Capture Wave Example for CAPT\_MODE 0**


In the following example, the TCM value is “01” and CAPT\_MODE is “1”- only rising edge of the PIEVENTCAPT will trigger a capture in TCAR1 on first enabled event and TCAR2 will update on the second enabled event.

**Figure 20-28. Capture Wave Example for CAPT\_MODE 1**


### 20.2.3.3 Compare Mode Functionality

When Compare Enable TCLR (CE bit) is set to “1”, the timer value (TCRR) is permanently compared to the value held in timer match register (TMAR). TMAR value can be loaded at any time (timer counting or stop). When the TCRR and the TMAR values match, an interrupt can be issued if the TIER (MAT\_IT\_ENA bit) is set. The right programming way is to write a compare value in TMAR register before setting TCLR (CE bit) to avoid any unwanted interrupt due to a reset value matching effect.

The dedicated output pin (PORTIMERPWM) can be programmed through TCLR (TRG and PT bits) to generate one positive pulse (TIMER clock duration) or to invert the current value (toggle mode) when an overflow and a match occur.

#### 20.2.3.4 Prescaler Functionality

A prescaler counter can be used to divide the timer counter input clock frequency. The prescaler is enabled when TCLR bit 5 is set (PRE). The  $2^n$  division ratio value (PTV) can be configured in the TCLR register.

The prescaler counter is reset when the timer counter is stopped or reloaded on the fly.

**Table 20-30. Prescaler/Timer Reload Values Versus Contexts**

| Contexts                       | Prescaler Counter | Timer Counter |
|--------------------------------|-------------------|---------------|
| Overflow (when Auto-reload on) | reset             | TLDR          |
| TCRR Write                     | reset             | TCRR          |
| TTGR Write                     | reset             | TLDR          |
| Stop                           | reset             | Frozen        |

#### 20.2.3.5 Pulse-Width Modulation

The timer can be configured to provide a programmable pulse-width modulation (PORTIMERPWM) output. The PORTIMERPWM output pin can be configured to toggle on specified event. TCLR (TRG bits) determines on which register value the PORTIMERPWM pin toggles. Either overflow or match can be used to toggle the PORTIMERPWM pin, when a compare condition occurs.

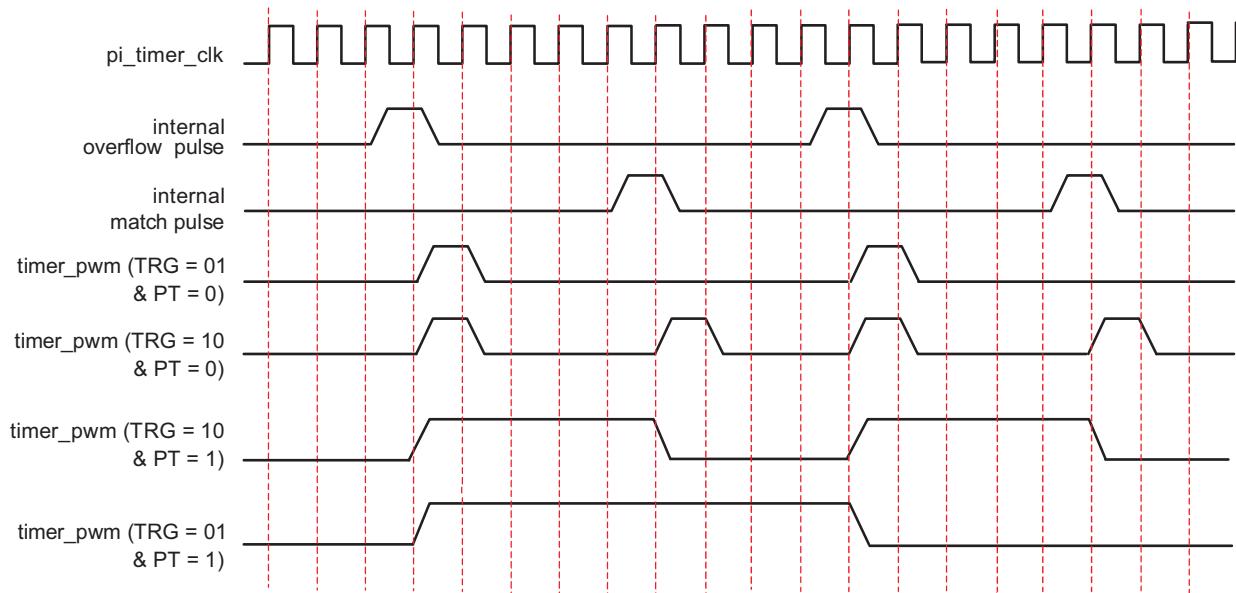
In case of overflow and match mode, the match event will be ignored from the moment the mode was set up until the first overflow event occurs

The TCLR (SCPWM bit) can be programmed to set or clear the PORTIMERPWM output signal while the counter is stopped or the triggering is off only. This allows fixing a deterministic state of the output pin when modulation is stopped. The modulation is synchronously stopped when TRG bit is cleared and overflow occurred.

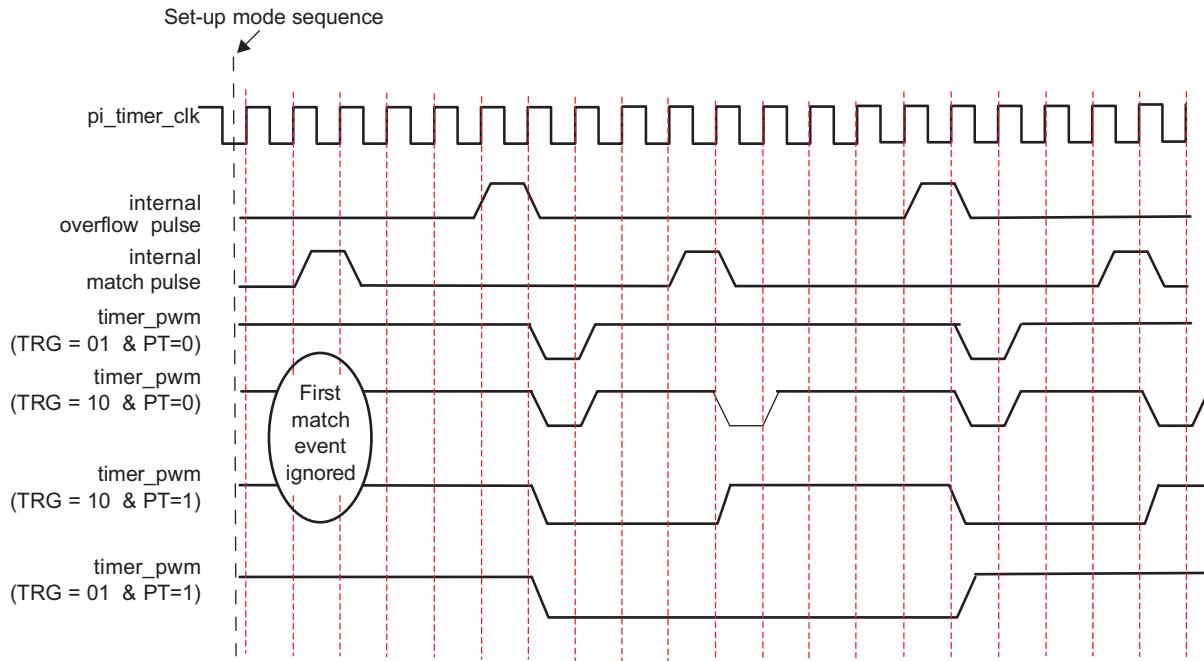
In the following timing diagram, the internal overflow pulse is set each time (0xFFFF FFFF – TLDR +1) value is reached, and the internal match pulse is set when the counter reaches TMAR register value. According to TCLR (TRG and PT bits) programming value, the timer provides pulse or PWM on the output pin (PORTIMERPWM).

The TLDR and TMAR registers must keep values smaller than the overflow value (0xFFFFFFFF) with at least 2 units. In case the PWM trigger events are both overflow and match, the difference between the values kept in TMAR register and the value in TLDR must be at least 2 units. When match event is used the compare mode TCLR (CE) must be set.

On the following wave TCLR (SCPWM bit) is set to '0'.

**Figure 20-29. Timing Diagram of Pulse-Width Modulation, SCPWM Bit = 0**


On the next wave **TCLR** (SCPWM bit) is set to '1'.

**Figure 20-30. Timing Diagram of Pulse-Width Modulation, SCPWM Bit = 1**


### 20.2.3.6 Timer Interrupt Control

The timer can issue an overflow interrupt, a timer match interrupt and a timer capture interrupt. Each internal interrupt sources can be independently enabled/disabled in the Interrupt Enable Register **TIER**. When the interrupt event has been issued the associated interrupt status bit is set in the Timer Status Register (**TISR**). The pending interrupt event is reset when the set status bit is overwritten by a "1" value. Reading the Interrupt Status Register and writing the value back allows fast acknowledge interrupt process.

### 20.2.3.7 Sleep Mode Request and Acknowledge

Upon a Sleep mode request issued by the host processor (the Idle Request PIOCPMIDLREQ signal is active), the timer module will go in Sleep mode according to the IdleMode field of the System configuration register (see TIOCP\_CFG and Ref. [4]).

If the IdleMode field sets No-Idle mode, the Timer does not go in Sleep mode and the Idle acknowledge signal (POROCPSIDLEACK) is never asserted.

If the IdleMode field sets Force-Idle mode, the timer goes in Sleep mode independently of the internal module state and the Idle acknowledge signal (POROCPSIDLEACK) is unconditionally asserted.

If the IdleMode field sets Smart-Idle mode, the timer module evaluates its internal capability to have the interface/functional clock switched off. Depending on the ClockActivity field setting the timer module evaluates the internal activity and asserts the Idle acknowledge signal (POROCPSIDLEACK), entering in Sleep mode, ready to issue a wake-up request.

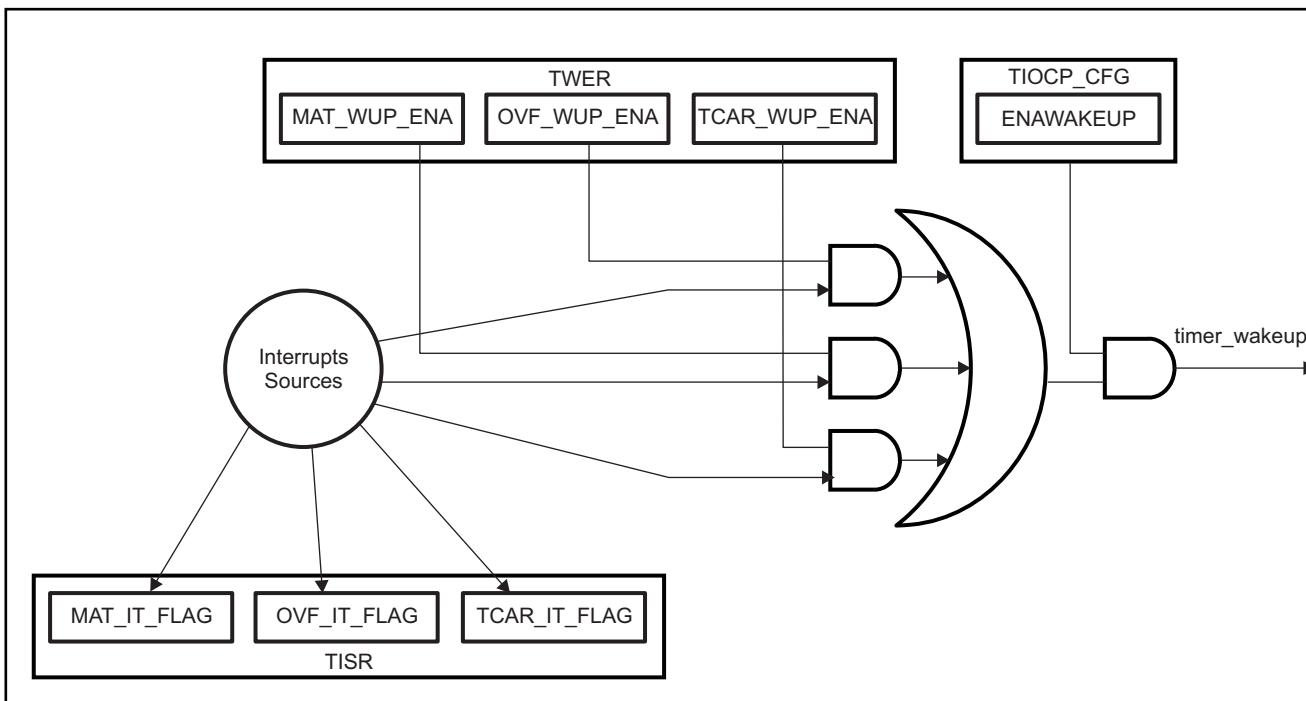
The following table describes the Smart Idle behavior according to the clock activity setting:

**Table 20-31. SmartIdle - Clock Activity Field Configuration**

| Clock Activity | Functional Clock | OCP Clock | Module Behavior  |
|----------------|------------------|-----------|--|
| 11             | ON               | ON        |  |
| 10             | ON               | OFF       | The Idle acknowledge signal is asserted when there are no pending activities on the OCP clock domain, without evaluating the pending activities on the functional clock domain. (The module will enter in Sleep mode and if a pending interrupt event is finished during Idle mode the wake-up signal will be asserted). |
| 01             | OFF              | ON        |  |
| 00             | OFF              | OFF       | The Idle acknowledge signal is asserted when there are no pending activities on the functional and OCP clock domains (Improved latency in assertion of Idle acknowledge). The Wake-up capability of the module is disabled.  |

This wake-up request is effectively sent only if the field ENAWAKEUP of TIOCP\_CFG enables the timer wake-up capability. When the system is awaken, the Idle Request signal goes inactive and the wake-up request signal is also de-asserted.

**Figure 20-31. Wake-up Request Generation**



#### 20.2.3.7.1 Wake-up Line Release

When the host processor receives wake-up request issued by the timer peripheral, the interface clock is re-activated: the host processor deactivates the PIOCPMIDLREQ, the timer deactivates the POROCPSIDLEACK signal and then the host can read the corresponding bit in TISR to find out which interrupt source has triggered wake-up request. After acknowledging the wake-up request, the processor resets the status bit and releases the interrupt line by writing a '1' in the corresponding bit of the TISR register.

#### 20.2.3.8 Timer Counting Rate

The dmtimer's counter is composed of a prescaler stage and a timer counter.

Ratio can be managed by accessing the ratio definition field of the control register (PTV and PRE of TCLR).

The timer rate is defined by:

- The value of the prescaler fields (PRE and PTV of TCLR register)
- The value loaded into the Timer Load Register (TLDR).

**Table 20-32. Prescaler Clock Ratios Value**

| PRE | PTV | Divisor (PS) |
|-----|-----|--------------|
| 0   | X   | 1            |
| 1   | 0   | 2            |
| 1   | 1   | 4            |
| 1   | 2   | 8            |
| 1   | 3   | 16           |
| 1   | 4   | 32           |

**Table 20-32. Prescaler Clock Ratios Value (continued)**

| PRE | PTV | Divisor (PS) |
|-----|-----|--------------|
| 1   | 5   | 64           |
| 1   | 6   | 128          |
| 1   | 7   | 256          |

The timer rate equation is as follows:

$$(0xFFFF FFFF - TLDR + 1) \times \text{timer Clock period} \times \text{Clock Divider (PS)}$$

With timer Clock period = 1/ timer Clock frequency and PS = 2(PTV + 1).

As example, if we consider a timer clock input of 32 KHz, with a PRE field equals to "0", the timer output period is:

**Table 20-33. Value and Corresponding Interrupt Period**

| TLDR        | Interrupt Period |
|-------------|------------------|
| 0x0000 0000 | 37 h             |
| 0xFFFF 0000 | 2 s              |
| 0xFFFF FFF0 | 500 us           |
| 0xFFFF FFFE | 62.5 us          |

### 20.2.3.9 Dual Mode Timer Under Emulation

During emulation mode (when PINSUSPENDN signal is active), the timer can/cannot continue running according to the value of the EmuFree bit of the timer OCP configuration register (TIOCP\_CFG).

If EmuFree is 1, timer execution is not stopped and, regardless of the value of PINSUSPENDN signal, and the interrupt assertion is still generated when overflow is reached.

If EmuFree is 0, counters (prescaler/timer) are frozen and an increment start occurs again as soon as PINSUSPENDN becomes inactive. The asynchronous input pin is internally synchronized on 2 TIMER clock rising edges.

### 20.2.4 Use Cases

### 20.2.5 DMTIMER\_1MS Registers

Table 20-34 lists the memory-mapped registers for the DMTIMER\_1MS. All register offset addresses not listed in Table 20-34 should be considered as reserved locations and the register contents should not be modified.

**Table 20-34. DMTIMER\_1MS REGISTERS**

| Offset | Acronym   | Register Name  | Section                          |
|--------|-----------|--|----------------------------------|
| 0h     | TIDR      | This register contains the IP revision code  | <a href="#">Section 20.2.5.1</a> |
| 10h    | TIOCP_CFG | This register controls the various parameters of the OCP interface                                     | <a href="#">Section 20.2.5.2</a> |
| 14h    | TISTAT    | This register provides status information about the module, excluding the interrupt status information | <a href="#">Section 20.2.5.3</a> |
| 18h    | TISR      | The Timer Status Register is used to determine which of the timer events requested an interrupt.       | <a href="#">Section 20.2.5.4</a> |
| 1Ch    | TIER      | This register controls (enable/disable) the interrupt events   | <a href="#">Section 20.2.5.5</a> |
| 20h    | TWER      | This register controls (enable/disable) the wakeup feature on specific interrupt events                | <a href="#">Section 20.2.5.6</a> |
| 24h    | TCLR      | This register controls optional features specific to the timer functionality                           | <a href="#">Section 20.2.5.7</a> |

**Table 20-34. DMIMER\_1MS REGISTERS (continued)**

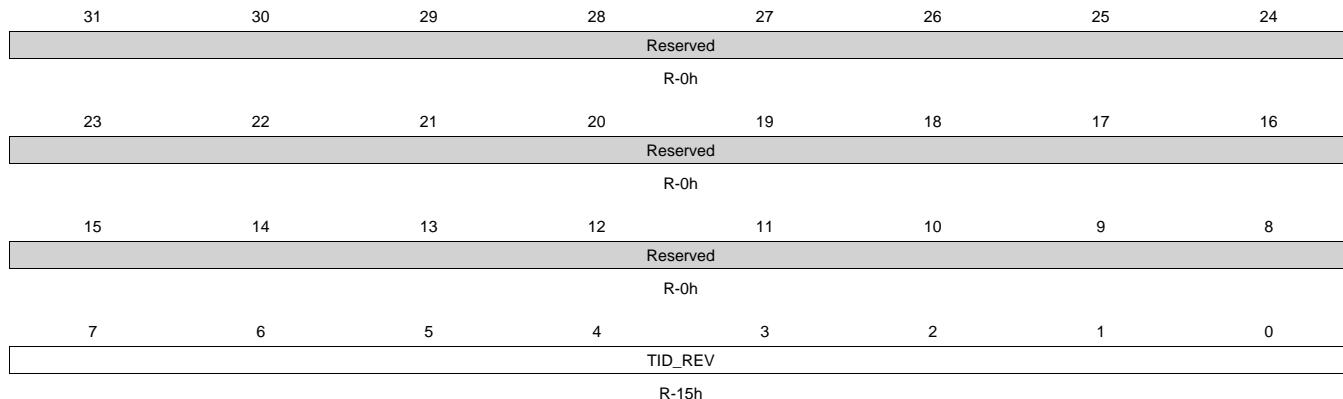
| Offset | Acronym | Register Name   | Section                           |
|--------|---------|---|-----------------------------------|
| 28h    | TCRR    | This register holds the value of the internal counter   | <a href="#">Section 20.2.5.8</a>  |
| 2Ch    | TLDR    | This register holds the timer's load value  | <a href="#">Section 20.2.5.9</a>  |
| 30h    | TTGR    | This register triggers a counter reload of timer by writing any value in it.  | <a href="#">Section 20.2.5.10</a> |
| 34h    | TWPS    | This register contains the write posting bits for all writable functional registers   | <a href="#">Section 20.2.5.11</a> |
| 38h    | TMAR    | This register holds the match value to be compared with the counter's value   | <a href="#">Section 20.2.5.12</a> |
| 3Ch    | TCAR1   | This register holds the value of the first counter register capture   | <a href="#">Section 20.2.5.13</a> |
| 40h    | TSICR   | Timer Synchronous Interface Control Register  | <a href="#">Section 20.2.5.14</a> |
| 44h    | TCAR2   | This register holds the value of the second counter register capture  | <a href="#">Section 20.2.5.15</a> |
| 48h    | TPIR    | This register is used for 1ms tick generation.<br>The TPIR register holds the value of the positive increment.<br><br>The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value. | <a href="#">Section 20.2.5.16</a> |
| 4Ch    | TNIR    | This register is used for 1ms tick generation.<br>The TNIR register holds the value of the negative increment.<br><br>The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value. | <a href="#">Section 20.2.5.17</a> |
| 50h    | TCVR    | This register is used for 1ms tick generation.<br>The TCVR register defines whether next value loaded in TCRR will be the sub-period value or the over-period value.  | <a href="#">Section 20.2.5.18</a> |
| 54h    | TOCR    | This register is used to mask the tick interrupt for a selected number of ticks.  | <a href="#">Section 20.2.5.19</a> |
| 58h    | TOWR    | This register holds the number of masked overflow interrupts.   | <a href="#">Section 20.2.5.20</a> |

### 20.2.5.1 TIDR Register (offset = 0h) [reset = 15h]

TIDR is shown in [Figure 20-32](#) and described in [Table 20-35](#).

This register contains the IP revision code

**Figure 20-32. TIDR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-35. TIDR Register Field Descriptions**

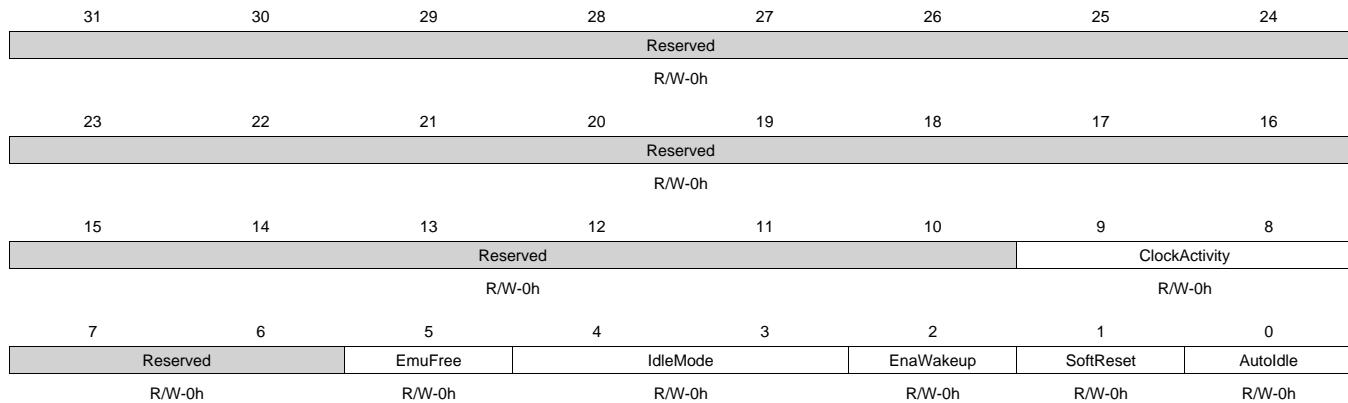
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    | Reads return 0   |
| 7-0  | TID_REV  | R    | 15h   | IP revision [<br>7:4] Major revision [<br>3:0] Minor revision Examples: 0x10 for 1.0, 0x21 for 2.1 |

### 20.2.5.2 TIOCP\_CFG Register (offset = 10h) [reset = 0h]

TIOCP\_CFG is shown in [Figure 20-33](#) and described in [Table 20-36](#).

This register controls the various parameters of the OCP interface

**Figure 20-33. TIOCP\_CFG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-36. TIOCP\_CFG Register Field Descriptions**

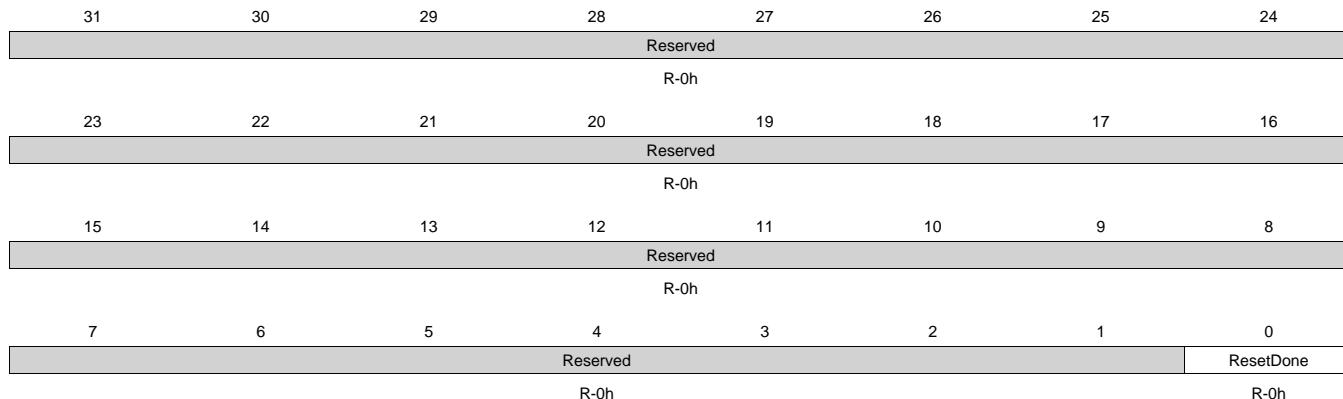
| Bit   | Field         | Type | Reset | Description   |
|-------|---------------|------|-------|---|
| 31-10 | Reserved      | R/W  | 0h    |   |
| 9-8   | ClockActivity | R/W  | 0h    |   |
| 7-6   | Reserved      | R/W  | 0h    | Write 0's for future compatibility Reads return 0   |
| 5     | EmuFree       | R/W  | 0h    | Emulation mode<br>0 = timer_frozen : Timer counter frozen in emulation<br>1 = timer_free : Timer counter free-running in emulation  |
| 4-3   | IdleMode      | R/W  | 0h    | Power Management, req/ack control<br>0 = fidle : Force-idle. An idle request is acknowledged unconditionally<br>1 = nidle : No-idle. An idle request is never acknowledged<br>2 = sidle : Smart-idle. Acknowledgement to an idle request is given based on the internal activity of the module<br>3 = res : reserved do not use |
| 2     | EnaWakeup     | R/W  | 0h    | Wake-up feature global control<br>0 = nowake : No wakeup line assertion in idle mode<br>1 = enwake : Wakeup line assertion enabled in smart-idle mode   |
| 1     | SoftReset     | R/W  | 0h    | Software reset.<br>This bit is automatically reset by the hardware.<br>During reads, it always return 0<br>0 = nmode : Normal mode<br>1 = rstmode : The module is reset   |
| 0     | Autoldle      | R/W  | 0h    | Internal OCP clock gating strategy<br>0 = clkfree : OCP clock is free-running<br>1 = clkgate : Automatic OCP clock gating strategy is applied, based on the OCP interface activity  |

### 20.2.5.3 TISTAT Register (offset = 14h) [reset = 0h]

TISTAT is shown in [Figure 20-34](#) and described in [Table 20-37](#).

This register provides status information about the module, excluding the interrupt status information

**Figure 20-34. TISTAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-37. TISTAT Register Field Descriptions**

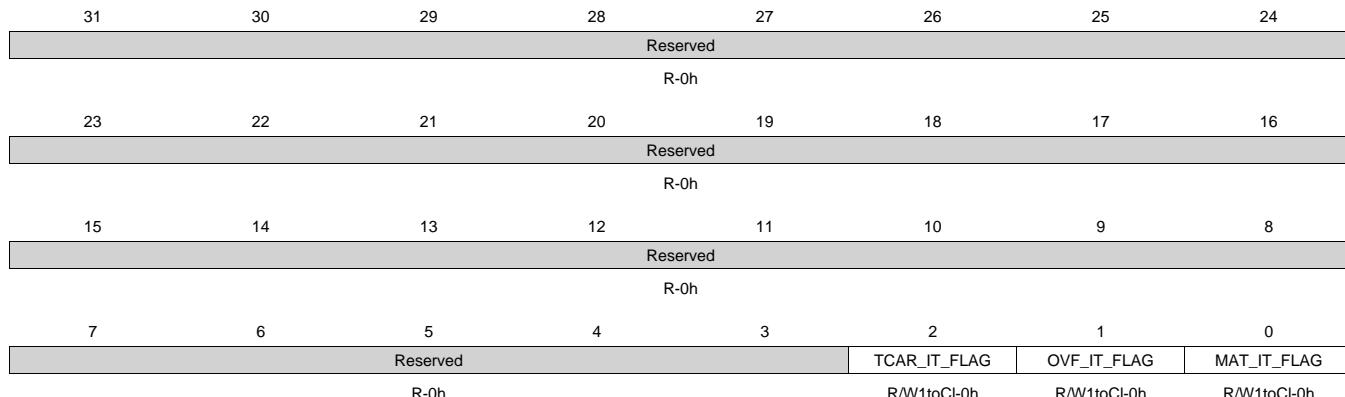
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-1 | Reserved  | R    | 0h    | Reads return 0 Reserved for OCP-socket status information  |
| 0    | ResetDone | R    | 0h    | Internal reset monitoring<br>0 = rstongoing : Internal module reset in on-going<br>1 = rstcomp : Reset completed |

#### 20.2.5.4 TISR Register (offset = 18h) [reset = 0h]

TISR is shown in [Figure 20-35](#) and described in [Table 20-38](#).

The Timer Status Register is used to determine which of the timer events requested an interrupt.

**Figure 20-35. TISR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-38. TISR Register Field Descriptions**

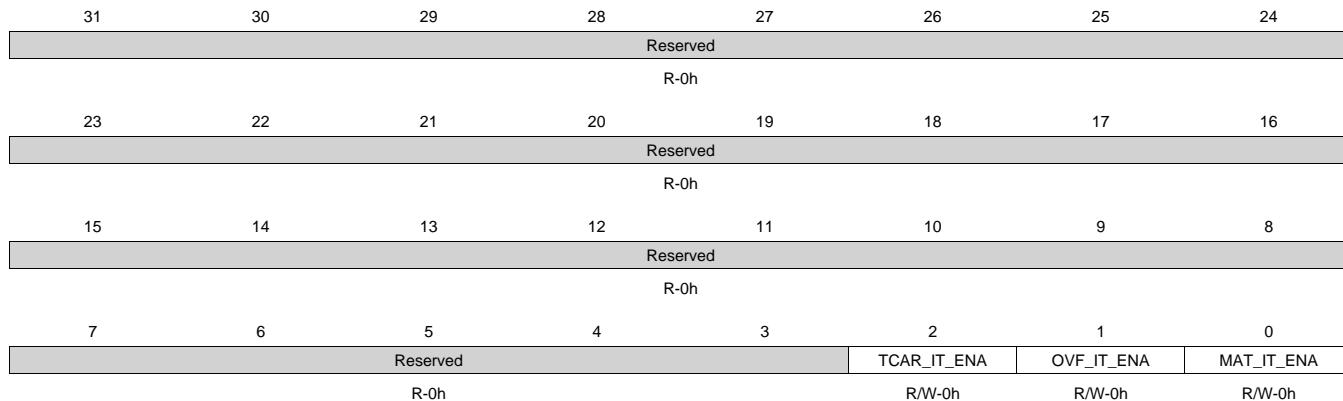
| Bit  | Field        | Type     | Reset | Description  |
|------|--------------|----------|-------|--|
| 31-3 | Reserved     | R        | 0h    | Reads return 0   |
| 2    | TCAR_IT_FLAG | R/W1toCl | 0h    | indicates when an external pulse transition of the correct polarity is detected on the external pin PIEVENTCAPT<br>0 = TCAR_IT_FLAG_0 : no capture interrupt request<br>1 = TACR_IT_FLAG_1 : capture interrupt request |
| 1    | OVF_IT_FLAG  | R/W1toCl | 0h    | TCRR overflow<br>0 = OVF_IT_FLAG_0 : no overflow interrupt request<br>1 = OVF_IT_FLAG_1 : overflow interrupt pending   |
| 0    | MAT_IT_FLAG  | R/W1toCl | 0h    | the compare result of TCRR and TMAR<br>0 = MAT_IT_FLAG_0 : no compare interrupt request<br>1 = MAT_IT_FLAG_1 : compare interrupt pending   |

### 20.2.5.5 TIER Register (offset = 1Ch) [reset = 0h]

TIER is shown in [Figure 20-36](#) and described in [Table 20-39](#).

This register controls (enable/disable) the interrupt events

**Figure 20-36. TIER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-39. TIER Register Field Descriptions**

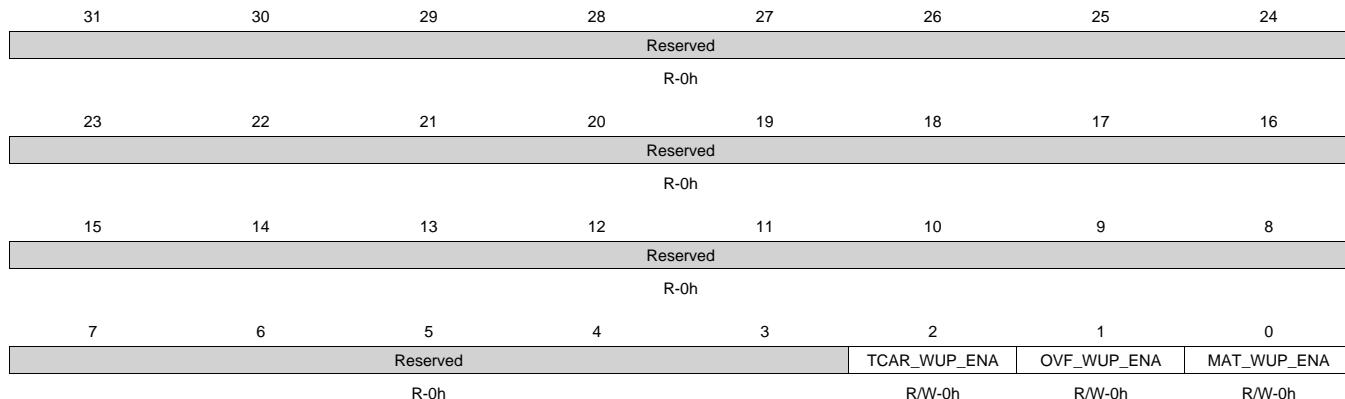
| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-3 | Reserved    | R    | 0h    | Reads return 0   |
| 2    | TCAR_IT_ENA | R/W  | 0h    | Enable capture interrupt<br>0 = Dsb_capt : Disable capture interrupt<br>1 = Enb_capt : Enable capture interrupt  |
| 1    | OVF_IT_ENA  | R/W  | 0h    | Enable overflow interrupt<br>0 = Dsb_ovf : Disable overflow interrupt<br>1 = Enb_ovf : Enable overflow interrupt |
| 0    | MAT_IT_ENA  | R/W  | 0h    | Enable match interrupt<br>0 = Dsb_match : Disable match interrupt<br>1 = Enb_match : Enable match interrupt      |

### 20.2.5.6 TWER Register (offset = 20h) [reset = 0h]

TWER is shown in Figure 20-37 and described in Table 20-40.

This register controls (enable/disable) the wakeup feature on specific interrupt events

**Figure 20-37. TWER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-40. TWER Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description  |
|------|--------------|------|-------|--|
| 31-3 | Reserved     | R    | 0h    | Reads return 0   |
| 2    | TCAR_WUP_ENA | R/W  | 0h    | Enable capture wake-up<br>0 = DsbWupCap : Disable capture wake-up<br>1 = EnbWupCapt : Enable capture wake-up   |
| 1    | OVF_WUP_ENA  | R/W  | 0h    | Enable overflow wake-up<br>0 = DsbWupOvf : Disable overflow wake-up<br>1 = EnbWupOvf : Enable overflow wake-up |
| 0    | MAT_WUP_ENA  | R/W  | 0h    | Enable match wake-up<br>0 = DsbWupMat : Disable match wake-up<br>1 = EnbWupMat : Enable match wake-up          |

### 20.2.5.7 TCLR Register (offset = 24h) [reset = 0h]

TCLR is shown in [Figure 20-38](#) and described in [Table 20-41](#).

This register controls optional features specific to the timer functionality

**Figure 20-38. TCLR Register**

|          |         |           |        |        |        |        |        |
|----------|---------|-----------|--------|--------|--------|--------|--------|
| 31       | 30      | 29        | 28     | 27     | 26     | 25     | 24     |
| Reserved |         |           |        |        |        |        |        |
| R-0h     |         |           |        |        |        |        |        |
| 23       | 22      | 21        | 20     | 19     | 18     | 17     | 16     |
| Reserved |         |           |        |        |        |        |        |
| R-0h     |         |           |        |        |        |        |        |
| 15       | 14      | 13        | 12     | 11     | 10     | 9      | 8      |
| Reserved | GPO_CFG | CAPT_MODE | PT     | TRG    | TCM    |        |        |
| R-0h     | R/W-0h  | R/W-0h    | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6       | 5         | 4      | 3      | 2      | 1      | 0      |
| SCPWM    | CE      | PRE       |        | PTV    | AR     | ST     |        |
| R/W-0h   | R/W-0h  | R/W-0h    |        | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-41. TCLR Register Field Descriptions**

| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-15 | Reserved  | R    | 0h    | Reads return 0   |
| 14    | GPO_CFG   | R/W  | 0h    |  |
| 13    | CAPT_MODE | R/W  | 0h    | Capture mode select bit (first/second)<br>0 = First_capt : Capture the first enabled capture event in TCAR1<br>1 = Sec_capt : Capture the second enabled capture event in TCAR2  |
| 12    | PT        | R/W  | 0h    | Pulse or Toggle select bit<br>0 = pulse : pulse modulation<br>1 = toggle : toggle modulation   |
| 11-10 | TRG       | R/W  | 0h    | Trigger Output Mode<br>0 = no_trg : No trigger<br>1 = ovf_trg : Overflow trigger<br>2 = ovf_mat_trg : Overflow and match trigger<br>3 = reserved : Reserved  |
| 9-8   | TCM       | R/W  | 0h    | Transition Capture Mode<br>0 = no_edge : No capture<br>1 = rise_edge : Capture on rising edges of PIEVETCAPT<br>2 = fall_edge : Capture on falling edges of PIEVETCAPT<br>3 = booth_edges : Capture on booth edges of PIEVETCAPT |
| 7     | SCPWM     | R/W  | 0h    | Pulse Width Modulation output pin default value<br>0 = def_low : default value of PORPWM: 0<br>1 = def_high : default value of PORPWM: 1   |
| 6     | CE        | R/W  | 0h    | Compare enable<br>0 = dsb_cmp : Compare disabled<br>1 = enb_cmp : Compare enabled  |
| 5     | PRE       | R/W  | 0h    | Prescaler enable<br>0 = no_prescal : Prescaler disabled<br>1 = prescal_on : Prescaler enabled  |
| 4-2   | PTV       | R/W  | 0h    | Trigger Output Mode  |

**Table 20-41. TCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 1   | AR    | R/W  | 0h    | Auto-reload mode<br>0 = one_shot : One shot mode overflow<br>1 = auto_rel : Auto-reload mode overflow |
| 0   | ST    | R/W  | 0h    | Start/Stop timer control<br>0 = cnt_stop : Stop the timer<br>1 = cnt_start : Start the timer          |

### 20.2.5.8 TCRR Register (offset = 28h) [reset = 0h]

TCRR is shown in [Figure 20-39](#) and described in [Table 20-42](#).

This register holds the value of the internal counter

**Figure 20-39. TCRR Register**

| 31            | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TIMER_COUNTER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-42. TCRR Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description                             |
|------|---------------|------|-------|---|
| 31-0 | TIMER_COUNTER | R/W  | 0h    | The value of the timer counter register |

### 20.2.5.9 TLDR Register (offset = 2Ch) [reset = 0h]

TLDR is shown in [Figure 20-40](#) and described in [Table 20-43](#).

This register holds the timer's load value

**Figure 20-40. TLDR Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOAD_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-43. TLDR Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                          |
|------|------------|------|-------|--------------------------------------|
| 31-0 | LOAD_VALUE | R/W  | 0h    | The value of the timer load register |

### 20.2.5.10 TTGR Register (offset = 30h) [reset = FFFFFFFFh]

TTGR is shown in [Figure 20-41](#) and described in [Table 20-44](#).

This register triggers a counter reload of timer by writing any value in it.

**Figure 20-41. TTGR Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| TTGR_VALUE      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-FFFFFFFFFFh |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-44. TTGR Register Field Descriptions**

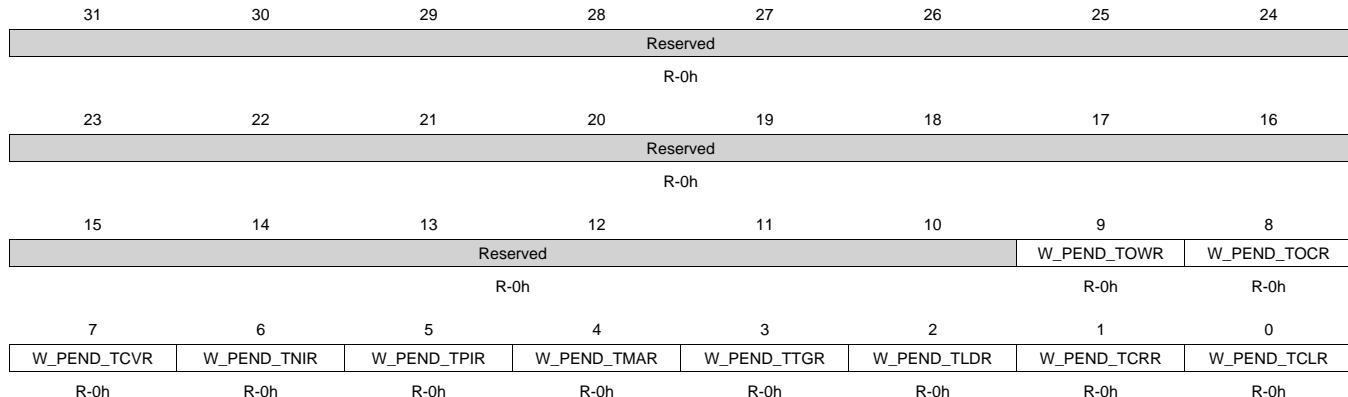
| Bit  | Field      | Type | Reset     | Description  |
|------|------------|------|-----------|--|
| 31-0 | TTGR_VALUE | R/W  | FFFFFFFFh | The value of the trigger register During reads, it always returns "0xFFFFFFFF" |

### 20.2.5.11 TWPS Register (offset = 34h) [reset = 0h]

TWPS is shown in [Figure 20-42](#) and described in [Table 20-45](#).

This register contains the write posting bits for all writ-able functional registers

**Figure 20-42. TWPS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-45. TWPS Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description   |
|-------|-------------|------|-------|---|
| 31-10 | Reserved    | R    | 0h    | Reads return 0  |
| 9     | W_PEND_TOWR | R    | 0h    | Write pending for register TOWR<br>0 = OWR_nPend : No Overflow Wrapping Register write pending.<br>1 = OWR_Pend : Overflow Wrapping Register write pending.   |
| 8     | W_PEND_TOCR | R    | 0h    | Write pending for register TOCR<br>0 = OCR_nPend : No Overflow Counter Register write pending.<br>1 = OCR_Pend : Overflow Counter Register write pending.     |
| 7     | W_PEND_TCVR | R    | 0h    | Write pending for register TCVR<br>0 = CVR_nPend : No Counter Register write pending.<br>1 = CVR_Pend : Counter Register write pending.                       |
| 6     | W_PEND_TNIR | R    | 0h    | Write pending for register TNIR<br>0 = NIR_nPend : No Negativ Increment Register write pending.<br>1 = NIR_Pend : Negativ Increment Register write pending.   |
| 5     | W_PEND_TPIR | R    | 0h    | Write pending for register TPIR<br>0 = PIR_nPend : No Positive Increment Register write pending.<br>1 = PIR_Pend : Positive Increment Register write pending. |
| 4     | W_PEND_TMAR | R    | 0h    | Write pending for register TMAR<br>0 = MAR_nPend : No Match Register write pending<br>1 = MAR_Pend : Match Register write pending                             |
| 3     | W_PEND_TTGR | R    | 0h    | Write pending for register TTGR<br>0 = TGR_nPend : No Trigger Register write pending<br>1 = TGR_Pend : Trigger Register write pending                         |
| 2     | W_PEND_TLDR | R    | 0h    | Write pending for register TLDR<br>0 = LDR_nPend : No Load Register write pending<br>1 = LDR_Pend : Load Register write pending                               |
| 1     | W_PEND_TCRR | R    | 0h    | Write pending for register TCRR<br>0 = CRR_nPend : No Counter Register write pending<br>1 = CRR_Pend : Counter Register write pending                         |

**Table 20-45. TWPS Register Field Descriptions (continued)**

| Bit | Field       | Type | Reset | Description   |
|-----|-------------|------|-------|---|
| 0   | W_PEND_TCLR | R    | 0h    | Write pending for register TCLR<br>0 = CLR_nPend : No Control Register write pending<br>1 = CLR_Pend : Control Register write pending |

### 20.2.5.12 TMAR Register (offset = 38h) [reset = 0h]

TMAR is shown in [Figure 20-43](#) and described in [Table 20-46](#).

This register holds the match value to be compared with the counter's value

**Figure 20-43. TMAR Register**

|               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31            | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMPARE_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-46. TMAR Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description                     |
|------|---------------|------|-------|---------------------------------|
| 31-0 | COMPARE_VALUE | R/W  | 0h    | The value of the match register |

### 20.2.5.13 TCAR1 Register (offset = 3Ch) [reset = 0h]

TCAR1 is shown in [Figure 20-44](#) and described in [Table 20-47](#).

This register holds the value of the first counter register capture

**Figure 20-44. TCAR1 Register**

| 31             | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| CAPTURE_VALUE1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-47. TCAR1 Register Field Descriptions**

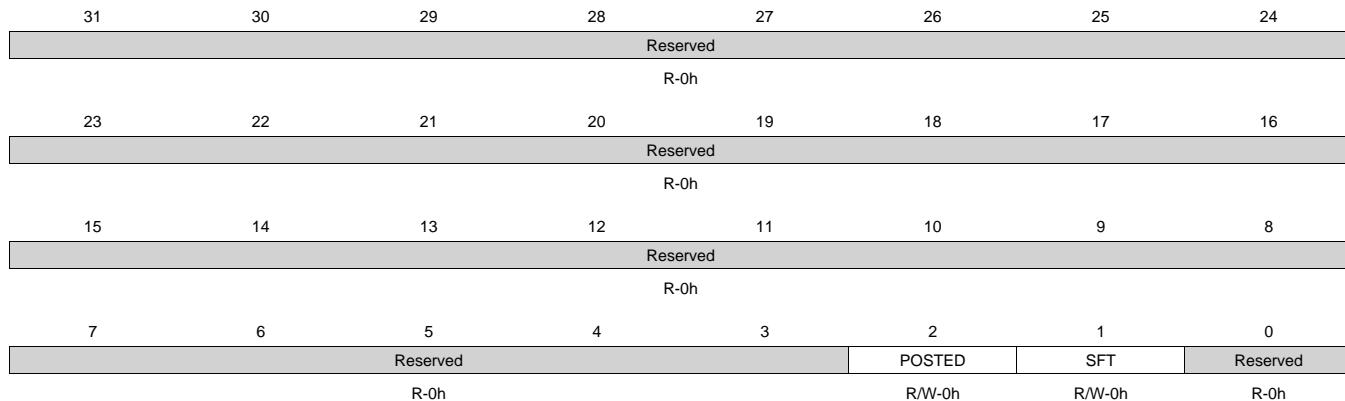
| Bit  | Field          | Type | Reset | Description                                  |
|------|----------------|------|-------|--|
| 31-0 | CAPTURE_VALUE1 | R    | 0h    | The value of first captured counter register |

### 20.2.5.14 TSICR Register (offset = 40h) [reset = 0h]

TSICR is shown in [Figure 20-45](#) and described in [Table 20-48](#).

Timer Synchronous Interface Control Register

**Figure 20-45. TSICR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-48. TSICR Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-3 | Reserved | R    | 0h    | Reads return 0   |
| 2    | POSTED   | R/W  | 0h    | posted mode active/inactive<br>0 = POSTED_0 : posted mode inactive: will delay the command accept output signal<br>1 = POSTED_1 : posted mode active (clocks ratio needs to fit freq (timer) and it; freq (OCP)/4 frequency requirement) |
| 1    | SFT      | R/W  | 0h    | This bit reset all the functional part of the module<br>0 = SFT_0 : software reset is disabled<br>1 = SFT_1 : software reset is enabled  |
| 0    | Reserved | R    | 0h    | Reads return 0   |

### 20.2.5.15 TCAR2 Register (offset = 44h) [reset = 0h]

TCAR2 is shown in [Figure 20-46](#) and described in [Table 20-49](#).

This register holds the value of the second counter register capture

**Figure 20-46. TCAR2 Register**

| 31             | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| CAPTURE_VALUE2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-49. TCAR2 Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description                                   |
|------|----------------|------|-------|---|
| 31-0 | CAPTURE_VALUE2 | R    | 0h    | The value of second captured counter register |

### 20.2.5.16 TPIR Register (offset = 48h) [reset = 0h]

TPIR is shown in [Figure 20-47](#) and described in [Table 20-50](#).

This register is used for 1ms tick generation. The TPIR register holds the value of the positive increment. The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value.

**Figure 20-47. TPIR Register**

|                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| POSITIVE_INC_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-50. TPIR Register Field Descriptions**

| Bit  | Field              | Type | Reset | Description                          |
|------|--------------------|------|-------|--------------------------------------|
| 31-0 | POSITIVE_INC_VALUE | R/W  | 0h    | The value of the positive increment. |

### 20.2.5.17 TNIR Register (offset = 4Ch) [reset = 0h]

TNIR is shown in [Figure 20-48](#) and described in [Table 20-51](#).

This register is used for 1ms tick generation. The TNIR register holds the value of the negative increment. The value of this register is added with the value of the TCVR to define whether next value loaded in TCRR will be the sub-period value or the over-period value.

**Figure 20-48. TNIR Register**

|                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31                 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NEGATIVE_INV_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-51. TNIR Register Field Descriptions**

| Bit  | Field              | Type | Reset | Description                          |
|------|--------------------|------|-------|--------------------------------------|
| 31-0 | NEGATIVE_INV_VALUE | R/W  | 0h    | The value of the negative increment. |

### 20.2.5.18 TCVR Register (offset = 50h) [reset = 0h]

TCVR is shown in [Figure 20-49](#) and described in [Table 20-52](#).

This register is used for 1ms tick generation. The TCVR register defines whether next value loaded in TCRR will be the sub-period value or the over-period value.

**Figure 20-49. TCVR Register**

|               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31            | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COUNTER_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-52. TCVR Register Field Descriptions**

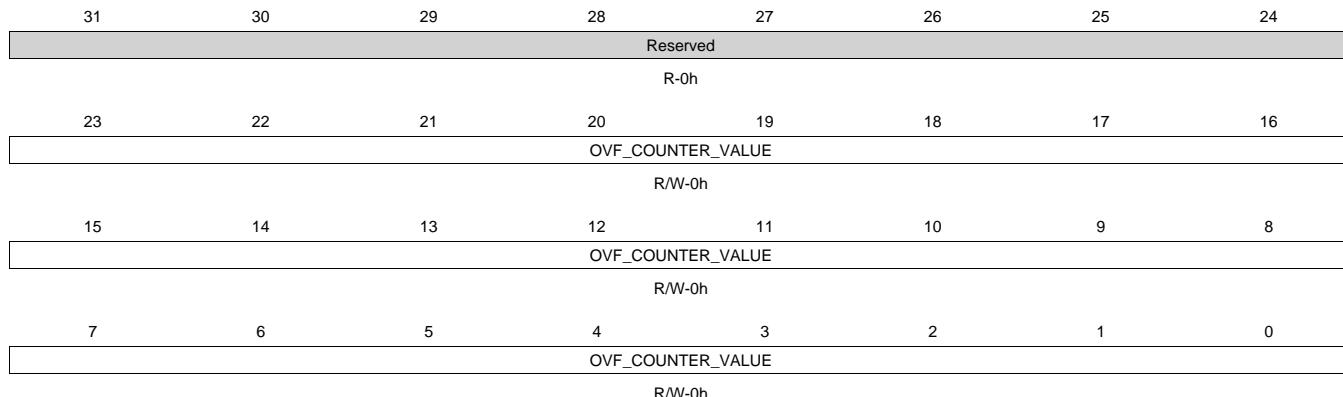
| Bit  | Field         | Type | Reset | Description               |
|------|---------------|------|-------|---------------------------|
| 31-0 | COUNTER_VALUE | R/W  | 0h    | The value of CVR counter. |

### 20.2.5.19 TOCR Register (offset = 54h) [reset = 0h]

TOCR is shown in [Figure 20-50](#) and described in [Table 20-53](#).

This register is used to mask the tick interrupt for a selected number of ticks.

**Figure 20-50. TOCR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-53. TOCR Register Field Descriptions**

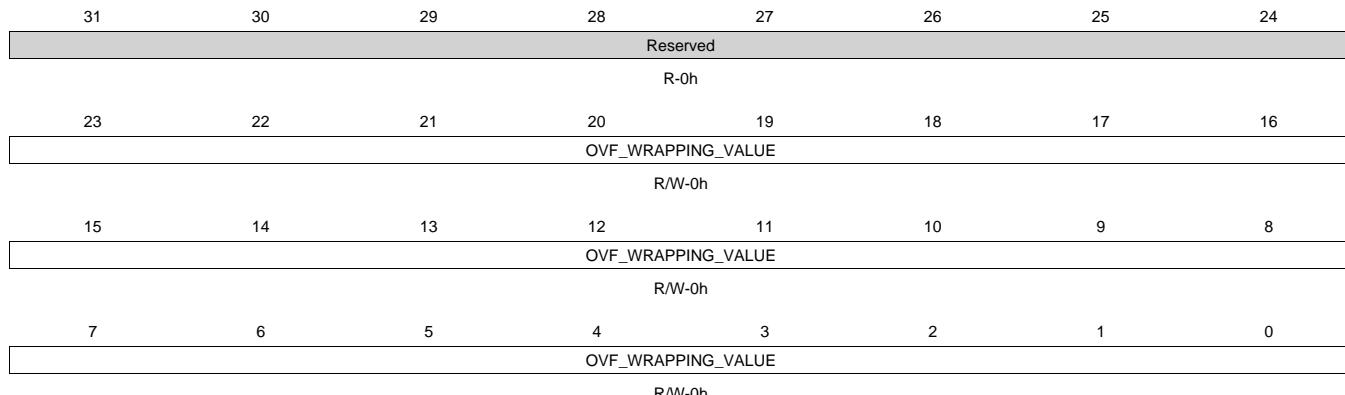
| Bit   | Field             | Type | Reset | Description                    |
|-------|-------------------|------|-------|--------------------------------|
| 31-24 | Reserved          | R    | 0h    | Reads return 0.                |
| 23-0  | OVF_COUNTER_VALUE | R/W  | 0h    | The number of overflow events. |

### 20.2.5.20 TOWR Register (offset = 58h) [reset = 0h]

TOWR is shown in [Figure 20-51](#) and described in [Table 20-54](#).

This register holds the number of masked overflow interrupts.

**Figure 20-51. TOWR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-54. TOWR Register Field Descriptions**

| Bit   | Field              | Type | Reset | Description                      |
|-------|--------------------|------|-------|----------------------------------|
| 31-24 | Reserved           | R    | 0h    | Reads return 0                   |
| 23-0  | OVF_WRAPPING_VALUE | R/W  | 0h    | The number of masked interrupts. |

## 20.3 RTC\_SS

### 20.3.1 Introduction

The real-time clock is a precise timer which can generate interrupts on intervals specified by the user. Interrupts can occur every second, minute, hour, or day. The clock itself can track the passage of real time for durations of several years, provided it has a sufficient power source the whole time.

The basic purpose for the RTC is to keep time of day. The other equally important purpose of RTC is for Digital Rights management. Some degree of tamper proofing is needed to ensure that simply stopping, resetting, or corrupting the RTC does not go unnoticed so that if this occurs, the application can re-acquire the time of day from a trusted source. The final purpose of RTC is to wake the rest of chip up from a power down state.

Alarms are available to interrupt the CPU at a particular time, or at periodic time intervals, such as once per minute or once per day. In addition, the RTC can interrupt the CPU every time the calendar and time registers are updated, or at programmable periodic intervals.

#### 20.3.1.1 Features

The real-time clock (RTC) provides the following features:

- 100-year calendar (xx00 to xx99)
- Counts seconds, minutes, hours, day of the week, date, month, and year with leap year compensation
- Binary-coded-decimal (BCD) representation of time, calendar, and alarm
- 12-hour clock mode (with AM and PM) or 24-hour clock mode
- Alarm interrupt
- Periodic interrupt
- Single interrupt to the CPU
- Supports external 32.768-kHz crystal or external clock source of the same frequency

#### 20.3.1.2 Unsupported RTC Features

This device supports only a single RTC external wake-up event.

## 20.3.2 Integration

This device includes a Real-Time Clock Subsystem (RTCSS) module to allow easy tracking of time and date and the generation of real time alarms.

### 20.3.2.1 RTC Connectivity Attributes

The general connectivity for the RTC module in the device is shown in [Table 20-55](#).

**Table 20-55. RTC Module Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | RTC   |
| Clock Domain        | PD_RTC_L4_RTC_GCLK (Interface/OCP)<br>PD_RTC_RTC32KCLK (Func)<br>CLK_32K_RTC (Func)   |
| Reset Signals       | RTC_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle  |
| Interrupt Requests  | 4 Interrupts<br>Alarm interrupt to MPU Subsystem (RTCINT) and WakeM3<br>Timer interrupt to MPU Subsystem (RTCALARMINT) and WakeM3<br>Alarm wakeup to WakeM3<br>Timer wakeup to WakeM3 |
| DMA Requests        | None  |
| Physical Address    | L4 Wakeup slave port  |

### 20.3.2.2 RTC Clock and Reset Management

The RTC functional clock (c32khz\_clk input) is sourced by default from the CLK32\_KHZ clock derived from the Peripheral PLL. It can also be sourced from the 32-KHz oscillator through a clock mux within and controlled by the RTC\_SS.

**Table 20-56. RTC Clock Signals**

| Clock Signal   | Max Freq   | Reference / Source       | Comments                        |
|--|------------|--------------------------|---------------------------------|
| ocp_clk<br>(Interface clock)                               | 100 MHz    | CORE_CLKOUTM4 / 2        | pd_rtc_l4_rtc_gclk<br>From PRCM |
| rtc_32K_clk_rtc_32k_clk<br>(Oscillator functional clock)   | 32.768 KHz | OSC1_IN                  | CLK_32K_RTC<br>From OSC1_IN     |
| rtc_32k_clk_rtc_32k_aux_clk<br>(Internal functional clock) | 32.768 KHz | PER_CLKOUTM2 / 5859.3752 | pd_rtc_rtc_32kclk<br>From PRCM  |

### 20.3.2.3 RTC Pin List

The RTC module does not include any external interface pins.

**Table 20-57. RTC Pin List**

| Pin                   | Type | Description   |
|-----------------------|------|---|
| RTC_PORZ              | I    | RTC Power On Reset                                    |
| EXT_WAKEUP            | I    | External wakeup                                       |
| PMIC_POWER_EN         | O    | Power enable control for external power management IC |
| <b>Analog Signals</b> |      |   |
| ENZ_KALDO_1P8V        | I    | Enable1.8V LDO  |
| VDDS_RTC              | P    | 1.8V Voltage Supply                                   |

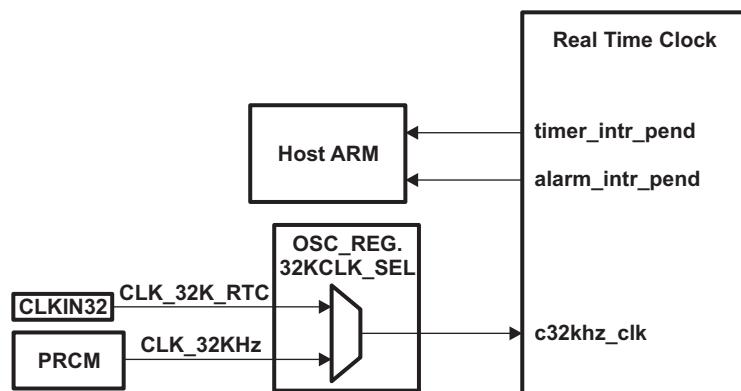
### 20.3.3 Functional Description

This section defines the module interrupt capabilities and requirements.

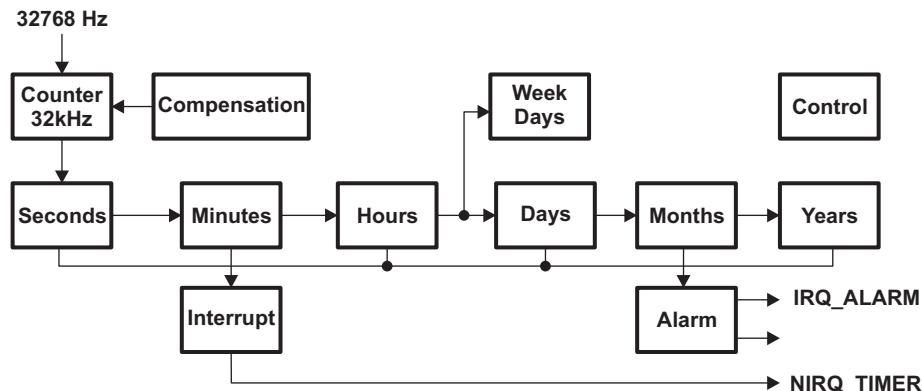
#### 20.3.3.1 Functional Block Diagram

[Figure 20-52](#) shows the RTC module block diagram. [Figure 20-53](#) shows a functional block diagram of the RTC.

**Figure 20-52. RTC Block Diagram**



**Figure 20-53. RTC Functional Block Diagram**



#### 20.3.3.2 Clock Source

The clock reference for the RTC can be sourced from an external crystal (used with the 32K RTC Oscillator), an external 32KHz oscillator, or from the Peripheral PLL. The RTC has an internal oscillator buffer to support direct operation with a crystal. The crystal is connected between pins RTC\_XTALIN and RTC\_XTALOUT. RTC\_XTALIN is the input to the on-chip oscillator and RTC\_XTALOUT is the output from the oscillator back to the crystal. The oscillator can be enabled or disabled by using the RTC\_OSC\_REG register. For more information about the RTC crystal connection, see your device-specific data manual.

An external 32.768-kHz clock oscillator may be used instead of a crystal. In such a case, the clock source is connected to RTC\_XTALIN, and RTC\_XTALOUT is left unconnected.

The source of the 32-KHz clock is selected using the OSC\_CLK.32KCLK\_SEL bit.

If the RTC is not used, the RTC\_XTALIN pin should be held low and RTC\_XTALOUT should be left unconnected. The RTC\_disable bit in the control register (CTRL\_REG) can be set to save power; however, the RTC\_disable bit should not be cleared once it has been set. If the application requires the RTC module to stop and continue, the STOP\_RTC bit in CTRL\_REG should be used instead.

### 20.3.3.3 Signal Descriptions

Table 20-58 lists the signals and their descriptions for the RTC.

**Table 20-58. RTC Signals**

| Signal      | I/O | Description  |
|-------------|-----|--|
| RTC_XTALIN  | I   | RTC time base input signal. RTC_XTALIN can either be driven with a 32.768-kHz reference clock, or RTC_XTALIN and RTC_XTALOUT can be connected to an external crystal. This signal is the input to the RTC internal oscillator. |
| RTC_XTALOUT | O   | RTC time base output signal. RTC_XTALOUT is the output from the RTC internal oscillator. If a crystal is not used as the time base for RTC_XTALIN, RTC_XTALOUT should be left unconnected.                                     |

### 20.3.3.4 Interrupt Support

#### 20.3.3.4.1 CPU Interrupts

The RTC generates two interrupt outputs:

- timer\_intr (RTCINT) is a timer interrupt.
- alarm\_intr (RTCALARMINT) is an alarm interrupt.

---

**NOTE:** Both interrupt outputs support high-level and high-pulse.

---

#### 20.3.3.4.2 Interrupt Description

##### 20.3.3.4.2.1 Timer Interrupt RTCINT (timer\_intr)

The timer interrupt can be generated periodically: every second, every minute, every hour, or every day (see INTERRUPTS\_REG[1:0] for a description of how to set this up). The IT\_TIMER bit of the interrupt register enables this interrupt. The timer interrupt is active-low.

The RTC\_STATUS\_REG[5:2] are only updated at each new interrupt and occur according to [Table 20-59](#). For example, bit 2 (SEC) will always be set when one second has passed. It will also be set when one minute has passed since the completion of one minute also marks the completion of one second (from 59 seconds to 60 seconds). The same holds true for hours and days: each of them will also correspond to the passing of a second.

Conversely, bit 5 (DAY) will always be set when a day has passed. It might also be set when an hour, minute, or second has passed. However, this only occurs when the elapsed hour, minute, or second corresponds to the start of a new day.

**Table 20-59. Interrupt Trigger Events**

|                      | One day has passed | One hour has passed | One minute has passed | One second has passed |
|----------------------|--------------------|---------------------|-----------------------|-----------------------|
| STATUS_REG[5] (DAY)  | 1                  | 0/1 <sup>(1)</sup>  | 0/1 <sup>(1)</sup>    | 0/1 <sup>(1)</sup>    |
| STATUS_REG[4] (HOUR) | 1                  | 1                   | 0/1 <sup>(1)</sup>    | 0/1 <sup>(1)</sup>    |
| STATUS_REG[3] (MIN)  | 1                  | 1                   | 1                     | 0/1 <sup>(1)</sup>    |
| STATUS_REG[2] (SEC)  | 1                  | 1                   | 1                     | 1                     |

<sup>(1)</sup> This event is only triggered when the elapsed time unit (for example, Day) corresponds to the passage of another unit (for example, Seconds). For example, when the clock ticks from 00:23:59:59 (days : hours : minutes : seconds) to 01:00:00:00.

##### 20.3.3.4.2.2 Alarm Interrupt RTCALARMINT (alarm\_intr)

The alarm interrupt can be generated when the time set into TC ALARM registers is exactly the same as in the TC registers. This interrupt is then generated if the IT\_ALARM bit of the interrupts register is set. This interrupt is low-level sensitive. RTC\_STATUS\_REG[6] indicates that IRQ\_ALARM\_CHIP has occurred. This interrupt is disabled by writing '1' into the RTC\_STATUS\_REG[6].

To set up an alarm:

- Modify the ALARM\_SECONDS, ALARM\_MINUTES, ALARM\_HOURS, ALARM\_DAY, ALARM\_WEEK, ALARM\_MONTH, and ALARM\_YEAR registers to the exact time you want an alarm to generate.
- Set the IT\_ALARM bit in the RTC\_INTERRUPTS register to enable the alarm interrupt.

### 20.3.3.5 Programming/Usage Guide

#### 20.3.3.5.1 Time/Calendar Data Format

The time and calendar data in the RTC is stored as binary-coded decimal (BCD) format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Although most of the time/calendar registers have 4 bits assigned to each BCD digit, some of the register fields are shorter since the range of valid numbers may be limited. For example, only 3 bits are required to represent the day of the week (WEEKS\_REG) since only BCD numbers 1 through 7 are required. The following time and calendar registers are supported (BCD Format):

Note that the ALARM registers which share the names above also share the same BCD formatting.

- SECOND - Second Count (00-59)
- MINUTE - Minute Count (00-59)
- HOUR - Hour Count (12HR: 01-12; 24HR: 00-23)
- DAY - Day of the Month Count (01-31)
- WEEK - Day of the Week (0-6: SUN = 0)
- MONTH - Month Count (01-12; JAN = 1)
- YEAR - Year Count (00-99)

#### 20.3.3.5.2 Register Access

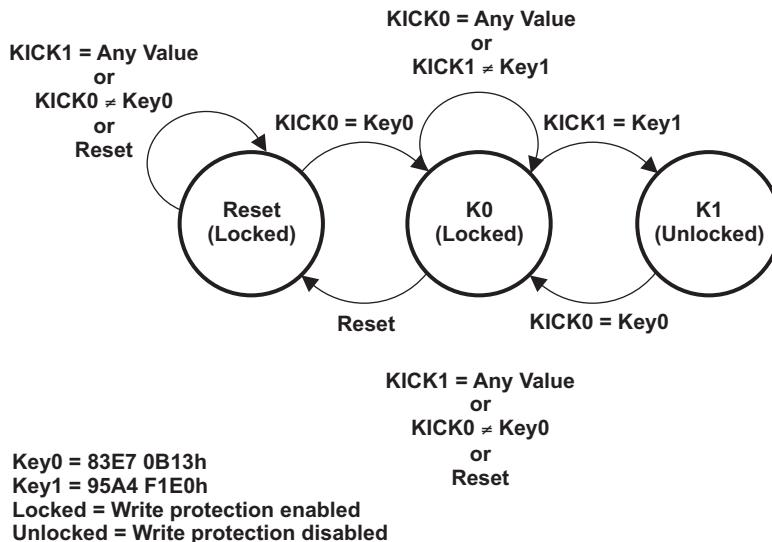
The three register types are as follows and each has its own access constraints:

- TC and TC alarm registers
- General registers
- Compensation registers

#### 20.3.3.5.3 OCP MMR Spurious WRT Protection

The module also contains a kicker mechanism (Figure 20-54) to prevent any spurious writes from changing the register values. This mechanism requires two MMR writes to the Kick0 and Kick1 registers with exact data values before the kicker lock mechanism is released. Once released, the MMRs are writeable. The Kick0 data is 83E7 0B13h; the Kick1 data is 95A4 F1E0h. Note that it remains in an unlocked state until an OCP reset or invalid data pattern is written to one of the Kick0 or Kick1 registers.

**Figure 20-54. Kick Register State Machine Diagram**



- S0 is the Reset/Idle state
- S1 is an OCP wrt cycle of 83E7 0B13h at Kick0 completed state
- S2 is the UNLOCK MMR WRT state
- S0 -> S1 when OCP wrt cycle of 83E7 0B13h at Kick0
- S1 -> S2 when OCP wrt cycle of 95A4 F1E0h at Kick1
- S1 -> S0 when OCP reset event
- S2 -> S0 when OCP reset event OR OCP wrt cycle of NOT 83E7 0B13h at Kick0 OR OCP wrt cycle at Kick1
- S2 -> S1 when OCP wrt cycle of 83E7 0B13h at Kick0

#### **20.3.3.5.4 Reading the Timer/Calendar (TC) Registers**

The TC registers have a read-show register. The reading of the Seconds register will update all of the TC registers. For example, the Year will only get updated on a reading of the Seconds register. The time/calendar registers are updated every second as the time changes. During a read of the SECONDS register, the RTC copies the current values of the time/date registers into shadow read registers. This isolation assures that the CPU can capture all the time/date values at the moment of the SECONDS read request and not be subject to changing register values from time updates.

If desired, the RTC also provides a one-time-triggered minute-rounding feature to round the MINUTE:SECOND registers to the nearest minute (with zero seconds). This feature is enabled by setting the ROUND\_30S bit in the control register (CTRL); the RTC automatically rounds the time values to the nearest minute upon the next read of the SECONDS register.

---

**NOTE:** Software should always read the Seconds register first. However, the software does not have to poll any status bit to determine when to read the TC registers. [Table 20-60](#) defines the TC set that gets shadowed.

---

**Table 20-60. RTC Register Names and Values**

| Time Unit | Range    | Remarks                      |
|-----------|----------|------------------------------|
| Year      | 00 to 99 |                              |
| Month     | 01 to 12 |                              |
| Day       | 01 to 31 | Months 1, 3, 5, 7, 8, 10, 12 |
|           | 01 to 30 | Months 4, 6, 9, 11           |
|           | 01 to 29 | Month 2 (leap year)          |
|           | 01 to 28 | Month 2 (common year)        |
| Week      | 00 to 06 | Day of week                  |
| Hour      | 00 to 23 | 24 hour mode                 |
|           | 01 to 12 | AM/PM mode                   |
| Minute    | 00 to 59 |                              |
| Seconds   | 00 to 59 |                              |

#### **20.3.3.5.4.1 Rounding Seconds**

Time can be rounded to the closest minute, by setting the ROUND\_30S bit of the control register. When this bit is set, TC values are set to the closest minute value at the next second. The ROUND\_30S bit will be automatically cleared when rounding time is performed.

**Example:**

- If current time is 10H59M45S, round operation will change time to 11H00M00S.
- If current time is 10H59M29S, round operation will change time to 10H59M00S.

### 20.3.3.5.5 Modifying the TC Registers

To write correct data from/to the TC and TC alarm registers and read the TC alarm registers, the ARM must first read the BUSY bit of the STATUS register until BUSY is equal to zero. Once the BUSY flag is zero, there is a 15  $\mu$ s access period in which the ARM can program the TC and TC alarm registers. Once the 15  $\mu$ s access period passes, the BUSY flag has to be read again from the STATUS register as described previously. If the ARM accesses the TC registers outside of the access period, then the access is not guaranteed.

The ARM can access the STATUS\_REG and CTRL\_REG at any time, with the exception of CTRL\_REG[5] which can only be changed when the RTC is stopped. The ARM can stop the RTC by clearing the STOP\_RTC bit of the control register. After clearing this bit, the RUN bit in the STATUS\_REG (bit 1) needs to be checked to verify the RTC has in fact stopped. Once this is confirmed, the TC values can be updated. After the values have been updated, the RTC can be re-started by resetting the STOP\_RTC bit.

---

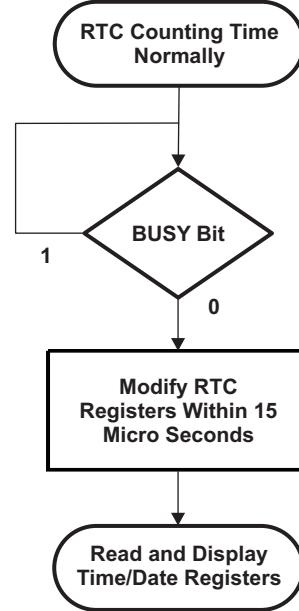
**NOTE:** After writing to a TC register, the user must wait 4 OCP clock cycles before reading the value from the register. If this wait time is not observed and the TC register is accessed, then old data will be read from the register.

---

#### CAUTION

In order to remove any possibility of interrupting the register's read process, thus introducing a potential risk of violating the authorized 15-microsecond access period, it is strongly recommended that you disable all incoming interrupts during the register read process.

Figure 20-55. Flow Control for Updating RTC Registers



### 20.3.3.5.5.1 General Registers

The ARM can access the STATUS\_REG and the CTRL\_REG at any time (except the CTRL\_REG[5] bit which must be changed only when the RTC is stopped). For the INTERRUPTS\_REG, the ARM should respect the available access period to prevent spurious interrupt.

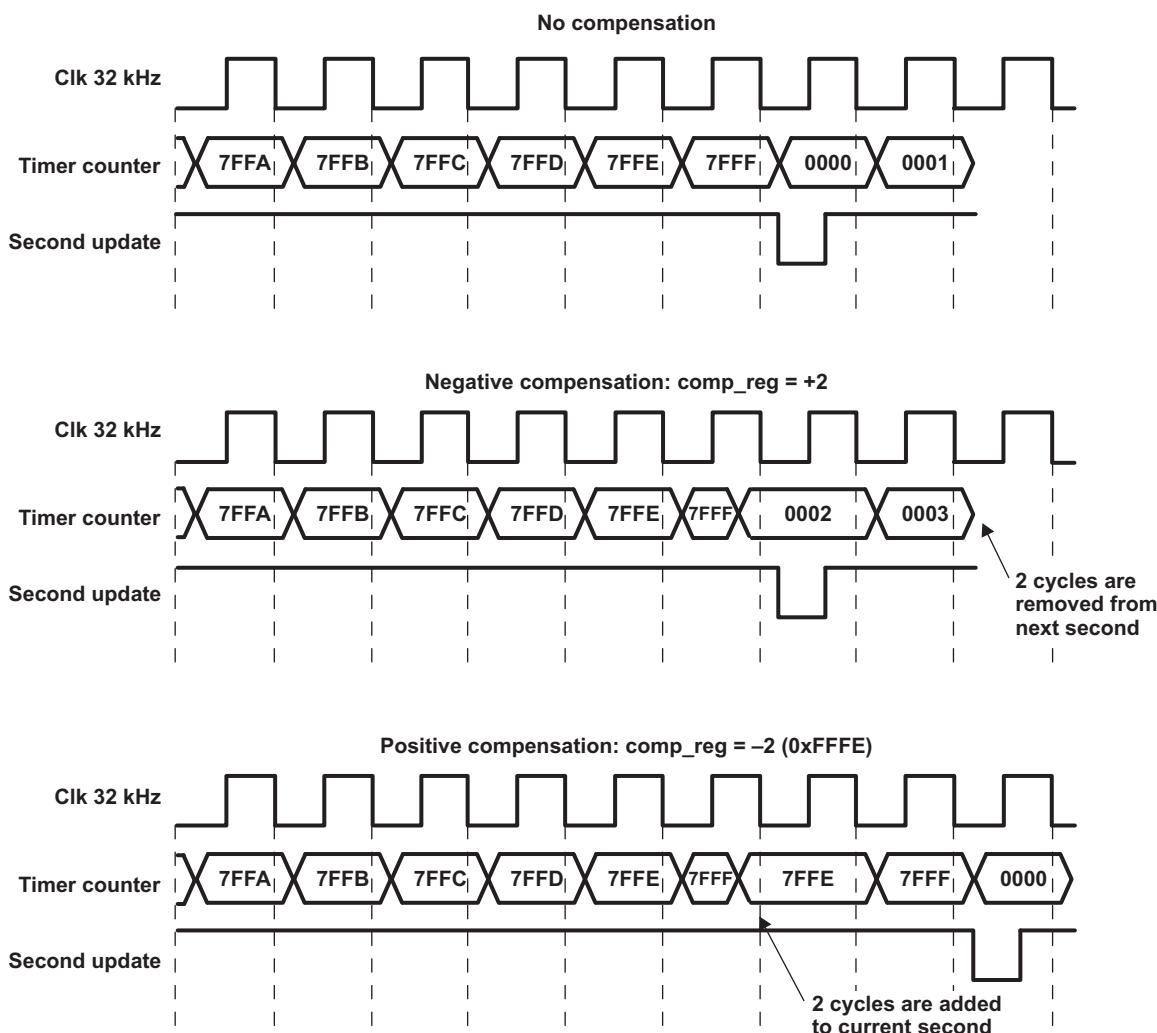
The RTC\_DISABLE bit of the CTRL register must only be used to completely disable the RTC function. When this bit is set, the 32 kHz clock is gated, and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. In order to save power, this bit should only be used if the RTC function is unwanted in the application.

#### 20.3.3.5.6 Crystal Compensation

To compensate for any inaccuracy of the 32 kHz oscillator, the ARM can perform a calibration of the oscillator frequency, calculate the drift compensation versus one-hour period, and load the compensation registers with the drift compensation value. Auto compensation is enabled by AUTO\_COMP\_EN bit in the RTC\_CTRL register. If the COMP\_REG value is positive, compensation occurs after the second change event. COMP\_REG cycles are removed from the next second. If the COMP\_REG value is negative, compensation occurs before the second change event. COMP\_REG cycles are added to the current second. This enables compensation with a 1 32-kHz period accuracy each hour. The waveform below summarizes positive and negative compensation effect.

Access to the COMP\_MSB\_REG and COMP\_LSB\_REG registers must respect the available access period. These registers should not be updated during compensation (first second of each hour), but it is alright to update them during the second preceding a compensation event. For example, the ARM could load the compensation value into these registers after each hour event, during an available access period.

**Figure 20-56. Compensation Illustration**



### 20.3.3.6 Scratch Registers

The RTC provides three general-purpose registers (SCRATCHx\_REG) that can be used to store 32-bit words -- these registers have no functional purpose for the RTC. Software using the RTC may find the SCRATCHx registers to be useful in indicating RTC states. For example, the SCRATCHx\_REG registers may be used to indicate write-protection lock status or unintentional power downs. To indicate write-protection, the software should write a unique value to one of the SCRATCHx\_REG registers when write-protection is disabled and another unique value when write-protection is enabled again. In this way, the lock-status of the registers can be determined quickly by reading the SCRATCH register. To indicate unintentional power downs, the software should write a unique value to one of the SCRATCHx\_REG registers when RTC is configured and enabled. If the RTC is unintentionally powered down, the value written to the SCRATCH register is cleared. For more information, see [Section 20.3.5, RTC Registers](#).

### 20.3.3.7 Power Management

The RTC supports the power idle protocol. It has two SWakeup ports: one for the alarm event and one for a timer event.

When the RTC is in IDLE mode, the OCP clock is turned off and the 32 kHz clock remains on. The time and calendar continue to count in IDLE mode. When the RTC is placed back in FUNCTIONAL mode, the TC registers can be read.

The Alarm SWakeup event can be used to wakeup the RTC when it is in IDLE state. In order to do so, the alarm needs to be set and enabled before RTC enters the IDLE state. Once this is done, the SWakeup will occur when the alarm event triggers.

---

**NOTE:** Since SWakeup is not periodic, using it to wake up the RTC when in IDLE state is not recommended. Please use Alarm SWakeup instead.

---

### 20.3.3.8 Power Management—System Level (PMIC Mode)

The RTC generates pmic\_power\_en control which can be used to control an external PMIC.

**Table 20-61. pmic\_power\_en Description**

| Port          | Direction | Function  |
|---------------|-----------|---|
| rtc_pwrnrstn  | Input     | Not optional.<br>RTC true power on domain reset.<br>Only assert when RTC has lost power.<br>Always de-assert when RTC voltage is greater than Vmin.<br>The port remains de-asserted during normal operations. |
| pmic_power_en | Output    | Optional.<br>Can be used to control an external PMIC.<br>0 = OFF<br>1 = ON (reset state)<br>ON → OFF (Turn OFF)<br>By ALARM2 event<br>OFF → ON (Turn ON)<br>By ALARM event OR ext_wakeup event                |
| ext_wakeup    | Input     |   |

### 20.3.4 Use Cases

The following list includes high-level steps to start using the RTC:

1. Enable the module clock domains (see the Integration section for details on which clock domain).
2. Enable the RTC module using CTRL\_REG.RTC\_disable.
3. Enable the 32K clock from PER PLL.
4. Write to the kick registers (KICK0R, KICK1R) in the RTC.
5. Do the required timer setup in the RTCSS.
6. Start the RTC (in CTRL\_REG.STOP\_RTC).

### 20.3.5 RTC Registers

Table 20-62 lists the memory-mapped registers for the RTC. All register offset addresses not listed in Table 20-62 should be considered as reserved locations and the register contents should not be modified.

**Table 20-62. RTC REGISTERS**

| Offset | Acronym            | Register Name                        | Section                           |
|--------|--------------------|--------------------------------------|-----------------------------------|
| 0h     | SECONDS_REG        | Seconds Register                     | <a href="#">Section 20.3.5.1</a>  |
| 4h     | MINUTES_REG        | Minutes Register                     | <a href="#">Section 20.3.5.2</a>  |
| 8h     | HOURS_REG          | Hours Register                       | <a href="#">Section 20.3.5.3</a>  |
| Ch     | DAYS_REG           | Day of the Month Register            | <a href="#">Section 20.3.5.4</a>  |
| 10h    | MONTHS_REG         | Month Register                       | <a href="#">Section 20.3.5.5</a>  |
| 14h    | YEARS_REG          | Year Register                        | <a href="#">Section 20.3.5.6</a>  |
| 18h    | WEEKS_REG          | Day of the Week Register             | <a href="#">Section 20.3.5.7</a>  |
| 20h    | ALARM_SECONDS_REG  | Alarm Seconds Register               | <a href="#">Section 20.3.5.8</a>  |
| 24h    | ALARM_MINUTES_REG  | Alarm Minutes Register               | <a href="#">Section 20.3.5.9</a>  |
| 28h    | ALARM_HOURS_REG    | Alarm Hours Register                 | <a href="#">Section 20.3.5.10</a> |
| 2Ch    | ALARM_DAYS_REG     | Alarm Day of the Month Register      | <a href="#">Section 20.3.5.11</a> |
| 30h    | ALARM_MONTHS_REG   | Alarm Months Register                | <a href="#">Section 20.3.5.12</a> |
| 34h    | ALARM_YEARS_REG    | Alarm Years Register                 | <a href="#">Section 20.3.5.13</a> |
| 40h    | RTC_CTRL_REG       | Control Register                     | <a href="#">Section 20.3.5.14</a> |
| 44h    | RTC_STATUS_REG     | Status Register                      | <a href="#">Section 20.3.5.15</a> |
| 48h    | RTC_INTERRUPTS_REG | Interrupt Enable Register            | <a href="#">Section 20.3.5.16</a> |
| 4Ch    | RTC_COMP_LSB_REG   | Compensation (LSB) Register          | <a href="#">Section 20.3.5.17</a> |
| 50h    | RTC_COMP_MSB_REG   | Compensation (MSB) Register          | <a href="#">Section 20.3.5.18</a> |
| 54h    | RTC_OSC_REG        | Oscillator Register                  | <a href="#">Section 20.3.5.19</a> |
| 60h    | RTC_SCRATCH0_REG   | Scratch 0 Register (General-Purpose) | <a href="#">Section 20.3.5.20</a> |
| 64h    | RTC_SCRATCH1_REG   | Scratch 1 Register (General-Purpose) | <a href="#">Section 20.3.5.21</a> |
| 68h    | RTC_SCRATCH2_REG   | Scratch 2 Register (General-Purpose) | <a href="#">Section 20.3.5.22</a> |
| 6Ch    | KICK0R             | Kick 0 Register (Write Protect)      | <a href="#">Section 20.3.5.23</a> |
| 70h    | KICK1R             | Kick 1 Register (Write Protect)      | <a href="#">Section 20.3.5.24</a> |
| 74h    | RTC_REVISION       | Revision Register                    | <a href="#">Section 20.3.5.25</a> |
| 78h    | RTC_SYSCONFIG      | System Configuration Register        | <a href="#">Section 20.3.5.26</a> |
| 7Ch    | RTC_IRQWAKEEN      | Wakeup Enable Register               | <a href="#">Section 20.3.5.27</a> |
| 80h    | ALARM2_SECONDS_REG | Alarm2 Seconds Register              | <a href="#">Section 20.3.5.28</a> |
| 84h    | ALARM2_MINUTES_REG | Alarm2 Minutes Register              | <a href="#">Section 20.3.5.29</a> |
| 88h    | ALARM2_HOURS_REG   | Alarm2 Hours Register                | <a href="#">Section 20.3.5.30</a> |
| 8Ch    | ALARM2_DAYS_REG    | Alarm2 Day of the Month Register     | <a href="#">Section 20.3.5.31</a> |
| 90h    | ALARM2_MONTHS_REG  | Alarm2 Months Register               | <a href="#">Section 20.3.5.32</a> |
| 94h    | ALARM2_YEARS_REG   | Alarm2 Years Register                | <a href="#">Section 20.3.5.33</a> |
| 98h    | RTC_PMIC           | RTC PMIC Register                    | <a href="#">Section 20.3.5.34</a> |

**Table 20-62. RTC REGISTERS (continued)**

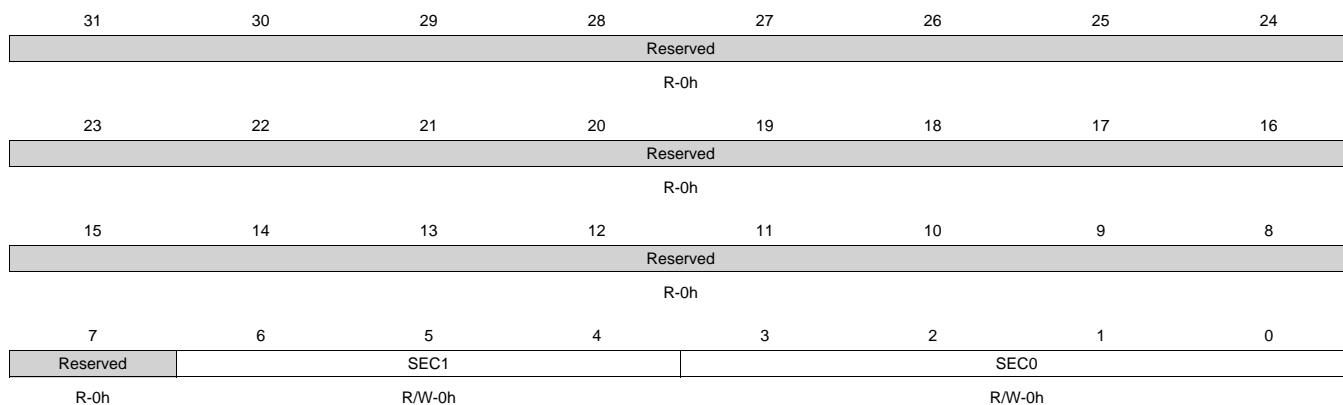
| Offset | Acronym      | Register Name         | Section                           |
|--------|--------------|-----------------------|-----------------------------------|
| 9Ch    | RTC_DEBOUNCE | RTC Debounce Register | <a href="#">Section 20.3.5.35</a> |

### 20.3.5.1 SECONDS\_REG Register (offset = 0h) [reset = 0h]

SECONDS\_REG is shown in [Figure 20-57](#) and described in [Table 20-63](#).

The SECONDS\_REG is used to program the required seconds value of the current time. Seconds are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the seconds value is 45, then the value of SEC0 is 5 and value of SEC1 is 4.

**Figure 20-57. SECONDS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-63. SECONDS\_REG Register Field Descriptions**

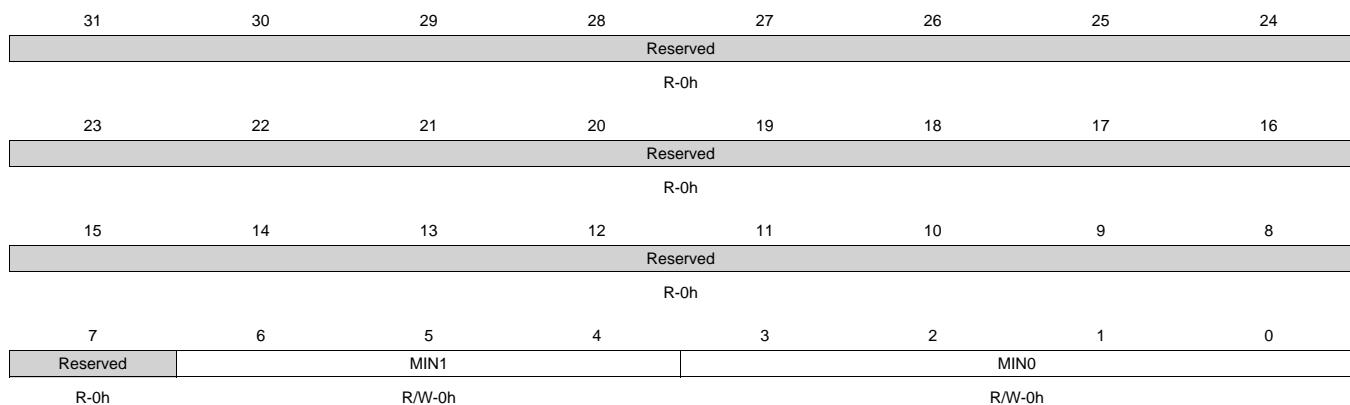
| Bit  | Field    | Type | Reset | Description                           |
|------|----------|------|-------|---------------------------------------|
| 31-7 | Reserved | R    | 0h    |                                       |
| 6-4  | SEC1     | R/W  | 0h    | 2nd digit of seconds, Range is 0 to 5 |
| 3-0  | SEC0     | R/W  | 0h    | 1st digit of seconds, Range is 0 to 9 |

### 20.3.5.2 MINUTES\_REG Register (offset = 4h) [reset = 0h]

MINUTES\_REG is shown in Figure 20-58 and described in Table 20-64.

The MINUTES\_REG is used to program the minutes value of the current time. Minutes are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the minutes value is 32, then the value of MIN0 is 2 and value of MIN1 is 3.

**Figure 20-58. MINUTES\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-64. MINUTES\_REG Register Field Descriptions**

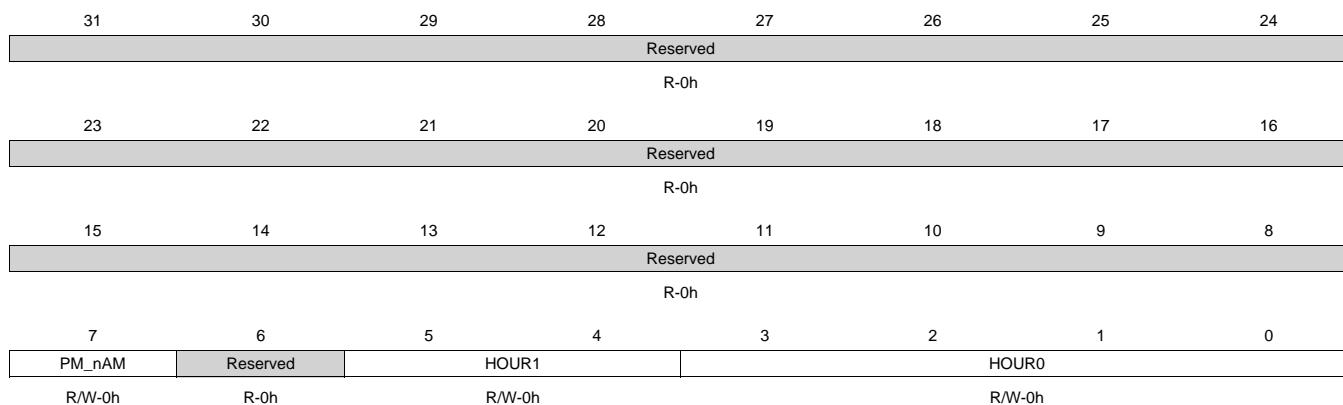
| Bit  | Field    | Type | Reset | Description                           |
|------|----------|------|-------|---------------------------------------|
| 31-7 | Reserved | R    | 0h    |                                       |
| 6-4  | MIN1     | R/W  | 0h    | 2nd digit of minutes, Range is 0 to 5 |
| 3-0  | MIN0     | R/W  | 0h    | 1st digit of minutes, Range is 0 to 9 |

### 20.3.5.3 HOURS\_REG Register (offset = 8h) [reset = 0h]

HOURS\_REG is shown in [Figure 20-59](#) and described in [Table 20-65](#).

The HOURS\_REG is used to program the hours value of the current time. Hours are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. In 24Hr time mode if you want to set the hour as 18, then HOUR0 is set as 8 and HOUR1 is set as 1.

**Figure 20-59. HOURS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-65. HOURS\_REG Register Field Descriptions**

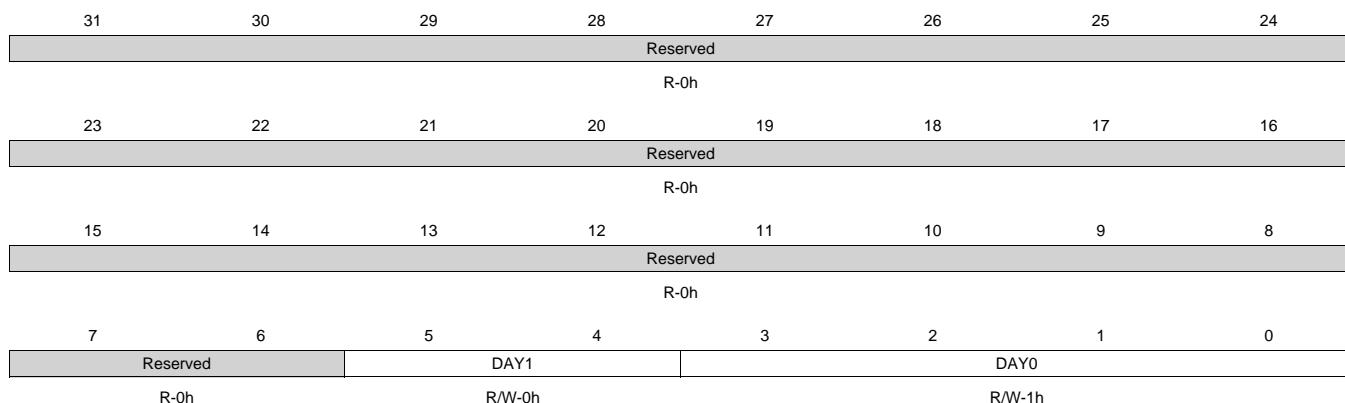
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7    | PM_nAM   | R/W  | 0h    | Only used in PM_AM mode (otherwise 0)<br>0x0 = AM<br>0x1 = PM |
| 6    | Reserved | R    | 0h    |   |
| 5-4  | HOUR1    | R/W  | 0h    | 2nd digit of hours, Range is 0 to 2                           |
| 3-0  | HOUR0    | R/W  | 0h    | 1st digit of hours, Range is 0 to 9                           |

#### 20.3.5.4 DAYS\_REG Register (offset = Ch) [reset = 1h]

DAYS\_REG is shown in [Figure 20-60](#) and described in [Table 20-66](#).

The DAYS\_REG is used to program the day of the month value of the current date. Days are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. If the day value of the date is 28, DAY0 is set as 8 and DAY1 is set as 2.

**Figure 20-60. DAYS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-66. DAYS\_REG Register Field Descriptions**

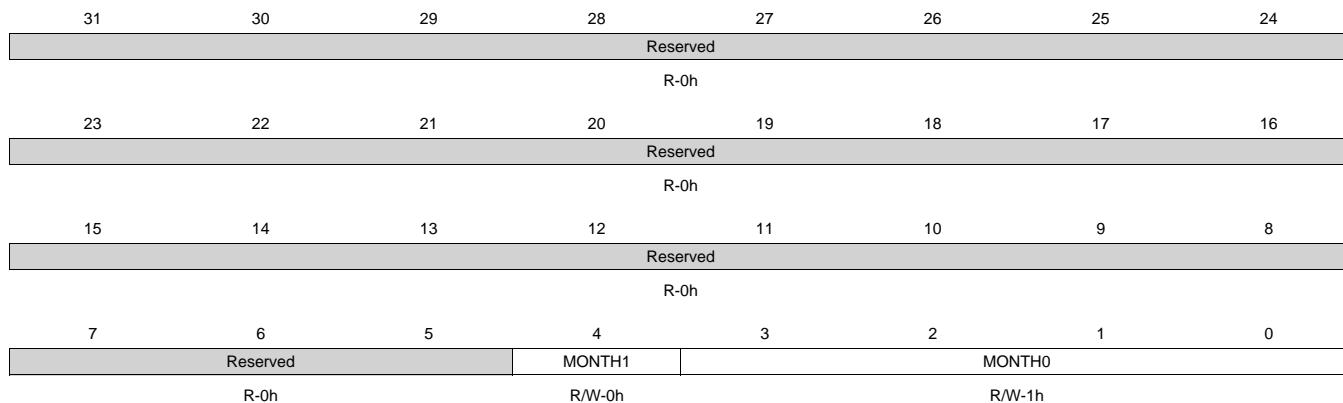
| Bit  | Field    | Type | Reset | Description                        |
|------|----------|------|-------|------------------------------------|
| 31-6 | Reserved | R    | 0h    |                                    |
| 5-4  | DAY1     | R/W  | 0h    | 2nd digit of days, Range is 0 to 3 |
| 3-0  | DAY0     | R/W  | 1h    | 1st digit of days, Range is 0 to 9 |

### 20.3.5.5 MONTHS\_REG Register (offset = 10h) [reset = 1h]

MONTHS\_REG is shown in [Figure 20-61](#) and described in [Table 20-67](#).

The MONTHS\_REG is used to set the month in the year value of the current date. Months are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Usual notation is taken for month value: 1 = January, 2 = February, continuing until 12 = December.

**Figure 20-61. MONTHS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-67. MONTHS\_REG Register Field Descriptions**

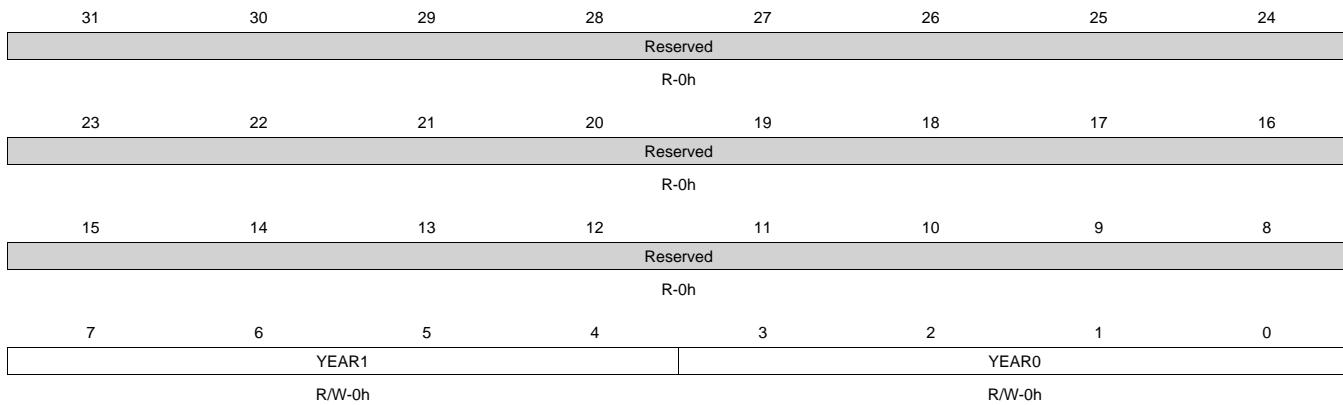
| Bit  | Field    | Type | Reset | Description                          |
|------|----------|------|-------|--------------------------------------|
| 31-5 | Reserved | R    | 0h    |                                      |
| 4    | MONTH1   | R/W  | 0h    | 2nd digit of months, Range is 0 to 1 |
| 3-0  | MONTH0   | R/W  | 1h    | 1st digit of months, Range is 0 to 9 |

### 20.3.5.6 YEARS\_REG Register (offset = 14h) [reset = 0h]

YEARS\_REG is shown in [Figure 20-62](#) and described in [Table 20-68](#).

The YEARS\_REG is used to program the year value of the current date. The year value is represented by only the last 2 digits and is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. The year 1979 is programmed as 79 with YEAR0 set as 9 and YEAR1 set as 7.

**Figure 20-62. YEARS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-68. YEARS\_REG Register Field Descriptions**

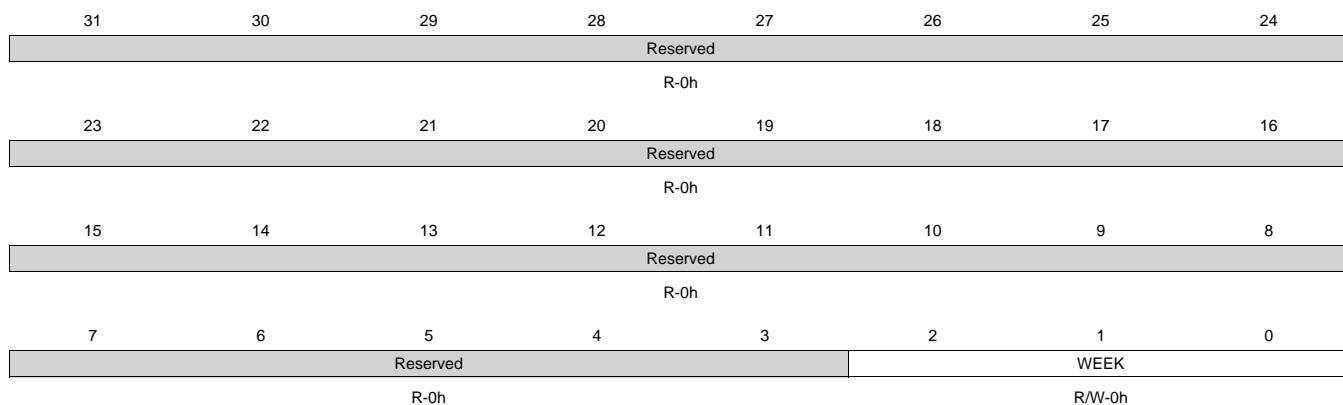
| Bit  | Field    | Type | Reset | Description                         |
|------|----------|------|-------|-------------------------------------|
| 31-8 | Reserved | R    | 0h    |                                     |
| 7-4  | YEAR1    | R/W  | 0h    | 2nd digit of years, Range is 0 to 9 |
| 3-0  | YEAR0    | R/W  | 0h    | 1st digit of years, Range is 0 to 9 |

### 20.3.5.7 WEEKS\_REG Register (offset = 18h) [reset = 0h]

WEEKS\_REG is shown in [Figure 20-63](#) and described in [Table 20-69](#).

The WEEKS\_REG is used to program the day of the week value of the current date. The day of the week is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent. Sunday is treated as 0, Monday 1, and ending at Saturday with 6.

**Figure 20-63. WEEKS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-69. WEEKS\_REG Register Field Descriptions**

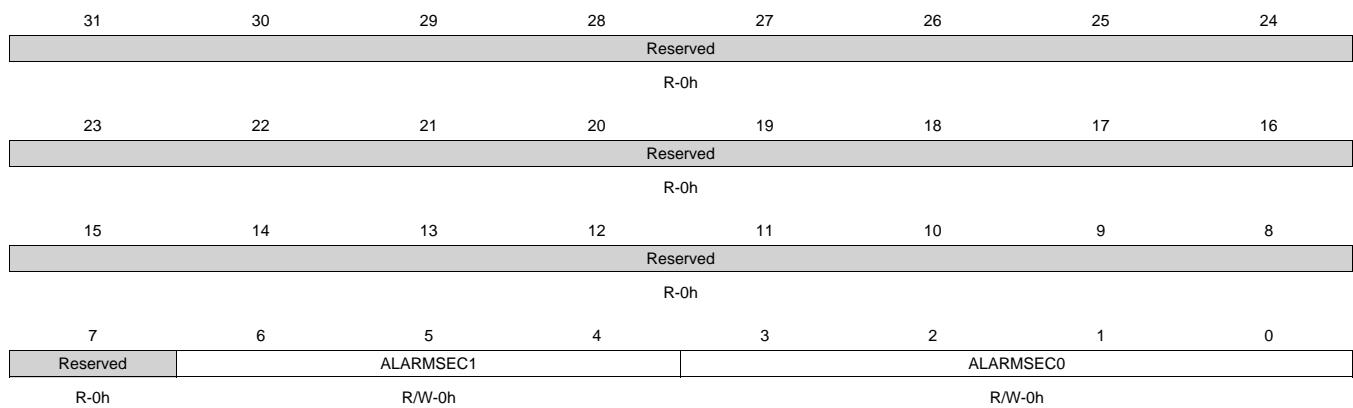
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-3 | Reserved | R    | 0h    |  |
| 2-0  | WEEK     | R/W  | 0h    | 1st digit of days in a week, Range from 0 (Sunday) to 6 (Saturday) |

### 20.3.5.8 ALARM\_SECONDS\_REG Register (offset = 20h) [reset = 0h]

ALARM\_SECONDS\_REG is shown in [Figure 20-64](#) and described in [Table 20-70](#).

The ALARM\_SECONDS\_REG is used to program the second value for the alarm interrupt. Seconds are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-64. ALARM\_SECONDS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-70. ALARM\_SECONDS\_REG Register Field Descriptions**

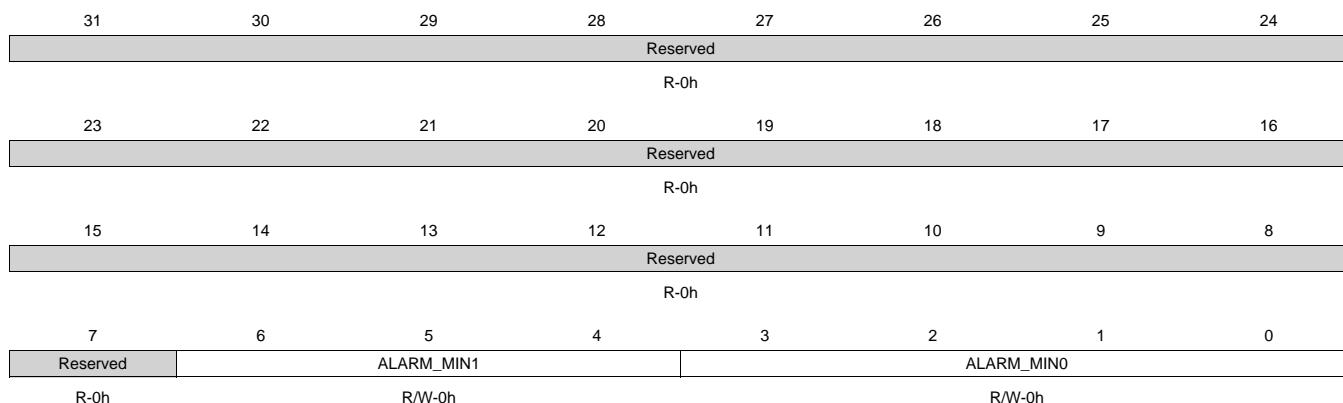
| Bit  | Field     | Type | Reset | Description                           |
|------|-----------|------|-------|---------------------------------------|
| 31-7 | Reserved  | R    | 0h    |                                       |
| 6-4  | ALARMSEC1 | R/W  | 0h    | 2nd digit of seconds, Range is 0 to 5 |
| 3-0  | ALARMSEC0 | R/W  | 0h    | 1st digit of seconds, Range is 0 to 9 |

### 20.3.5.9 ALARM\_MINUTES\_REG Register (offset = 24h) [reset = 0h]

ALARM\_MINUTES\_REG is shown in [Figure 20-65](#) and described in [Table 20-71](#).

The ALARM\_MINUTES\_REG is used to program the minute value for the alarm interrupt. Minutes are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-65. ALARM\_MINUTES\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-71. ALARM\_MINUTES\_REG Register Field Descriptions**

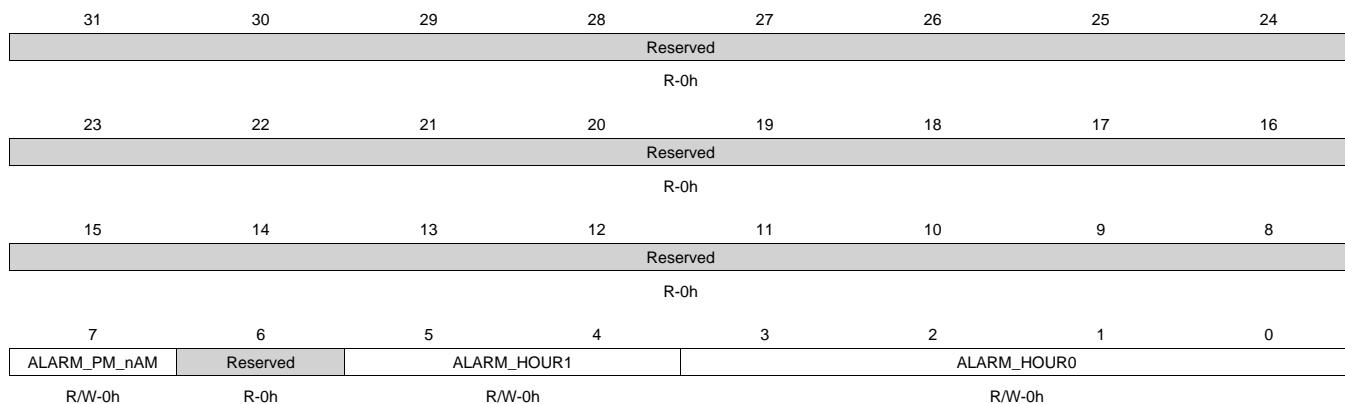
| Bit  | Field      | Type | Reset | Description                           |
|------|------------|------|-------|---------------------------------------|
| 31-7 | Reserved   | R    | 0h    |                                       |
| 6-4  | ALARM_MIN1 | R/W  | 0h    | 2nd digit of minutes, Range is 0 to 5 |
| 3-0  | ALARM_MIN0 | R/W  | 0h    | 1st digit of minutes, Range is 0 to 9 |

### 20.3.5.10 ALARM\_HOURS\_REG Register (offset = 28h) [reset = 0h]

ALARM\_HOURS\_REG is shown in [Figure 20-66](#) and described in [Table 20-72](#).

The ALARM\_HOURS\_REG is used to program the hour value for the alarm interrupt. Hours are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-66. ALARM\_HOURS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-72. ALARM\_HOURS\_REG Register Field Descriptions**

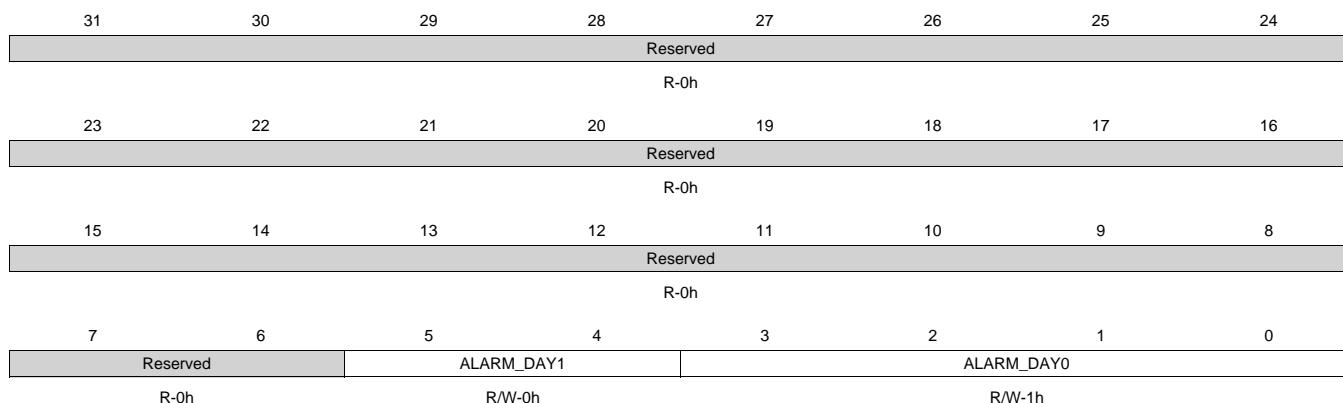
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-8 | Reserved     | R    | 0h    |   |
| 7    | ALARM_PM_nAM | R/W  | 0h    | Only used in PM_AM mode (otherwise 0)<br>0x0 = AM<br>0x1 = PM |
| 6    | Reserved     | R    | 0h    |   |
| 5-4  | ALARM_HOUR1  | R/W  | 0h    | 2nd digit of hours, Range is 0 to 2                           |
| 3-0  | ALARM_HOUR0  | R/W  | 0h    | 1st digit of hours, Range is 0 to 9                           |

### 20.3.5.11 ALARM\_DAYS\_REG Register (offset = 2Ch) [reset = 1h]

ALARM\_DAYS\_REG is shown in [Figure 20-67](#) and described in [Table 20-73](#).

The ALARM\_DAYS\_REG is used to program the day of the month value for the alarm interrupt. Days are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-67. ALARM\_DAYS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-73. ALARM\_DAYS\_REG Register Field Descriptions**

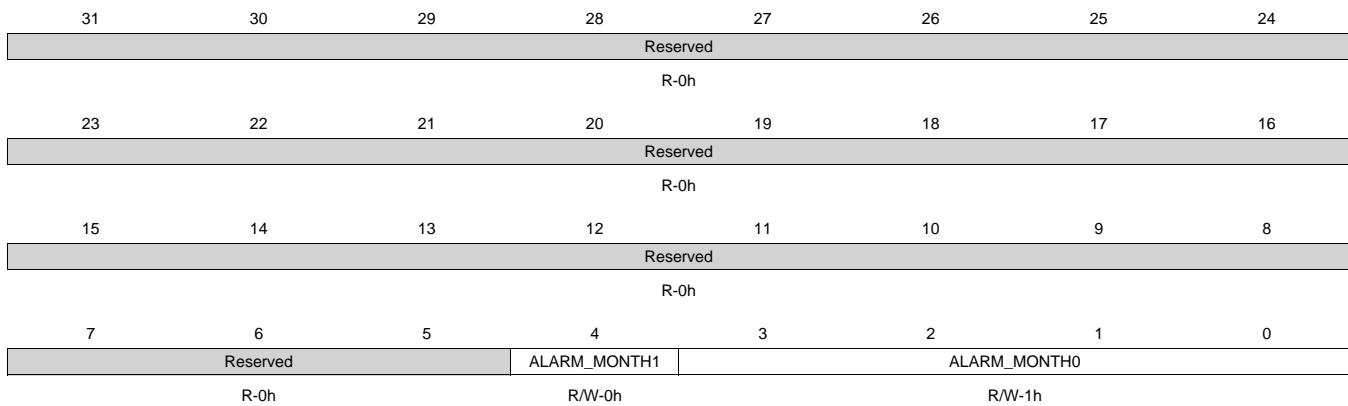
| Bit  | Field      | Type | Reset | Description                           |
|------|------------|------|-------|---------------------------------------|
| 31-6 | Reserved   | R    | 0h    |                                       |
| 5-4  | ALARM_DAY1 | R/W  | 0h    | 2nd digit for days, Range from 0 to 3 |
| 3-0  | ALARM_DAY0 | R/W  | 1h    | 1st digit for days, Range from 0 to 9 |

### 20.3.5.12 ALARM\_MONTHS\_REG Register (offset = 30h) [reset = 1h]

ALARM\_MONTHS\_REG is shown in [Figure 20-68](#) and described in [Table 20-74](#).

The ALARM\_MONTHS\_REG is used to program the month in the year value for the alarm interrupt. The month is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-68. ALARM\_MONTHS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-74. ALARM\_MONTHS\_REG Register Field Descriptions**

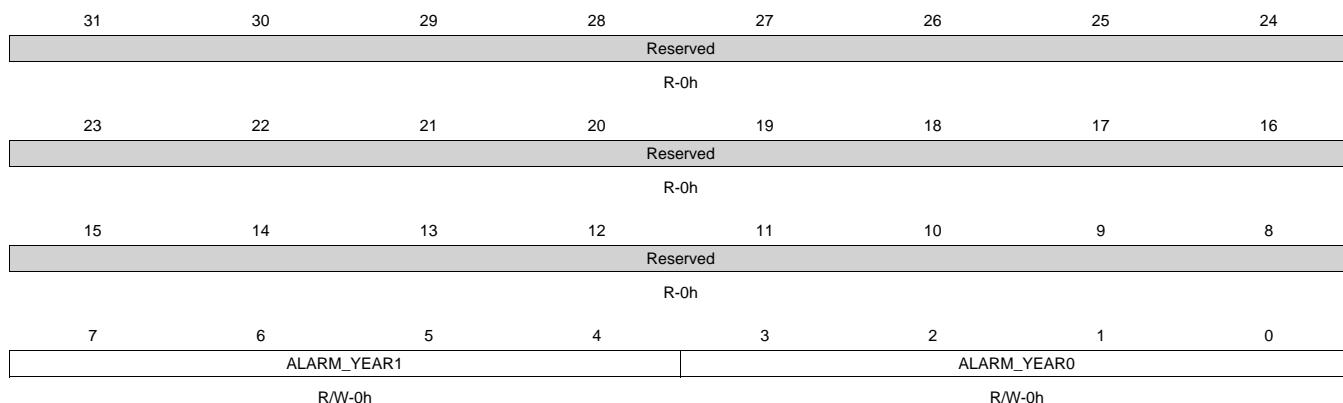
| Bit  | Field        | Type | Reset | Description                            |
|------|--------------|------|-------|--|
| 31-5 | Reserved     | R    | 0h    |  |
| 4    | ALARM_MONTH1 | R/W  | 0h    | 2nd digit of months, Range from 0 to 1 |
| 3-0  | ALARM_MONTH0 | R/W  | 1h    | 1st digit of months, Range from 0 to 9 |

### 20.3.5.13 ALARM\_YEARS\_REG Register (offset = 34h) [reset = 0h]

ALARM\_YEARS\_REG is shown in [Figure 20-69](#) and described in [Table 20-75](#).

The ALARM\_YEARS\_REG is used to program the year for the alarm interrupt. Only the last two digits are used to represent the year and is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-69. ALARM\_YEARS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-75. ALARM\_YEARS\_REG Register Field Descriptions**

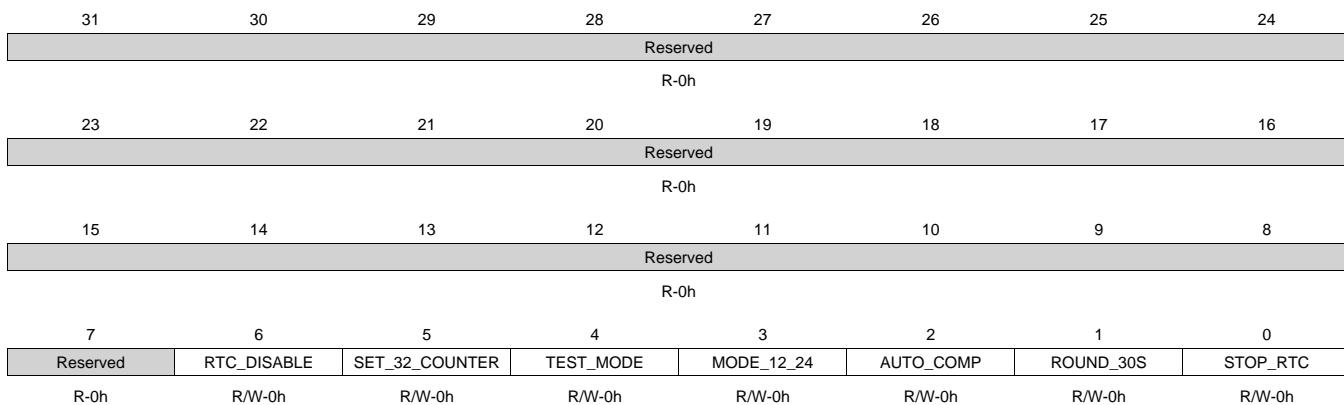
| Bit  | Field       | Type | Reset | Description                           |
|------|-------------|------|-------|---------------------------------------|
| 31-8 | Reserved    | R    | 0h    |                                       |
| 7-4  | ALARM_YEAR1 | R/W  | 0h    | 2nd digit of years, Range from 0 to 9 |
| 3-0  | ALARM_YEAR0 | R/W  | 0h    | 1st digit of years, Range from 0 to 9 |

#### 20.3.5.14 RTC\_CTRL\_REG Register (offset = 40h) [reset = 0h]

RTC\_CTRL\_REG is shown in [Figure 20-70](#) and described in [Table 20-76](#).

The RTC\_CTRL\_REG contains the controls to enable/disable the RTC, set the 12/24 hour time mode, to enable the 30 second rounding feature, and to STOP/START the RTC. The SET\_32\_COUNTER bit must only be used when the RTC is frozen. The RTC\_DISABLE bit must only be used to completely disable the RTC function. When this bit is set, the 32 kHz clock is gated and the RTC is frozen. From this point, resetting this bit to zero can lead to unexpected behavior. This bit should only be used if the RTC function is unwanted in the application, in order to save power. MODE\_12\_24: It is possible to switch between the two modes at any time without disturbing the RTC. Read or write is always performed with the current mode. Auto compensation is enabled by the AUTO\_COMP bit. If the COMP\_REG value is positive, compensation occurs after the second change event. COMP\_REG cycles are removed from the next second. If the COMP\_REG value is negative, compensation occurs before the second change event. COMP\_REG cycles are added to the current second. This enables it to compensate with one 32-kHz period accuracy each hour. The ROUND\_30S bit is a toggle bit; the ARM can only write 1 and the RTC clears it. If the ARM sets the ROUND\_30S bit and then reads it, the ARM reads 1 until the round-to-the-closest-minute is performed at the next second. The ARM can stop the RTC by clearing the STOP\_RTC bit (owing to internal resynchronization, the RUN bit of the status register (STATUS\_REG) must be checked to ensure that the RTC is frozen), then update TC values, and re-start the RTC by resetting the STOP\_RTC bit.

**Figure 20-70. RTC\_CTRL\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-76. RTC\_CTRL\_REG Register Field Descriptions**

| Bit  | Field          | Type | Reset | Description   |
|------|----------------|------|-------|---|
| 31-7 | Reserved       | R    | 0h    |   |
| 6    | RTC_DISABLE    | R/W  | 0h    | Disable RTC module and gate 32-kHz reference clock.<br>0x0 = RTC enable<br>0x1 = RTC disable (no 32 kHz clock)  |
| 5    | SET_32_COUNTER | R/W  | 0h    | Set the 32-kHz counter with the value stored in the compensation registers when the SET_32_COUNTER bit is set.<br>0x0 = No action.<br>0x1 = Set the 32Khz counter with compensation registers value |
| 4    | TEST_MODE      | R/W  | 0h    | Test mode.<br>0x0 = Functional mode<br>0x1 = Test mode (Auto compensation is enabled when the 32Khz counter reaches its end)  |

**Table 20-76. RTC\_CTRL\_REG Register Field Descriptions (continued)**

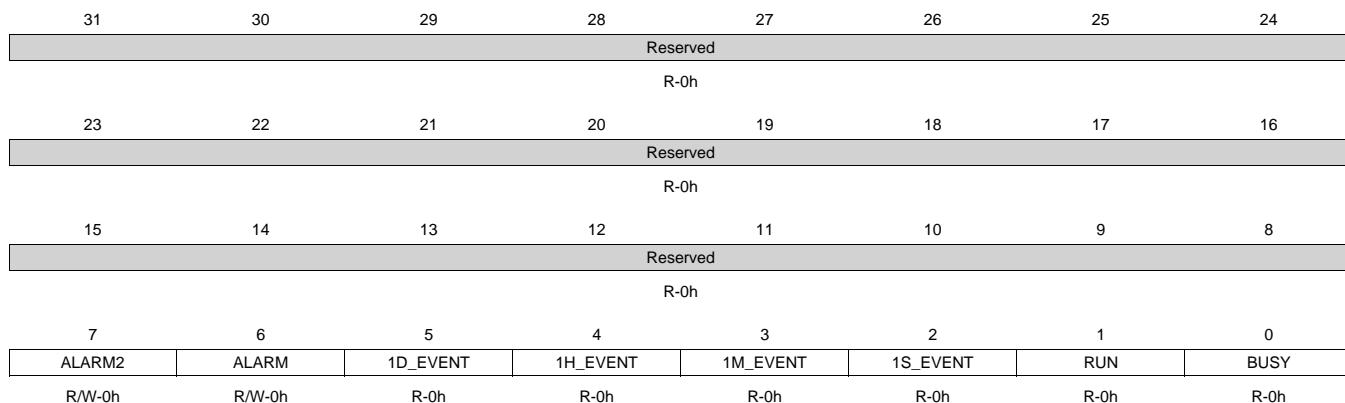
| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 3   | MODE_12_24 | R/W  | 0h    | Enable 12-hour mode for HOURS and ALARMHOURS registers.<br>0x0 = 24-hr mode<br>0x1 = 12-hour mode  |
| 2   | AUTO_COMP  | R/W  | 0h    | Enable oscillator compensation mode.<br>0x0 = No auto compensation<br>0x1 = Auto compensation enabled                                    |
| 1   | ROUND_30S  | R/W  | 0h    | Enable one-time rounding to nearest minute on next time register read.<br>0x0 = No update<br>0x1 = Time is rounded to the nearest minute |
| 0   | STOP_RTC   | R/W  | 0h    | Stop the RTC 32-kHz counter.<br>0x0 = RTC is frozen<br>0x1 = RTC is running  |

### 20.3.5.15 RTC\_STATUS\_REG Register (offset = 44h) [reset = 0h]

RTC\_STATUS\_REG is shown in [Figure 20-71](#) and described in [Table 20-77](#).

The RTC\_STATUS\_REG contains bits that signal the status of interrupts, events to the processor. Status for the alarm interrupt and timer events are notified by the register. The alarm interrupt keeps its low level until the ARM writes 1 in the ALARM bit of the RTC\_STATUS\_REG register. ALARM2: This bit will indicate the status of the alarm interrupt. Writing a 1 to the bit clears the interrupt. ALARM: This bit will indicate the status of the alarm interrupt. Writing a 1 to the bit clears the interrupt. 1D\_EVENT1: This bit will indicate if a day event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller. 1H\_EVENT1: This bit will indicate if an hour event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller. 1M\_EVENT1: This bit will indicate if a minute event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller. 1S\_EVENT1: This bit will indicate if a second event has occurred. An interrupt will be generated to the processor based on the masking of the interrupt controller. RUN: This bit will indicate if RTC is frozen or it is running. The RUN bit shows the real state of the RTC. Indeed, because the STOP\_RTC signal is resynchronized on 32-kHz clock the action of this bit is delayed. BUSY: This bit will give the status of RTC module. The Time and alarm registers can be modified only when this bit is 0. The timer interrupt is a negative edge sensitive low-level pulse (1 OCP cycle duration).

**Figure 20-71. RTC\_STATUS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-77. RTC\_STATUS\_REG Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7    | ALARM2   | R/W  | 0h    | Indicates that an alarm2 interrupt has been generated. Software needs to wait 31 s before it clears this status to allow pmic_pwr_enable 1 '0 transmission. |
| 6    | ALARM    | R/W  | 0h    | Indicates that an alarm interrupt has been generated  |
| 5    | 1D_EVENT | R    | 0h    | One day has occurred  |
| 4    | 1H_EVENT | R    | 0h    | One hour has occurred   |
| 3    | 1M_EVENT | R    | 0h    | One minute has occurred   |
| 2    | 1S_EVENT | R    | 0h    | One second has occurred   |
| 1    | RUN      | R    | 0h    | RTC is frozen or is running.<br>0x0 = RTC is frozen<br>0x1 = RTC is running   |
| 0    | BUSY     | R    | 0h    | Status of RTC module.<br>0x0 = Updating event in more than 15 s<br>0x1 = Updating event   |

### 20.3.5.16 RTC\_INTERRUPTS\_REG Register (offset = 48h) [reset = 0h]

RTC\_INTERRUPTS\_REG is shown in Figure 20-72 and described in Table 20-78.

The RTC\_INTERRUPTS\_REG is used to enable or disable the RTC from generating interrupts. The timer interrupt and alarm interrupt can be controlled using this register. The ARM must respect the BUSY period to prevent spurious interrupt. To set a period timer interrupt, the respective period value must be set in the EVERY field. For example, to set a periodic timer interrupt for every hour, the EVERY field has to be set to 2. Along with this the IT\_TIMER bit also has to be set for the periodic interrupt to be generated. IT\_ALARM bit has to be set to generate an alarm interrupt.

**Figure 20-72. RTC\_INTERRUPTS\_REG Register**

|          |    |           |    |          |    |          |    |
|----------|----|-----------|----|----------|----|----------|----|
| 31       | 30 | 29        | 28 | 27       | 26 | 25       | 24 |
| Reserved |    |           |    |          |    |          |    |
| R-0h     |    |           |    |          |    |          |    |
| 23       | 22 | 21        | 20 | 19       | 18 | 17       | 16 |
| Reserved |    |           |    |          |    |          |    |
| R-0h     |    |           |    |          |    |          |    |
| 15       | 14 | 13        | 12 | 11       | 10 | 9        | 8  |
| Reserved |    |           |    |          |    |          |    |
| R-0h     |    |           |    |          |    |          |    |
| 7        | 6  | 5         | 4  | 3        | 2  | 1        | 0  |
| Reserved |    | IT_ALARM2 |    | IT_ALARM |    | IT_TIMER |    |
| R-0h     |    | R/W-0h    |    | R/W-0h   |    | R/W-0h   |    |
| R/W-0h   |    |           |    |          |    |          |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-78. RTC\_INTERRUPTS\_REG Register Field Descriptions**

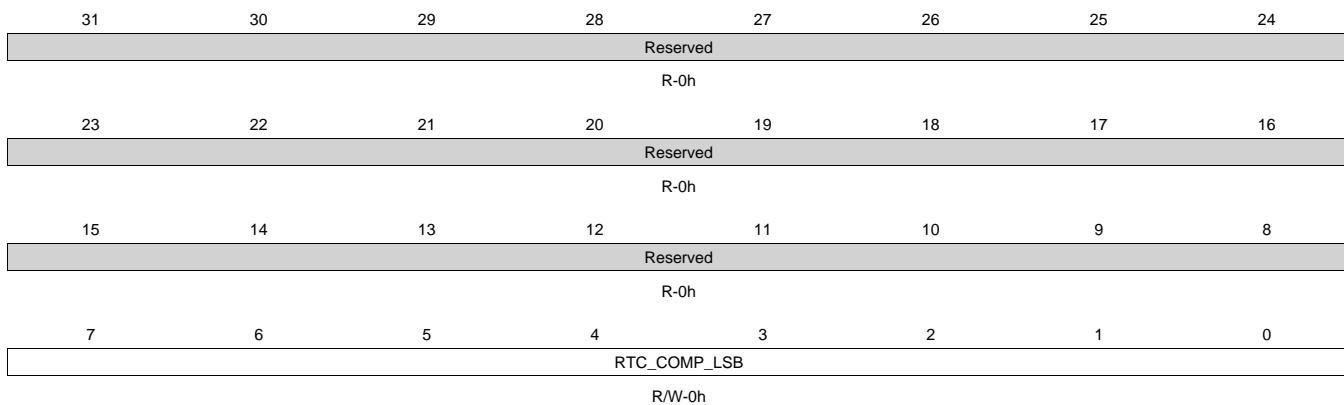
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-5 | Reserved  | R    | 0h    |  |
| 4    | IT_ALARM2 | R/W  | 0h    | Enable one interrupt when the alarm value is reached (TC ALARM2 registers) by the TC registers       |
| 3    | IT_ALARM  | R/W  | 0h    | Enable one interrupt when the alarm value is reached (TC ALARM registers) by the TC registers        |
| 2    | IT_TIMER  | R/W  | 0h    | Enable periodic interrupt.<br>0x0 = Interrupt disabled<br>0x1 = Interrupt enabled                    |
| 1-0  | EVERY     | R/W  | 0h    | Interrupt period.<br>0x0 = Every second<br>0x1 = Every minute<br>0x2 = Every hour<br>0x3 = Every day |

### 20.3.5.17 RTC\_COMP\_LSB\_REG Register (offset = 4Ch) [reset = 0h]

RTC\_COMP\_LSB\_REG is shown in [Figure 20-73](#) and described in [Table 20-79](#).

The COMP\_LSB\_REG is used to program the LSB value of the 32 kHz periods to be added to the 32 kHz counter every hour. This is used to compensate the oscillator drift. The COMP\_LSB\_REG works together with the compensation (MSB) register (COMP\_MSB\_REG). The AUTOCOMP bit in the control register (CTRL\_REG) must be enabled for compensation to take place. This register must be written in two's complement. That means that to add one 32-kHz oscillator period every hour, the ARM must write FFFFh into RTC\_COMP\_MSB\_REG and RTC\_COMP\_LSB\_REG. To remove one 32-kHz oscillator period every hour, the ARM must write 0001h into RTC\_COMP\_MSB\_REG and RTC\_COMP\_LSB\_REG. The 7FFFh value is forbidden.

**Figure 20-73. RTC\_COMP\_LSB\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-79. RTC\_COMP\_LSB\_REG Register Field Descriptions**

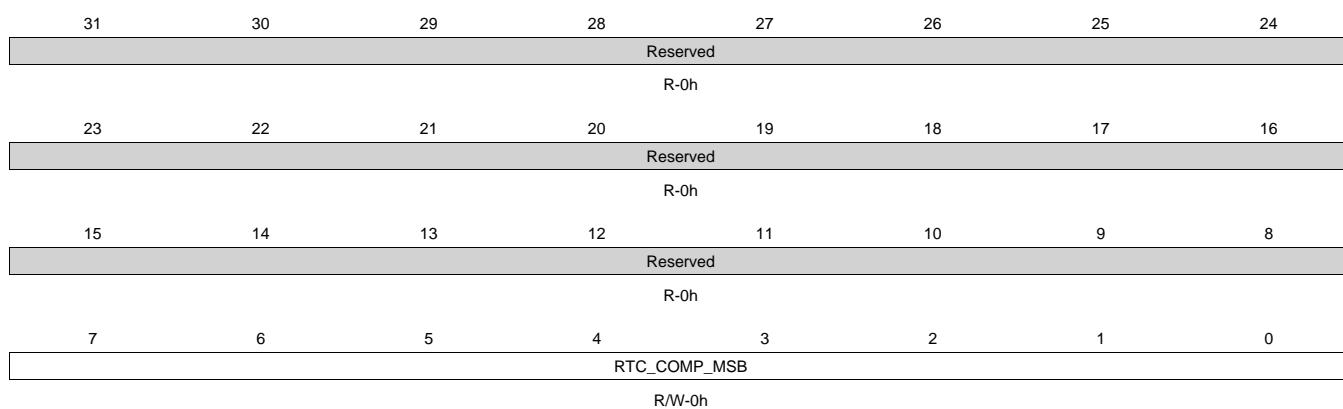
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-8 | Reserved     | R    | 0h    |   |
| 7-0  | RTC_COMP_LSB | R/W  | 0h    | Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour |

### 20.3.5.18 RTC\_COMP\_MSB\_REG Register (offset = 50h) [reset = 0h]

RTC\_COMP\_MSB\_REG is shown in [Figure 20-74](#) and described in [Table 20-80](#).

The COMP\_MSB\_REG is used to program the MSB value of the 32 kHz periods to be added to the 32 kHz counter every hour. This is used to compensate the oscillator drift. The COMP\_MSB\_REG works together with the compensation (LSB) register (COMP\_LSB\_REG) to set the hourly oscillator compensation value. The AUTOCOMP bit in the control register (CTRL\_REG) must be enabled for compensation to take place. This register must be written in two's complement. That means that to add one 32-kHz oscillator period every hour, the ARM must write FFFFh into RTC\_COMP\_MSB\_REG and RTC\_COMP\_LSB\_REG. To remove one 32-kHz oscillator period every hour, the ARM must write 0001h into RTC\_COMP\_MSB\_REG and RTC\_COMP\_LSB\_REG. The 7FFFh value is forbidden.

**Figure 20-74. RTC\_COMP\_MSB\_REG Register**



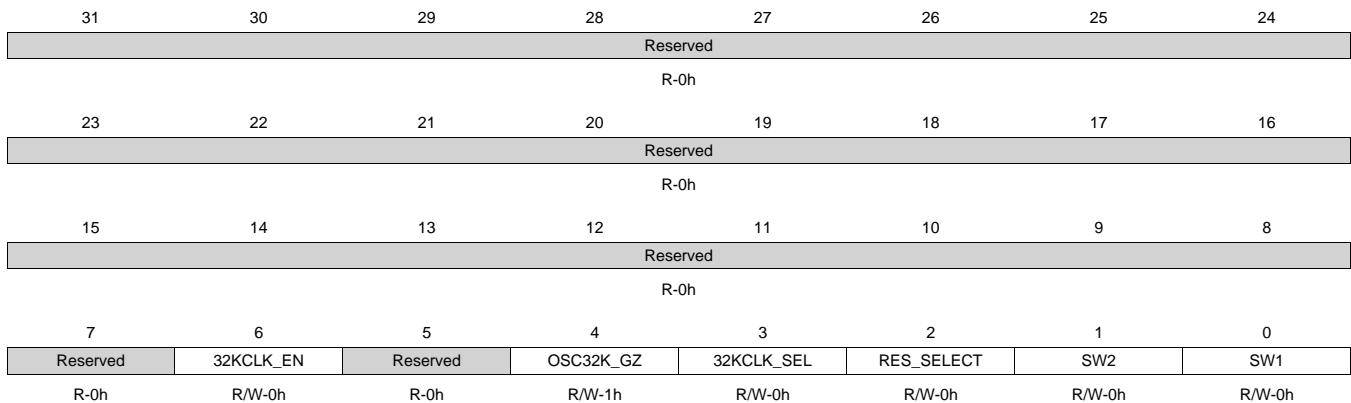
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-80. RTC\_COMP\_MSB\_REG Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-8 | Reserved     | R    | 0h    |   |
| 7-0  | RTC_COMP_MSB | R/W  | 0h    | Indicates number of 32-kHz periods to be added into the 32-kHz counter every hour |

**20.3.5.19 RTC\_OSC\_REG Register (offset = 54h) [reset = 10h]**

 RTC\_OSC\_REG is shown in [Figure 20-75](#) and described in [Table 20-81](#).

**Figure 20-75. RTC\_OSC\_REG Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-81. RTC\_OSC\_REG Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-7 | Reserved   | R    | 0h    |   |
| 6    | 32KCLK_EN  | R/W  | 0h    | 32-kHz clock enable post clock mux of rtc_32k_clk_RTC_32k_aux_clk and rtc_32k_clk_RTC_32k_clk   |
| 5    | Reserved   | R    | 0h    |   |
| 4    | OSC32K_GZ  | R/W  | 1h    | Disable the oscillator and apply high impedance to the output<br>0x0 = Enable<br>0x1 = Disabled and high impedance  |
| 3    | 32KCLK_SEL | R/W  | 0h    | 32-kHz clock source select<br>0x0 = Selects internal clock source, namely rtc_32k_clk_RTC_32k_aux_clk<br>0x1 = Selects external clock source, namely rtc_32k_clk_RTC_32k_clk that is from the 32-kHz oscillator |
| 2    | RES_SELECT | R/W  | 0h    | External feedback resistor<br>0x0 = Internal<br>0x1 = External  |
| 1    | SW2        | R/W  | 0h    | Inverter size adjustment  |
| 0    | SW1        | R/W  | 0h    | Inverter size adjustment  |

### 20.3.5.20 RTC\_SCRATCH0\_REG Register (offset = 60h) [reset = 0h]

RTC\_SCRATCH0\_REG is shown in [Figure 20-76](#) and described in [Table 20-82](#).

The RTC\_SCRATCH0\_REG is used to hold some required values for the RTC register.

**Figure 20-76. RTC\_SCRATCH0\_REG Register**

| 31          | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| RTCSCRATCH0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-82. RTC\_SCRATCH0\_REG Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description                             |
|------|-------------|------|-------|---|
| 31-0 | RTCSCRATCH0 | R/W  | 0h    | Scratch registers, available to program |

### 20.3.5.21 RTC\_SCRATCH1\_REG Register (offset = 64h) [reset = 0h]

RTC\_SCRATCH1\_REG is shown in [Figure 20-77](#) and described in [Table 20-83](#).

The RTC\_SCRATCH1\_REG is used to hold some required values for the RTC register.

**Figure 20-77. RTC\_SCRATCH1\_REG Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| RTCSRATCH1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-83. RTC\_SCRATCH1\_REG Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                             |
|------|------------|------|-------|---|
| 31-0 | RTCSRATCH1 | R/W  | 0h    | Scratch registers, available to program |

### 20.3.5.22 RTC\_SCRATCH2\_REG Register (offset = 68h) [reset = 0h]

RTC\_SCRATCH2\_REG is shown in [Figure 20-78](#) and described in [Table 20-84](#).

The RTC\_SCRATCH2\_REG is used to hold some required values for the RTC register.

**Figure 20-78. RTC\_SCRATCH2\_REG Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| RTCSRATCH2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-84. RTC\_SCRATCH2\_REG Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                             |
|------|------------|------|-------|---|
| 31-0 | RTCSRATCH2 | R/W  | 0h    | Scratch registers, available to program |

### 20.3.5.23 KICK0R Register (offset = 6Ch) [reset = 0h]

KICK0R is shown in [Figure 20-79](#) and described in [Table 20-85](#).

The kick registers (KICKnR) are used to enable and disable write protection on the RTC registers. Out of reset, the RTC registers are write-protected. To disable write protection, correct keys must be written to the KICKnR registers. The Kick0 register allows writing to unlock the kick0 data. To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R. RTC register write protection is enabled when any value is written to KICK0R.

**Figure 20-79. KICK0R Register**

|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31     | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KICK0_ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-85. KICK0R Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description |
|------|--------|------|-------|-------------|
| 31-0 | KICK0_ | W    | 0h    | Kick0 data  |

### 20.3.5.24 KICK1R Register (offset = 70h) [reset = 0h]

KICK1R is shown in [Figure 20-80](#) and described in [Table 20-86](#).

The kick registers (KICKnR) are used to enable and disable write protection on the RTC registers. Out of reset, the RTC registers are write-protected. To disable write protection, correct keys must be written to the KICKnR registers. The Kick1 register allows writing to unlock the kick1 data and the kicker mechanism to write to other MMRs. To disable RTC register write protection, the value of 83E7 0B13h must be written to KICK0R, followed by the value of 95A4 F1E0h written to KICK1R.

**Figure 20-80. KICK1R Register**

|       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KICK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| W-0h  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

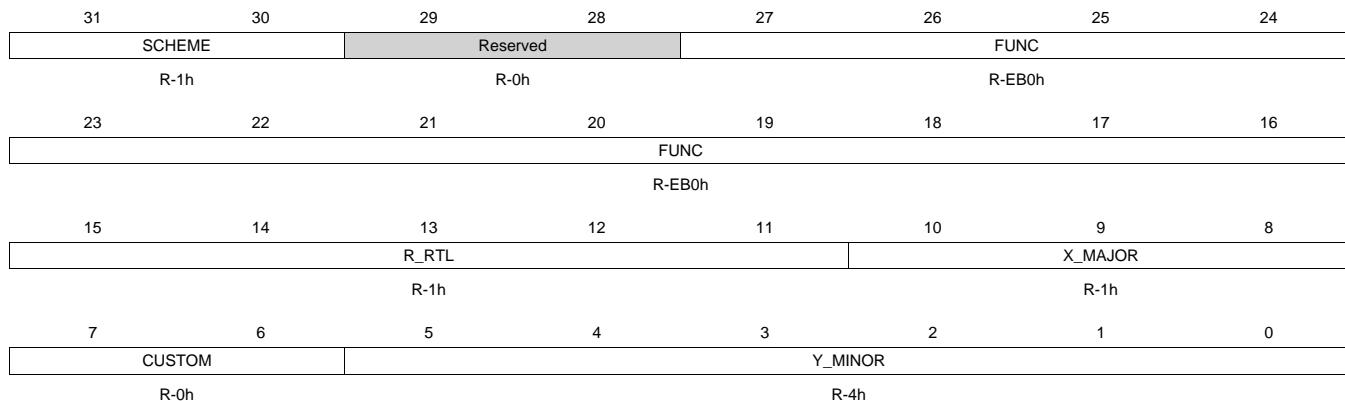
**Table 20-86. KICK1R Register Field Descriptions**

| Bit  | Field | Type | Reset | Description |
|------|-------|------|-------|-------------|
| 31-0 | KICK1 | W    | 0h    | Kick1 data  |

### 20.3.5.25 RTC\_REVISION Register (offset = 74h) [reset = 4EB00904h]

RTC\_REVISION is shown in [Figure 20-81](#) and described in [Table 20-87](#).

**Figure 20-81. RTC\_REVISION Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

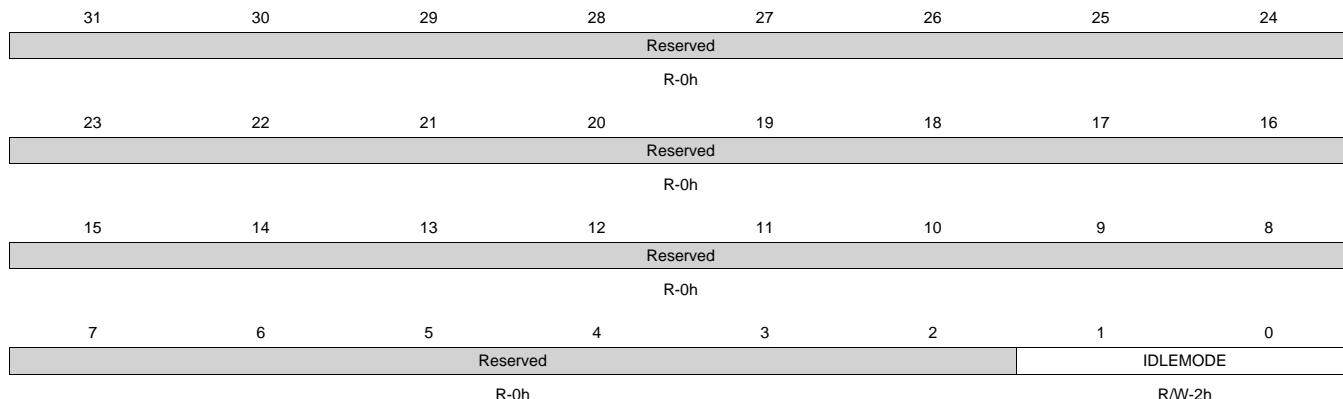
**Table 20-87. RTC\_REVISION Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-30 | SCHEME   | R    | 1h    | Used to distinguish between old scheme and current     |
| 29-28 | Reserved | R    | 0h    |  |
| 27-16 | FUNC     | R    | EB0h  | Function indicates a software compatible module family |
| 15-11 | R RTL    | R    | 1h    | RTL Version (R)  |
| 10-8  | X_MAJOR  | R    | 1h    | Major Revision   |
| 7-6   | CUSTOM   | R    | 0h    | Indicates a special version for a particular device    |
| 5-0   | Y_MINOR  | R    | 4h    | Minor Revision (Y)                                     |

### 20.3.5.26 RTC\_SYSConfig Register (offset = 78h) [reset = 2h]

RTC\_SYSConfig is shown in [Figure 20-82](#) and described in [Table 20-88](#).

**Figure 20-82. RTC\_SYSConfig Register**



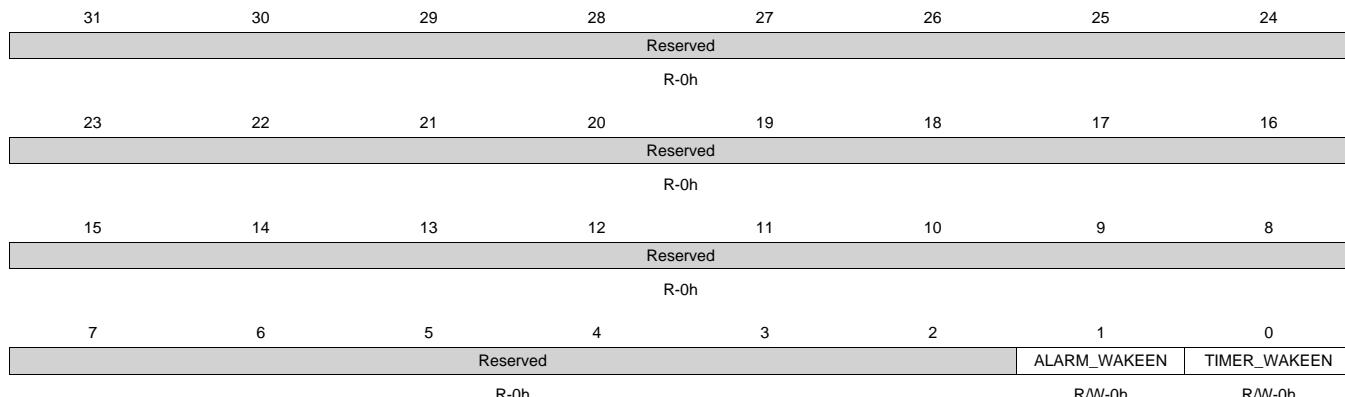
LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-88. RTC\_SYSConfig Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-2 | Reserved | R    | 0h    |   |
| 1-0  | IDLEMODE | R/W  | 2h    | <p>Configuration of the local target state management mode, By definition target can handle read/write transaction as long as it is out of IDLE state.</p> <p>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e., regardless of the IP module's internal requirements; Backup mode, for debug only.</p> <p>0x1 = No-idle mode: local target never enters idle state, Backup mode, for debug only.</p> <p>0x2 = Smart-idle mode: local target's state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements, IP module shall not generate (IRQ- or DMA-request-related) wakeup events.</p> <p>0x3 = Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges the system's idle requests, depending on the IP module's internal requirements, IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state.</p> |

**20.3.5.27 RTC\_IRQWAKEEN Register (offset = 7Ch) [reset = 0h]**

 RTC\_IRQWAKEEN is shown in [Figure 20-83](#) and described in [Table 20-89](#).

**Figure 20-83. RTC\_IRQWAKEEN Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-89. RTC\_IRQWAKEEN Register Field Descriptions**

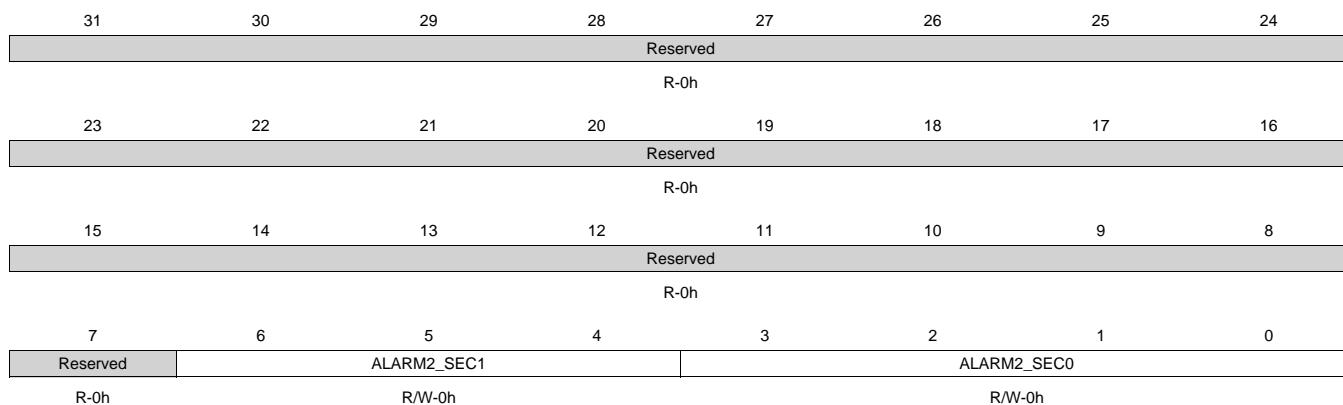
| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-2 | Reserved     | R    | 0h    |   |
| 1    | ALARM_WAKEEN | R/W  | 0h    | Wakeup generation for event Alarm.<br>0x0 = Wakeup disabled<br>0x1 = Wakeup enabled |
| 0    | TIMER_WAKEEN | R/W  | 0h    | Wakeup generation for event Timer.<br>0x0 = Wakeup disabled<br>0x1 = Wakeup enabled |

### 20.3.5.28 ALARM2\_SECONDS\_REG Register (offset = 80h) [reset = 0h]

ALARM2\_SECONDS\_REG is shown in [Figure 20-84](#) and described in [Table 20-90](#).

The ALARM2\_SECONDS\_REG is used to program the second value for the alarm2 time. Seconds are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-84. ALARM2\_SECONDS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-90. ALARM2\_SECONDS\_REG Register Field Descriptions**

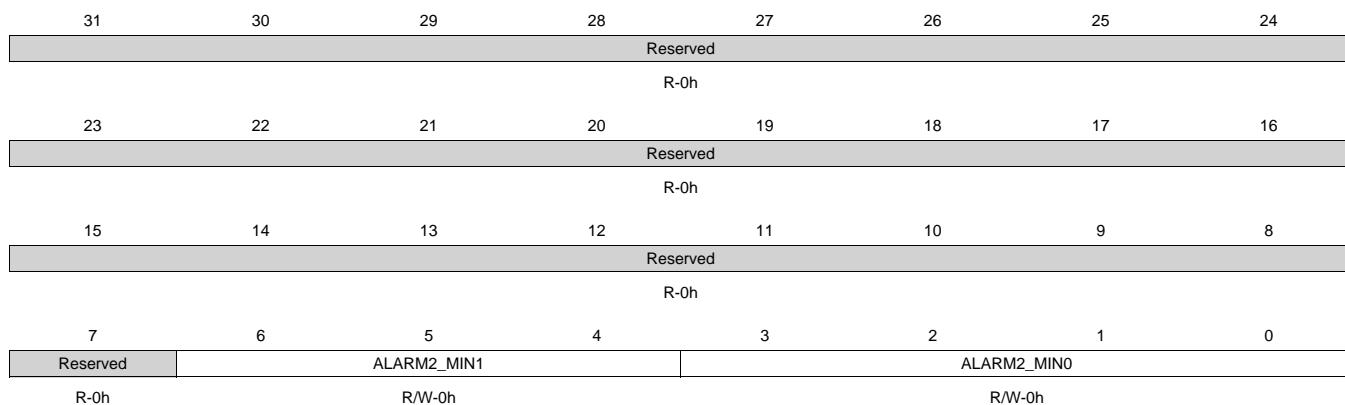
| Bit  | Field       | Type | Reset | Description                           |
|------|-------------|------|-------|---------------------------------------|
| 31-7 | Reserved    | R    | 0h    |                                       |
| 6-4  | ALARM2_SEC1 | R/W  | 0h    | 2nd digit of seconds, Range is 0 to 5 |
| 3-0  | ALARM2_SEC0 | R/W  | 0h    | 1st digit of seconds, Range is 0 to 9 |

### 20.3.5.29 ALARM2\_MINUTES\_REG Register (offset = 84h) [reset = 0h]

ALARM2\_MINUTES\_REG is shown in [Figure 20-85](#) and described in [Table 20-91](#).

The ALARM2\_MINUTES\_REG is used to program the minute value for the alarm2 time. Minutes are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-85. ALARM2\_MINUTES\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-91. ALARM2\_MINUTES\_REG Register Field Descriptions**

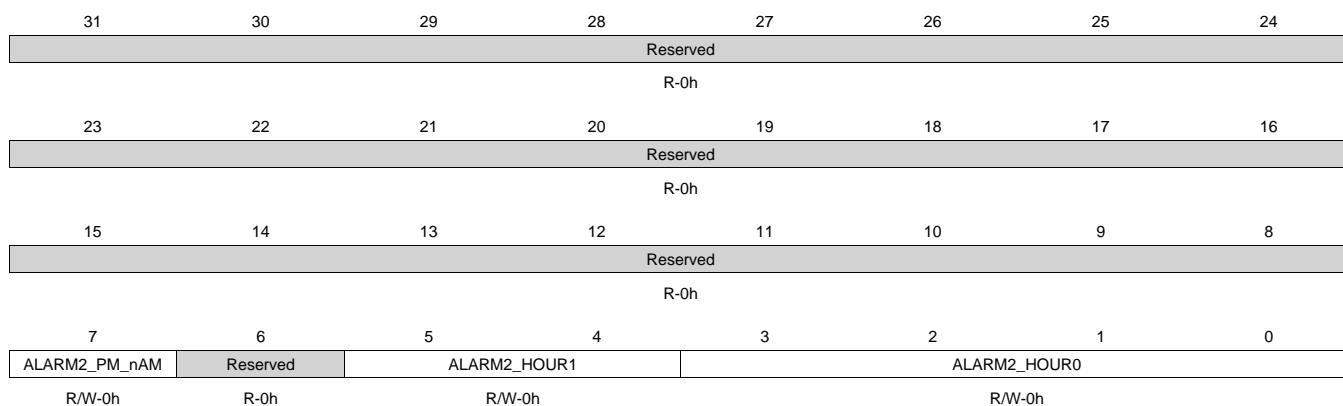
| Bit  | Field       | Type | Reset | Description                           |
|------|-------------|------|-------|---------------------------------------|
| 31-7 | Reserved    | R    | 0h    |                                       |
| 6-4  | ALARM2_MIN1 | R/W  | 0h    | 2nd digit of minutes, Range is 0 to 5 |
| 3-0  | ALARM2_MIN0 | R/W  | 0h    | 1st digit of minutes, Range is 0 to 9 |

### 20.3.5.30 ALARM2\_HOURS\_REG Register (offset = 88h) [reset = 0h]

ALARM2\_HOURS\_REG is shown in [Figure 20-86](#) and described in [Table 20-92](#).

The ALARM2\_HOURS\_REG is used to program the hour value for the alarm2 time. Hours are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-86. ALARM2\_HOURS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-92. ALARM2\_HOURS\_REG Register Field Descriptions**

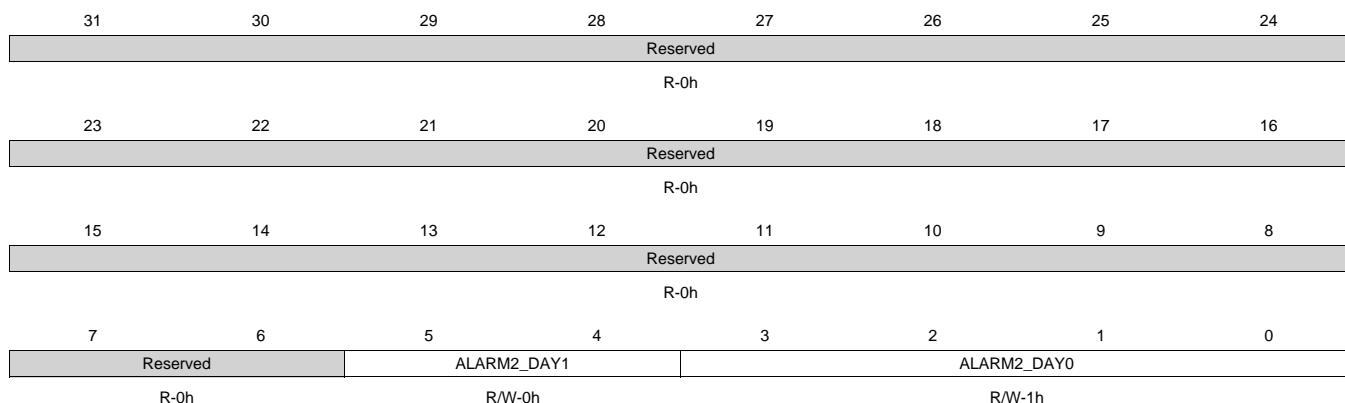
| Bit  | Field         | Type | Reset | Description   |
|------|---------------|------|-------|---|
| 31-8 | Reserved      | R    | 0h    |   |
| 7    | ALARM2_PM_nAM | R/W  | 0h    | Only used in PM_AM mode (otherwise 0)<br>0x0 = AM<br>0x1 = PM |
| 6    | Reserved      | R    | 0h    |   |
| 5-4  | ALARM2_HOUR1  | R/W  | 0h    | 2nd digit of hours, Range is 0 to 2                           |
| 3-0  | ALARM2_HOUR0  | R/W  | 0h    | 1st digit of hours, Range is 0 to 9                           |

### 20.3.5.31 ALARM2\_DAYS\_REG Register (offset = 8Ch) [reset = 1h]

ALARM2\_DAYS\_REG is shown in [Figure 20-87](#) and described in [Table 20-93](#).

The ALARM2\_DAYS\_REG is used to program the day of the month value for the alarm2 date. Days are stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-87. ALARM2\_DAYS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-93. ALARM2\_DAYS\_REG Register Field Descriptions**

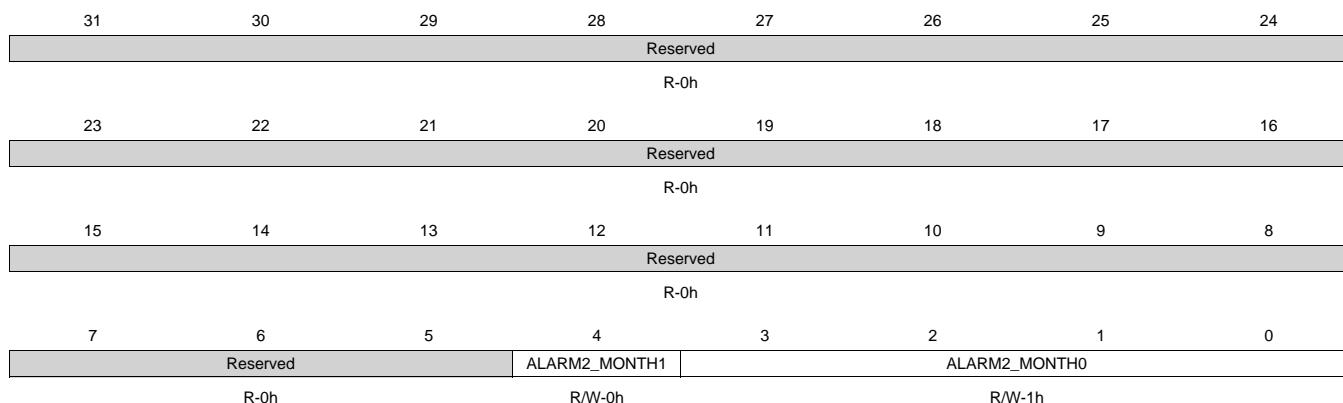
| Bit  | Field       | Type | Reset | Description                           |
|------|-------------|------|-------|---------------------------------------|
| 31-6 | Reserved    | R    | 0h    |                                       |
| 5-4  | ALARM2_DAY1 | R/W  | 0h    | 2nd digit for days, Range from 0 to 3 |
| 3-0  | ALARM2_DAY0 | R/W  | 1h    | 1st digit for days, Range from 0 to 9 |

### 20.3.5.32 ALARM2\_MONTHS\_REG Register (offset = 90h) [reset = 1h]

ALARM2\_MONTHS\_REG is shown in [Figure 20-88](#) and described in [Table 20-94](#).

The ALARM2\_MONTHS\_REG is used to program the month in the year value for the alarm2 date. The month is stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-88. ALARM2\_MONTHS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-94. ALARM2\_MONTHS\_REG Register Field Descriptions**

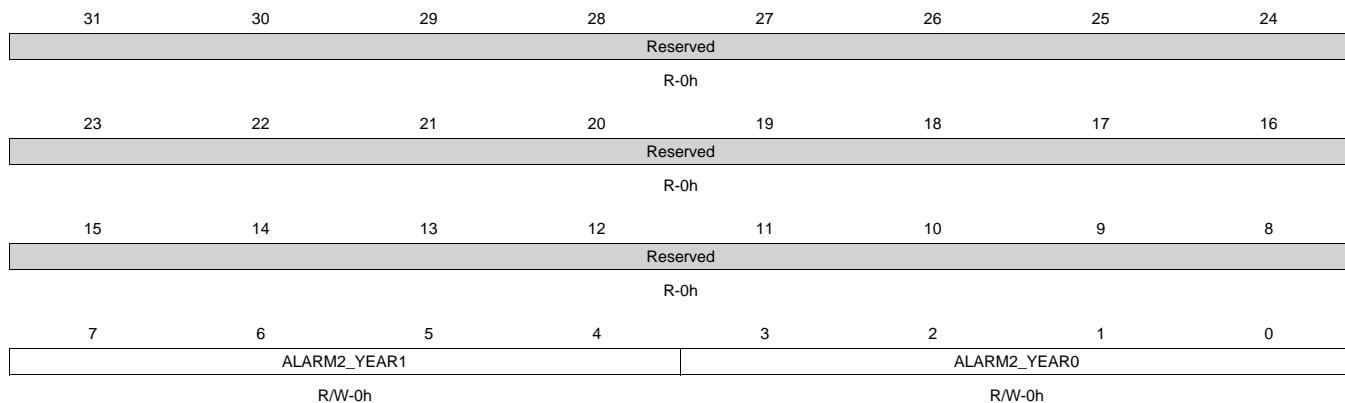
| Bit  | Field         | Type | Reset | Description                            |
|------|---------------|------|-------|--|
| 31-5 | Reserved      | R    | 0h    |  |
| 4    | ALARM2_MONTH1 | R/W  | 0h    | 2nd digit of months, Range from 0 to 1 |
| 3-0  | ALARM2_MONTH0 | R/W  | 1h    | 1st digit of months, Range from 0 to 9 |

### 20.3.5.33 ALARM2\_YEARS\_REG Register (offset = 94h) [reset = 0h]

ALARM2\_YEARS\_REG is shown in [Figure 20-89](#) and described in [Table 20-95](#).

The ALARM2\_YEARS\_REG is used to program the year for the alarm2 date. Only the last two digits are used to represent the year and stored as BCD format. In BCD format, the decimal numbers 0 through 9 are encoded with their binary equivalent.

**Figure 20-89. ALARM2\_YEARS\_REG Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-95. ALARM2\_YEARS\_REG Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description                           |
|------|--------------|------|-------|---------------------------------------|
| 31-8 | Reserved     | R    | 0h    |                                       |
| 7-4  | ALARM2_YEAR1 | R/W  | 0h    | 2nd digit of years, Range from 0 to 9 |
| 3-0  | ALARM2_YEAR0 | R/W  | 0h    | 1st digit of years, Range from 0 to 9 |

### 20.3.5.34 RTC\_PMIC Register (offset = 98h) [reset = 0h]

RTC\_PMIC is shown in [Figure 20-90](#) and described in [Table 20-96](#).

**Figure 20-90. RTC\_PMIC Register**

|                   |    |    |    |                  |    |               |    |
|-------------------|----|----|----|------------------|----|---------------|----|
| 31                | 30 | 29 | 28 | 27               | 26 | 25            | 24 |
| Reserved          |    |    |    |                  |    |               |    |
| R-0h              |    |    |    |                  |    |               |    |
| 23                | 22 | 21 | 20 | 19               | 18 | 17            | 16 |
| Reserved          |    |    |    | PWR_ENABLE_SM    |    | PWR_ENABLE_EN |    |
| R-0h              |    |    |    |                  |    |               |    |
| 15                | 14 | 13 | 12 | 11               | 10 | 9             | 8  |
| EXT_WAKEUP_STATUS |    |    |    | EXT_WAKEUP_DB_EN |    |               |    |
| R/W-0h            |    |    |    |                  |    |               |    |
| 7                 | 6  | 5  | 4  | 3                | 2  | 1             | 0  |
| EXT_WAKEUP_POL    |    |    |    | EXT_WAKEUP_EN    |    |               |    |
| R/W-0h            |    |    |    |                  |    |               |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-96. RTC\_PMIC Register Field Descriptions**

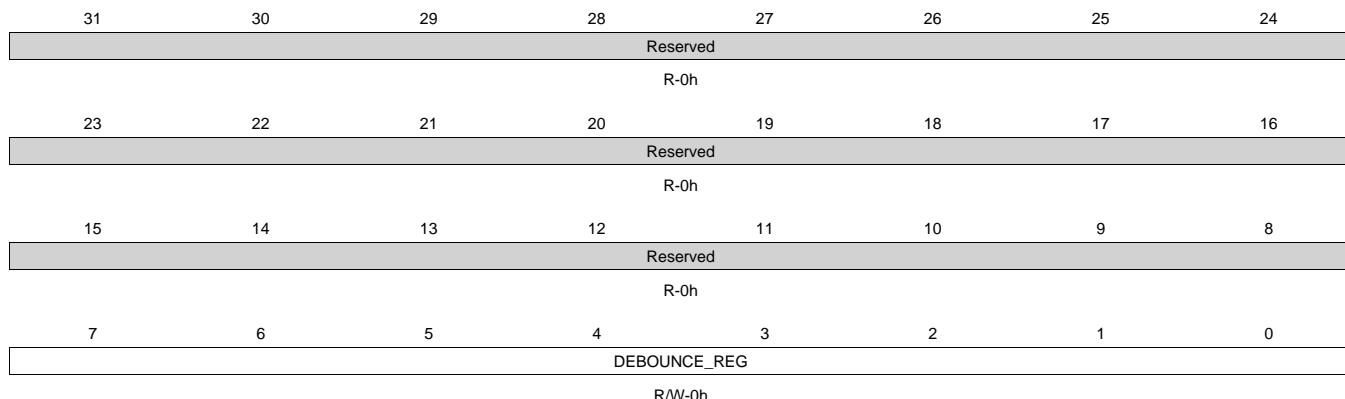
| Bit   | Field             | Type | Reset | Description  |
|-------|-------------------|------|-------|--|
| 31-19 | Reserved          | R    | 0h    |  |
| 18-17 | PWR_ENABLE_SM     | R    | 0h    | Power state machine state.<br>0x00 = Idle/Default<br>0x01 = Shutdown (ALARM2 and PWR_ENABLE_EN is set to 1).<br>Note: 31 us latency from ALARM2 event).<br>0x10 = Time-based wakeup (ALARM status is set).<br>0x11 = External-event-based wakeup (one or more bit set in EXT_WAKEUP_STATUS)  |
| 16    | PWR_ENABLE_EN     | R/W  | 0h    | Enable for PMIC_POWER_EN signal<br>0x0 = Disable. When Disabled, pmic_power_en signal will always be driven as 1, ON state.<br>0x1 = Enable. When Enabled: pmic_power_en signal will be controlled by ext_wakeup, alarm, and alarm2; ON -> OFF (Turn OFF) only by ALARM2 event; OFF -> ON (TURN ON) only by ALARM event OR ext_wakeup event. |
| 15-12 | EXT_WAKEUP_STATUS | R/W  | 0h    | External wakeup status.<br>Write 1 to clear EXT_WAKEUP_STATUS[n] status of ext_wakeup[n].<br>0x0 = External wakeup event has not occurred<br>0x1 = External wakeup event has occurred  |
| 11-8  | EXT_WAKEUP_DB_EN  | R/W  | 0h    | External wakeup debounce enabled.<br>EXT_WAKEUP_DB_EN[n] controls ext_wakeup[n]<br>0x0 = Disable<br>0x1 = Enable. When enabled, RTC_DEBOUNCE_REG defines the debounce time.  |
| 7-4   | EXT_WAKEUP_POL    | R/W  | 0h    | External wakeup inputs polarity.<br>EXT_WAKEUP_POL[n] controls ext_wakeup[n].<br>0x0 = Active high<br>0x1 = Active low   |
| 3-0   | EXT_WAKEUP_EN     | R/W  | 0h    | Enable external wakeup inputs.<br>EXT_WAKEUP_EN[n] controls ext_wakeup[n].<br>0x0 = Ext. wakeup disabled<br>0x1 = Ext. wakeup enabled  |

### 20.3.5.35 RTC\_DEBOUNCE Register (offset = 9Ch) [reset = 0h]

RTC\_DEBOUNCE is shown in [Figure 20-91](#) and described in [Table 20-97](#).

The debounce timer uses the 32768-Hz clock. It allows choosing the timing or the accuracy of debouncing. A register receives a bit from the reference pin. You will choose the timing if you use the debouncing like a timer, or you will choose the accuracy if you use the debouncing like a real debouncing. The debouncing will be finished when the reference pin will stay the same value (defined in DEBOUNCE\_REG) for a defined time.

**Figure 20-91. RTC\_DEBOUNCE Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-97. RTC\_DEBOUNCE Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-8 | Reserved     | R    | 0h    |   |
| 7-0  | DEBOUNCE_REG | R/W  | 0h    | Debounce time.<br>A value, n, other than 0 results in a debounce time of $30.52 \text{ s} * (n+1)$ .<br>0x0 = Debounce time is 30.52 s. |

## 20.4 WATCHDOG

### 20.4.1 Introduction

The watchdog timer is an upward counter capable of generating a pulse on the reset pin and an interrupt to the device system modules following an overflow condition. The watchdog timer serves resets to the PRCM module and serves watchdog interrupts to the host ARM. The reset of the PRCM module causes warm reset of the device.

The watchdog timer can be accessed, loaded, and cleared by registers through the L4 interface. The watchdog timer have the 32-kHz clock for their timer clock input.

The watchdog timer connects to a single target agent port on the L4 interconnect. The default state of the watchdog timer is enabled and not running.

#### 20.4.1.1 Features

The main features of the watchdog timer controllers are:

- L4 slave interface support:
  - 32-bit data bus width
  - 32-/16-bit access supported
  - 8-bit access not supported
  - 11-bit address bus width
  - Burst mode not supported
  - Write nonposted transaction mode only
- Free-running 32-bit upward counter
- Programmable divider clock source ( $2^n$  where  $n = 0-7$ )
- On-the-fly read/write register (while counting)
- Subset programming model of the GP timer
- The watchdog timers are reset either on power-on or after a warm reset before they start counting.
- Reset or interrupt actions when a timer overflow condition occurs
- The watchdog timer generates a reset or an interrupt in its hardware integration.

#### 20.4.1.2 Unsupported Features

There are no unsupported WD Timer features in this device.

## 20.4.2 Integration

### 20.4.2.1 Public WD Timer Connectivity Attributes

The general connectivity for the WD Timer module in this device is shown in Table 20-98.

**Table 20-98. Public WD Timer Module Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Wakeup Domain  |
| Clock Domain        | PD_WKUP_L4_WKUP_GCLK (OCP)<br>PD_WKUP_WDT1_GCLK (Func) |
| Reset Signals       | WKUP_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle / Slave Wakeup                              |
| Interrupt Requests  | 1 Interrupt to MPU Subsystem (WDT1INT) and WakeM3      |
| DMA Requests        | None   |
| Physical Address    | L4 Wakeup slave port                                   |

### 20.4.2.2 Public WD Timer Clock and Reset Management

The Watchdog Timer functional clock (pi\_sys\_clk input) is sourced from either the on-chip ~32678 Hz oscillator (CLK\_RC32K) or the PER PLL generated 32.768 KHz clock (CLK\_32KHZ) as selected using CLKSEL\_WDT1\_CLK[CLKSEL] in the PRCM.

**Table 20-99. Public WD Timer Clock Signals**

| Clock Signal                   | Max Freq | Reference / Source                                     | Comments                          |
|--------------------------------|----------|--|-----------------------------------|
| PI_OCP_CLK<br>Interface clock  | 100 MHz  | CORE_CLKOUTM4 / 2                                      | pd_wkup_l4_wkup_gclk<br>from PRCM |
| PI_SYS_CLK<br>Functional clock | 32768 Hz | CLK_RC32K or<br>CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375) | pd_wkup_wdt1_gclk<br>from PRCM    |

## 20.4.3 Functional Description

### 20.4.3.1 Power Management

There are two clock domains in the watchdog timers:

- Functional clock domain: WDTi\_FCLK is a 32 kHz watchdog timer functional clock. It is used to clock the watchdog timer internal logic.
- Interface clock domain: WDTi\_ICLK is a 125 MHz watchdog timer interface clock. It is used to synchronize the watchdog timer L4 port to the L4 interconnect. All accesses from the interconnect are synchronous to WDTi\_ICLK.

In this device, the clocks to the watchdog timers are always On. The clocks cannot be turned off, if the watchdog timers is not being used.

### 20.4.3.2 Interrupts

[Table 20-100](#) list the event flags, and their masks, that cause module interrupts.

**Table 20-100. Watchdog Timer Events**

| Event Flag                | Event Mask                                   | Mapping | Comments                     |
|---------------------------|--|---------|------------------------------|
| WDT_WIRQSTAT[0] EVENT_OVF | WDT_WIRQENSET/WDT_WIRQENCLR[0]<br>OVF_IT_ENA | WDTINT  | Watchdog timer overflow      |
| WDT_WIRQSTAT[1] EVENT_DLY | WDT_WIRQENSET/WDT_WIRQENCLR[1]<br>DLY_IT_ENA | WDTINT  | Watchdog delay value reached |

### 20.4.3.3 General Watchdog Timer Operation

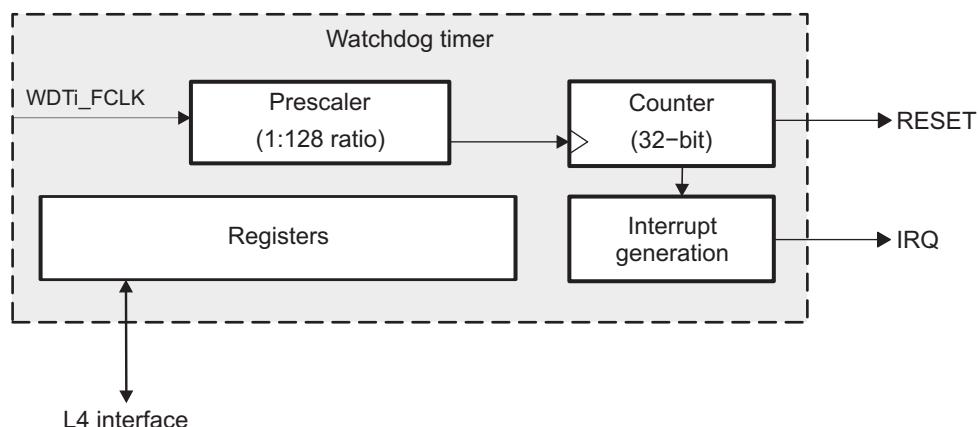
The watchdog timers are based on an upward 32-bit counter coupled with a prescaler. The counter overflow is signaled through two independent signals: a simple reset signal and an interrupt signal, both active low. [Figure 20-92](#) is a functional block diagram of the watchdog timer.

The interrupt generation mechanism is controlled through the WDT\_WIRQENSET/WDT\_WIRQENCLR and WDT\_WIRQSTAT registers.

The prescaler ratio can be set from 1 to 128 by accessing the WDT\_WCLR[4:2] PTV bit field and the WDT\_WCLR[5] PRE bit of the watchdog control register (WDT\_WCLR).

The current timer value can be accessed on-the-fly by reading the watchdog timer counter register (WDT\_WCRR), modified by accessing the watchdog timer load register (WDT\_WLDR) (no on-the-fly update), or reloaded by following a specific reload sequence on the watchdog timer trigger register (WDT\_WTGR). A start/stop sequence applied to the watchdog timer start/stop register (WDT\_WSPR) can start and stop the watchdog timers.

**Figure 20-92. 32-Bit Watchdog Timer Functional Block Diagram**



#### 20.4.3.4 Reset Context

The watchdog timers are enabled after reset. [Table 20-101](#) lists the default reset values of the two watchdog timer load registers (the WDT\_WLDR) and prescaler ratios (the WDT\_WCLR[4:2] PTV bit field). To get these values, software must read the corresponding WDT\_WCLR[4:2] PTV bit field and the 32-bit register to retrieve the static configuration of the module.

**Table 20-101. Count and Prescaler Default Reset Values**

| Timer | WDT_WLDR Reset Value | PTV Reset Value |
|-------|----------------------|-----------------|
| WDT   | FFFF FFBEH           | 0               |

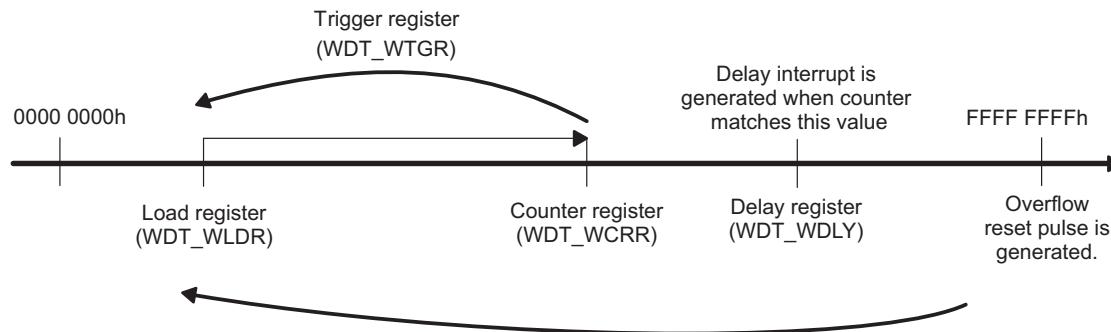
#### 20.4.3.5 Overflow/Reset Generation

When the watchdog timer counter register (WDT\_WCRR) overflows, an active-low reset pulse is generated to the PRCM module. This RESET pulse causes the PRCM module to generate global WARM reset of the device. It is also driven out of the device through the WD\_OUT pin. This pulse is one prescaled timer clock cycle wide and occurs at the same time as the timer counter overflow.

After reset generation, the counter is automatically reloaded with the value stored in the watchdog load register (WDT\_WLDR) and the prescaler is reset (the prescaler ratio remains unchanged). When the reset pulse output is generated, the timer counter begins incrementing again.

[Figure 20-93](#) shows a general functional view of the watchdog timers.

**Figure 20-93. Watchdog Timers General Functional View**



#### 20.4.3.6 Prescaler Value/Timer Reset Frequency

Each watchdog timer is composed of a prescaler stage and a timer counter.

The timer rate is defined by the following values:

- Value of the prescaler fields (the WDT\_WCLR[5] PRE bit and the WDT\_WCLR[4:2] PTV bit field)
- Value loaded into the timer load register (WDT\_WLDR)

The prescaler stage is clocked with the timer clock and acts as a clock divider for the timer counter stage. The ratio is managed by accessing the ratio definition field (the WDT\_WCLR[4:2] PTV bit field) and is enabled with the WDT\_WCLR[5] PRE bit.

[Table 20-102](#) lists the prescaler clock ratio values.

**Table 20-102. Prescaler Clock Ratio Values**

| <b>WDT_WCLR[5] PRE</b> | <b>WDT_WCLR[4:2] PTV</b> | <b>Clock Divider (PS)</b> |
|------------------------|--------------------------|---------------------------|
| 0                      | X                        | 1                         |
| 1                      | 0                        | 1                         |
| 1                      | 1                        | 2                         |
| 1                      | 2                        | 4                         |
| 1                      | 3                        | 8                         |
| 1                      | 4                        | 16                        |
| 1                      | 5                        | 32                        |
| 1                      | 6                        | 64                        |
| 1                      | 7                        | 128                       |

Thus the watchdog timer overflow rate is expressed as:

$$\text{OVF\_Rate} = (\text{FFFF FFFFh} - \text{WDT_WLDR} + 1) \times (\text{wd-functional clock period}) \times \text{PS}$$

where wd-functional clock period =  $1/(\text{wd-functional clock frequency})$  and  $\text{PS} = 2^{(\text{PTV})}$

#### CAUTION

Internal resynchronization causes some latency in any software write to WDT\_WSPR before WDT\_WSPR is updated with the programmed value:

$1.5 \times \text{functional clock cycles} \leq \text{write\_WDT\_WSPR\_latency} \leq 2.5 \times \text{functional clock cycles}$

Remember to consider this latency whenever the watchdog timer must be started or stopped.

For example, for a timer clock input of 32 kHz with a prescaler ratio value of 1 (clock divided by 2) and WDT\_WCLR[5] PRE = 1 (clock divider enabled), the reset period is as listed in [Table 20-103](#).

**Table 20-103. Reset Period Examples**

| <b>WDT_WLDR Value</b> | <b>Reset Period</b> |
|-----------------------|---------------------|
| 0000 0000h            | 74 h 56 min         |
| FFFF 0000h            | 4 s                 |
| FFFF FFF0h            | 1 ms                |
| FFFF FFFFh            | 62.5 us             |

#### CAUTION

- Ensure that the reloaded value allows the correct operation of the application. When a watchdog timer is enabled, software must periodically trigger a reload before the counter overflows. Hence, the value of the WDT\_WLDR[31:0] bit field must be chosen according to the ongoing activity preceding the watchdog reload.
- Due to design reasons, WDT\_WLDR[31:0] = FFFF FFFFh is a special case, although such a value of WDT\_WLDR is meaningless. When WDT\_WLDR is programmed with the overflow value, a triggering event generates a reset/interrupt one functional clock cycle later, even if the watchdog timer is stopped.

**Table 20-104** lists the default reset periods for the watchdog timers.

**Table 20-104. Default Watchdog Timer Reset Periods**

| Watchdog Timers | Clock Source | Default Reset Period |
|-----------------|--------------|----------------------|
| WDT             | 32 kHz       | 2 s                  |

#### 20.4.3.7 Triggering a Timer Reload

To reload the timer counter and reset the prescaler before reaching overflow, a reload command is executed by accessing the watchdog timer trigger register (WDT\_WTGR) using a specific reload sequence.

The specific reload sequence is performed whenever the written value on the WDT\_WTGR register differs from its previous value. In this case, reload is executed in the same way as an overflow autoreload, but without the generation of a reset pulse.

The timer counter is loaded with the value of the watchdog timer load register (the WDT\_WLDR[31:0] TIMER\_LOAD bit field), and the prescaler is reset.

#### 20.4.3.8 Start/Stop Sequence for Watchdog Timers (Using the WDT\_WSPR Register)

To start and stop a watchdog timer, access must be made through the start/stop register (WDT\_WSPR) using a specific sequence.

To disable the timer, follow this sequence:

1. Write XXXX AAAAh in WDT\_WSPR.
2. Poll for posted write to complete using WDT\_WWPS.W\_PEND\_WSPR.
3. Write XXXX 5555h in WDT\_WSPR.
4. Poll for posted write to complete using WDT\_WWPS.W\_PEND\_WSPR.

To enable the timer, follow this sequence:

1. Write XXXX BBBBh in WDT\_WSPR.
2. Poll for posted write to complete using WDT\_WWPS.W\_PEND\_WSPR.
3. Write XXXX 4444h in WDT\_WSPR.
4. Poll for posted write to complete using WDT\_WWPS.W\_PEND\_WSPR.

All other write sequences on the WDT\_WSPR register have no effect on the start/stop feature of the module.

#### 20.4.3.9 Modifying Timer Count/Load Values and Prescaler Setting

To modify the timer counter value (the WDT\_WCRR register), prescaler ratio (the WDT\_WCLR[4:2] PTV bit field), delay configuration value (the WDT\_WDLY[31:0] DLY\_VALUE bit field), or the load value (the WDT\_WLDR[31:0] TIMER\_LOAD bit field), the watchdog timer must be disabled by using the start/stop sequence (the WDT\_WSPR register).

After a write access, the load register value and prescaler ratio registers are updated immediately, but new values are considered only after the next consecutive counter overflow or after a new trigger command (the WDT\_WTGR register).

#### 20.4.3.10 Watchdog Counter Register Access Restriction (WDT\_WCRR Register)

A 32-bit shadow register is implemented to read a coherent value of the WDT\_WCRR register because the WDT\_WCRR register is directly related to the timer counter value and is updated on the timer clock (WDT\_FCLK). The shadow register is updated by a 16-bit LSB read command.

---

**NOTE:** Although the L4 clock (WDT\_ICLK) is completely asynchronous with the timer clock (WDT\_FCLK), some synchronization is performed to ensure that the value of the WDT\_WCRR register is not read while it is being incremented.

---

When 32-bit read access is performed, the shadow register is not updated. Read access is performed directly from the accessed register.

To ensure that a coherent value is read inside WDT\_WCRR, the first read access is to the lower 16 bits (offset = 8h), followed by read access to the upper 16 bits (offset = Ah).

#### 20.4.3.11 Watchdog Timer Interrupt Generation

When an interrupt source occurs, the interrupt status bit (the WDT\_WIRQSTAT[0] EVENT\_OVF or WDT\_WIRQSTAT[1] EVENT\_DLY bit) is set to 1. The output interrupt line (WDTi\_IRQ) is asserted (active low) when status (the EVENT\_xxx bit) and enable (the xxx\_IT\_ENA bit) flags are set to 1; the order is not relevant. Writing 1 to the enable bit (the status is already set at 1) also triggers the interrupt in the normal order (enable first, status next). The pending interrupt event is cleared when the set status bit is overwritten by a value of 1 by a write command in the WDT\_WIRQSTAT register. Reading the WDT\_WIRQSTAT register and writing the value back allows a fast interrupt acknowledge process.

The watchdog timer issues an overflow interrupt if this interrupt is enabled in the watchdog interrupt enable register (WDT\_WIRQENSET[0] OVF\_IT\_ENA = 1). When the overflow occurs, the interrupt status bit (the WDT\_WIRQSTAT[0] EVENT\_OVF bit) is set to 1. The output interrupt line (WDT\_IRQ) is asserted (active low) when status (EVENT\_OVF) and enable (OVF\_IT\_ENA) flags are set to 1; the order is not relevant. This interrupt can be disabled by setting the WDT\_WIRQENCLR[0] OVF\_IT\_ENA bit to 1.

The watchdog can issue the delay interrupt if this interrupt is enabled in the interrupt enable register (WDT\_WIRQENSET[1] DLY\_IT\_ENA = 1). When the counter is running and the counter value matches the value stored in the delay configuration register (WDT\_WDLY), the corresponding interrupt status bit is set in the watchdog status register (WDT\_WIRQSTAT) and the output interrupt line is asserted (active low) when the flag (EVENT\_DLY) and enable (DLY\_IT\_ENA) bits are 1 in the WDT\_WIRQSTAT and WDT\_WIRQENSET registers, respectively; the order (normally enable, then flag), is not relevant. This interrupt can be disabled by setting the WDT\_WIRQENCLR[1] DLY\_IT\_ENA bit to 1.

---

**NOTE:** Writing 0 to the WDT\_WIRQSTAT[0] EVENT\_OVF bit or the WDT\_WIRQSTAT[1] EVENT\_DLY bit has no effect.

---

The two clock domains are resynchronized because the interrupt event is generated on the functional clock domain (WDTi\_FCLK) during the updating of the interrupt status register (WDT\_WIRQSTAT).

The WDT\_WDLY register is used to specify the value of the delay configuration register. The delay time to interrupt is the difference between the reload value stored in the counter load register (WDT\_WLDR) and the programmed value in this register (WDT\_WDLY).

Use the following formula to estimate the delay time:

$$\text{Delay time period} = (WDT_WDLY - WDT_WLDR + 1) \times \text{Timer clock period} \times \text{Clock divider}$$

Where:

- Timer clock period = 1/(Timer clock frequency)
- Clock divider = 2PTV

If the counter value (WDT\_WCRR) reaches the programmed value (WDT\_WDLY), the status bit (EVENT\_DLY) gets set in the interrupt status register (WDT\_WIRQSTAT), and an interrupt occurs if the corresponding enable bit is set in the interrupt enable register (WDT\_WIRQENSET).

**CAUTION**

If the reload event occurs (after a triggering sequence or after a reset sequence) before reaching the programmed value (WDT\_WDLY[31:0] WDLY\_VALUE), no interrupt is generated.

Also, no interrupt is generated if the value programmed in the delay configuration register (WDT\_WDLY) is less than the value stored in the counter load register (WDT\_WLDR).

#### 20.4.3.12 Watchdog Timers Under Emulation

During emulation mode, the watchdog timer can/cannot continue running, according to the value of the WDT\_WDSC[5] EMUFREE bit of the system configuration register (WDT\_WDSC).

- When EMUFREE is 1, watchdog timer execution is not stopped and a reset pulse is still generated when overflow is reached.
- When EMUFREE is 0, the counters (prescaler/timer) are frozen and incrementation restarts after exiting from emulation mode.

#### 20.4.3.13 Accessing Watchdog Timer Registers

Posted/nonposted selection applies only to functional registers that require synchronization on/from the timer functional clock domain (WDTi\_FCLK). For write/read operation, the following registers are affected:

- WDT\_WCLR
- WDT\_WCRR
- WDT\_WLDR
- WDT\_WTGR
- WDT\_WDLY
- WDT\_WSPR

The timer interface clock domain synchronous registers are not affected by the posted/nonposted selection; the write/read operation is effective and acknowledged (command accepted) after one WDT\_ICLK cycle from the command assertion. The timer interface clock domain synchronous registers are:

- WDT\_WIDR
- WDT\_WDSC
- WDT\_WDST
- WDT\_WIRQSTATRAW
- WDT\_WIRQSTAT
- WDT\_WIRQENSET
- WDT\_WIRQENCLR
- WDT\_WWPS

#### **20.4.3.14 Low-Level Programming Model**

This section covers the low-level hardware programming sequences for configuration and use of the module.

##### **20.4.3.14.1 Global Initialization**

###### **20.4.3.14.1.1 Surrounding Modules Global Initialization**

This section identifies the requirements for initializing the surrounding modules when the watchdog timer is to be used for the first time after a device reset. This initialization of surrounding modules is based on the integration and environment of the watchdog timer (see [Table 20-105](#)).

**Table 20-105. Global Initialization of Surrounding Modules**

| Surrounding Modules | Comments   |
|---------------------|--|
| PRCM                | The module interface and functional clocks must be enabled.                                    |
| Control module      | Module-specific pad multiplexing must be set in the control module.                            |
| MPU INTC            | The MPU INTC configuration must be performed to enable the interrupts from the watchdog timer. |

###### **20.4.3.14.1.2 Main Sequence – Watchdog Timer Module Global Initialization**

[Table 20-106](#) lists the steps for initializing the watchdog timer module when the module is to be used for the first time.

**Table 20-106. Watchdog Timer Module Global Initialization**

| Step                       | Register/Bit Field/Programming Model | Value |
|----------------------------|--------------------------------------|-------|
| Execute software reset.    | WDT_WDSC[1] SOFTRESET                | 1     |
| Wait until reset release?  | WDT_WDSC[1] SOFTRESET                | 0     |
| Enable delay interrupt.    | WDT_WIRQENSET[1] ENABLE_DLY          | 1     |
| Enable overflow interrupt. | WDT_WIRQENSET[0] ENABLE_OVF          | 1     |

#### **20.4.3.14.2 Operational Mode Configuration**

##### **20.4.3.14.2.1 Main Sequence – Watchdog Timer Basic Configuration**

[Table 20-107](#) lists the steps for the basic configuration of the watchdog timer.

**Table 20-107. Watchdog Timer Basic Configuration**

| Step                            | Register/Bit Field/Programming Model        | Value |
|---------------------------------|---|-------|
| Disable the watchdog timer.     | See <a href="#">Section 20.4.3.14.2.2</a> . |       |
| Set prescaler value.            | WDT_WCLR[4:2] PTV                           | xxx   |
| Enable prescaler.               | WDT_WCLR[5] PRE                             | 1     |
| Load delay configuration value. | WDT_WDLY                                    | xxx   |
| Load timer counter value.       | WDT_WCRR                                    | xxx   |
| Enable the watchdog timer.      | See <a href="#">Section 20.4.3.14.2.3</a> . |       |

#### 20.4.3.14.2.2 Subsequence – Disable the Watchdog Timer

Table 20-108 lists the steps to disable the watchdog timer.

**Table 20-108. Disable the Watchdog Timer**

| Step                          | Register/Bit Field/Programming Model | Value      |
|-------------------------------|--------------------------------------|------------|
| Write disable sequence Data1. | WDT_WSPR                             | XXXX AAAAh |
| Write disable sequence Data2. | WDT_WSPR                             | XXXX 5555h |

#### 20.4.3.14.2.3 Subsequence – Enable the Watchdog Timer

Table 20-109 lists the steps to enable the watchdog timer.

**Table 20-109. Enable the Watchdog Timer**

| Step                         | Register/Bit Field/Programming Model | Value      |
|------------------------------|--------------------------------------|------------|
| Write enable sequence Data1. | WDT_WSPR                             | XXXX BBBBh |
| Write enable sequence Data2. | WDT_WSPR                             | XXXX 4444h |

### 20.4.4 Watchdog Registers

**CAUTION**

The watchdog timers registers are limited to 32-bit and 16-bit data accesses; 8-bit access is not allowed and can corrupt register content.

**NOTE:**

- The WDT\_WISR and WDT\_WIRQSTATRAW registers have the same functionality. The WDT\_WISR register is used for software backward compatibility.
- The WDT\_WIER and WDT\_WIRQENSET/WDT\_WIRQENCLR registers have the same functionality. The WDT\_WIER register is used for software backward compatibility.
- The WDT\_WIRQSTATRAW and WDT\_WIRQSTAT registers give the same information when read. The WDT\_WIRQSTATRAW register is used for debug.

#### 20.4.4.1 WATCHDOG\_TIMER Registers

Table 20-110 lists the memory-mapped registers for the WATCHDOG\_TIMER. All register offset addresses not listed in Table 20-110 should be considered as reserved locations and the register contents should not be modified.

**Table 20-110. WATCHDOG\_TIMER REGISTERS**

| Offset | Acronym  | Register Name | Section            |
|--------|----------|---------------|--------------------|
| 0h     | WDT_WIDR | WDT_WIDR      | Section 20.4.4.1.1 |
| 10h    | WDT_WDSC | WDT_WDSC      | Section 20.4.4.1.2 |
| 14h    | WDT_WDST | WDT_WDST      | Section 20.4.4.1.3 |
| 18h    | WDT_WISR | WDT_WISR      | Section 20.4.4.1.4 |
| 1Ch    | WDT_WIER | WDT_WIER      | Section 20.4.4.1.5 |
| 24h    | WDT_WCLR | WDT_WCLR      | Section 20.4.4.1.6 |
| 28h    | WDT_WCRR | WDT_WCRR      | Section 20.4.4.1.7 |
| 2Ch    | WDT_WLDR | WDT_WLDR      | Section 20.4.4.1.8 |

**Table 20-110. WATCHDOG\_TIMER REGISTERS (continued)**

| <b>Offset</b> | <b>Acronym</b>  | <b>Register Name</b> | <b>Section</b>                      |
|---------------|-----------------|----------------------|-------------------------------------|
| 30h           | WDT_WTGR        | WDT_WTGR             | <a href="#">Section 20.4.4.1.9</a>  |
| 34h           | WDT_WWPS        | WDT_WWPS             | <a href="#">Section 20.4.4.1.10</a> |
| 44h           | WDT_WDLY        | WDT_WDLY             | <a href="#">Section 20.4.4.1.11</a> |
| 48h           | WDT_WSPR        | WDT_WSPR             | <a href="#">Section 20.4.4.1.12</a> |
| 54h           | WDT_WIRQSTATRAW | WDT_WIRQSTATRAW      | <a href="#">Section 20.4.4.1.13</a> |
| 58h           | WDT_WIRQSTAT    | WDT_WIRQSTAT         | <a href="#">Section 20.4.4.1.14</a> |
| 5Ch           | WDT_WIRQENSET   | WDT_WIRQENSET        | <a href="#">Section 20.4.4.1.15</a> |
| 60h           | WDT_WIRQENCLR   | WDT_WIRQENCLR        | <a href="#">Section 20.4.4.1.16</a> |

#### 20.4.4.1.1 WDT\_WIDR Register (offset = 0h) [reset = 0h]

WDT\_WIDR

WDT\_WIDR is shown in [Figure 20-94](#) and described in [Table 20-111](#).

**Figure 20-94. WDT\_WIDR Register**

|          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31       | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REVISION |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-111. WDT\_WIDR Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description |
|------|----------|------|-------|-------------|
| 31-0 | REVISION | R    | 0h    | IP Revision |

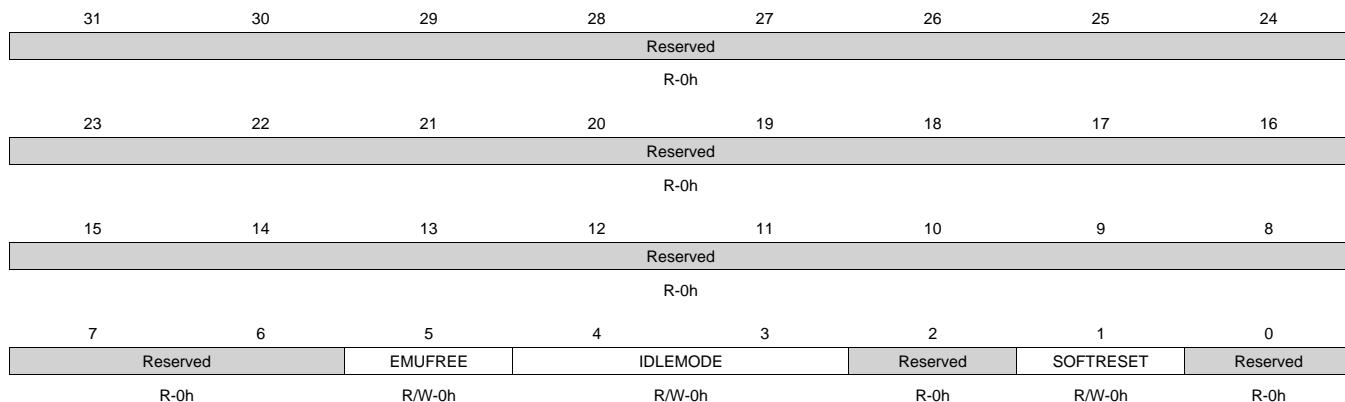
#### 20.4.4.1.2 WDT\_WDSC Register (offset = 10h) [reset = 0h]

##### WDT\_WDSC

WDT\_WDSC is shown in [Figure 20-95](#) and described in [Table 20-112](#).

This register controls the various parameters of the L4 interface.

**Figure 20-95. WDT\_WDSC Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-112. WDT\_WDSC Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-6 | Reserved  | R    | 0h    |  |
| 5    | EMUFREE   | R/W  | 0h    | Emulation mode<br>0x0 = Timer counter frozen in emulation<br>0x1 = Timer counter free-running in emulation   |
| 4-3  | IDLEMODE  | R/W  | 2h    | Configuration of the local target state management mode.By definition, target can handle read/write transaction as long as it is out of IDLE state.<br>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only.<br>0x1 = No-idle mode: local target never enters idle state.Backup mode, for debug only.<br>0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events.<br>0x3 = Smart-idle wakeup-capable mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module may generate (IRQ- or DMA-request-related) wakeup events when in idle state. |
| 2    | Reserved  | R    | 0h    |  |
| 1    | SOFTRESET | R/W  | 0h    | Software reset.<br>(Optional)<br>0x0(W) = No action<br>0x0(R) = Reset done, no pending action<br>0x1(W) = Initiate software reset.<br>0x1(R) = Reset (software or other) ongoing   |
| 0    | Reserved  | R    | 0h    |  |

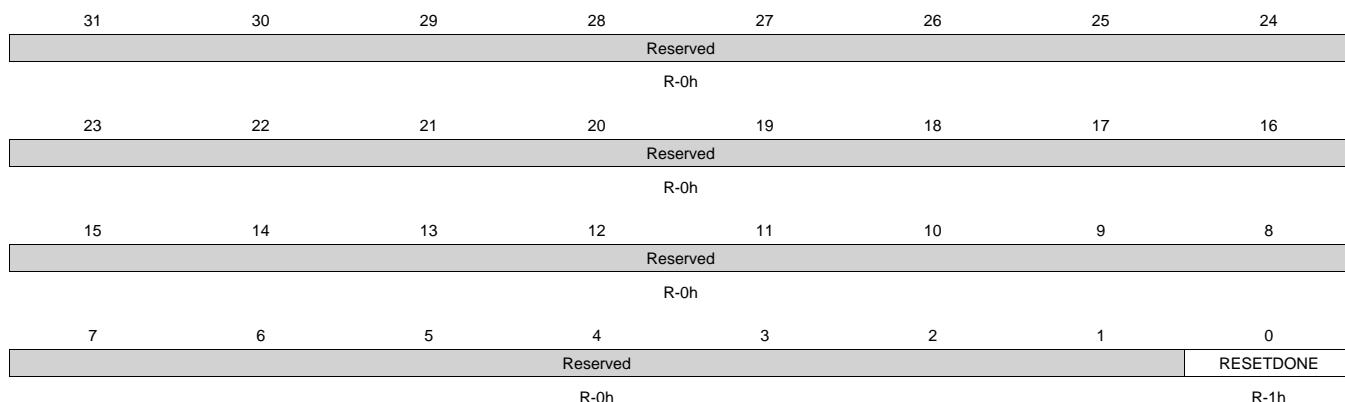
#### 20.4.4.1.3 WDT\_WDST Register (offset = 14h) [reset = 1h]

WDT\_WDST

WDT\_WDST is shown in [Figure 20-96](#) and described in [Table 20-113](#).

The WDT\_WDST register provides status information about the module.

**Figure 20-96. WDT\_WDST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-113. WDT\_WDST Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-1 | Reserved  | R    | 0h    |  |
| 0    | RESETDONE | R    | 1h    | Internal module reset monitoring<br>0x0 = Internal module reset is ongoing.<br>0x1 = Reset completed |

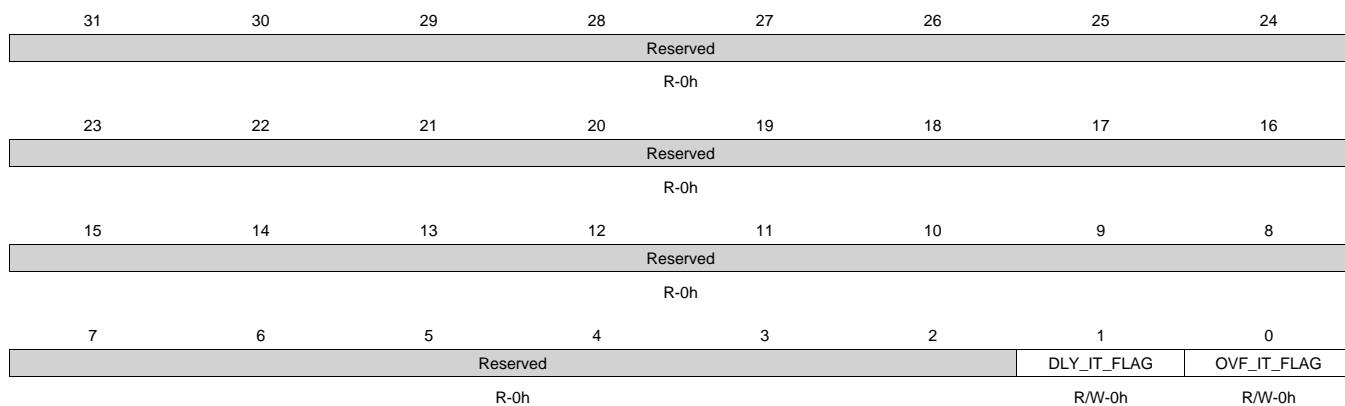
#### **20.4.4.1.4 WDT\_WISR Register (offset = 18h) [reset = 0h]**

##### **WDT\_WISR**

WDT\_WISR is shown in [Figure 20-97](#) and described in [Table 20-114](#).

This register shows which interrupt events are pending inside the module.

**Figure 20-97. WDT\_WISR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-114. WDT\_WISR Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description   |
|------|-------------|------|-------|---|
| 31-2 | Reserved    | R    | 0h    |   |
| 1    | DLY_IT_FLAG | R/W  | 0h    | Pending delay interrupt status.<br>0x0(W) = Status unchanged<br>0x0(R) = No delay interrupt pending<br>0x1(W) = Status bit cleared<br>0x1(R) = Delay interrupt pending          |
| 0    | OVF_IT_FLAG | R/W  | 0h    | Pending overflow interrupt status.<br>0x0(W) = Status unchanged<br>0x0(R) = No overflow interrupt pending<br>0x1(W) = Status bit cleared<br>0x1(R) = Overflow interrupt pending |

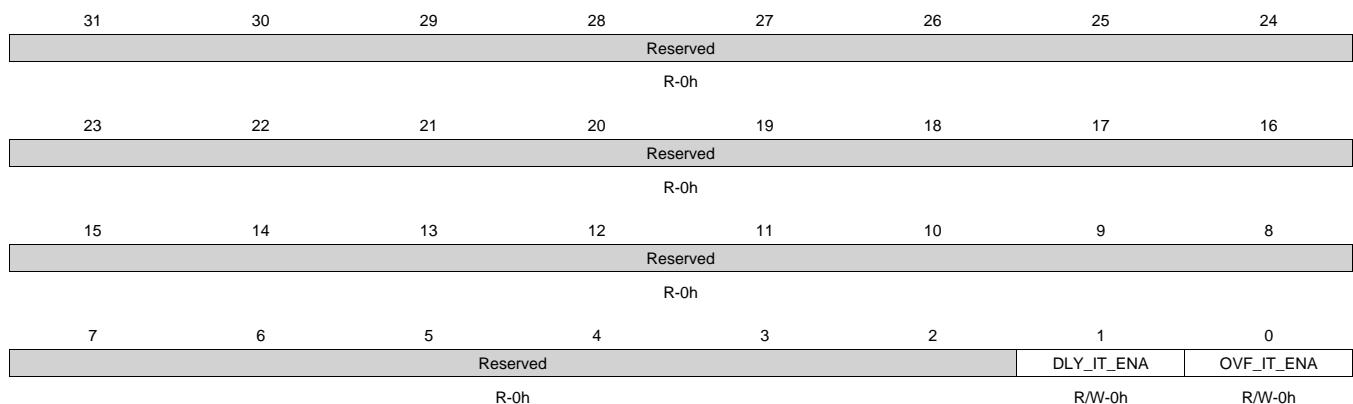
#### 20.4.4.1.5 WDT\_WIER Register (offset = 1Ch) [reset = 0h]

##### WDT\_WIER

WDT\_WIER is shown in [Figure 20-98](#) and described in [Table 20-115](#).

The WDT\_WIERs register controls (enable/disable) the interrupt events.

**Figure 20-98. WDT\_WIER Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-115. WDT\_WIER Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-2 | Reserved   | R    | 0h    |  |
| 1    | DLY_IT_ENA | R/W  | 0h    | Delay interrupt enable/disable<br>0x0 = Disable delay interrupt.<br>0x1 = Enable delay interrupt.          |
| 0    | OVF_IT_ENA | R/W  | 0h    | Overflow interrupt enable/disable<br>0x0 = Disable overflow interrupt.<br>0x1 = Enable overflow interrupt. |

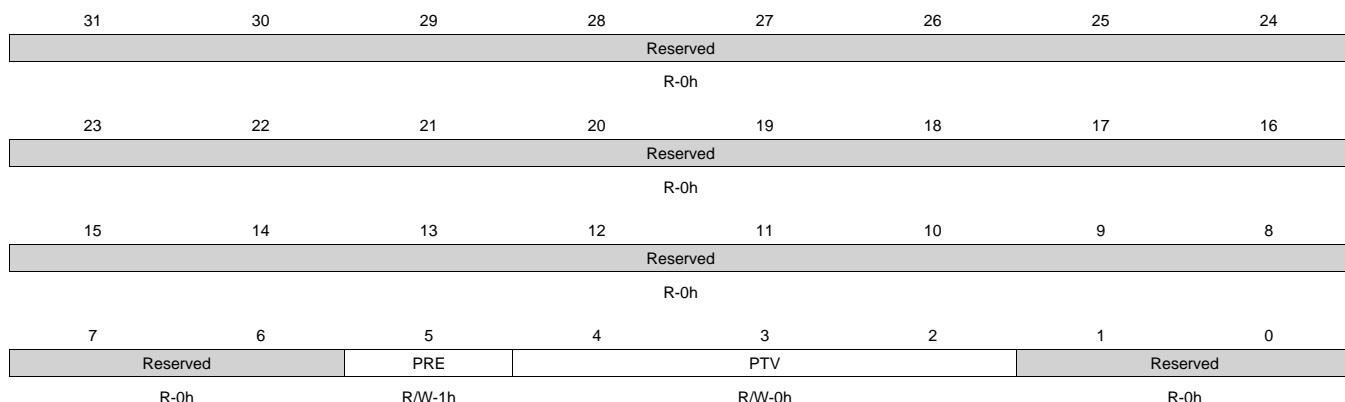
#### **20.4.4.1.6 WDT\_WCLR Register (offset = 24h) [reset = 20h]**

WDT\_WCLR

WDT\_WCLR is shown in [Figure 20-99](#) and described in [Table 20-116](#).

The WDT\_WCLR register controls the prescaler stage of the counter.

**Figure 20-99. WDT\_WCLR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-116. WDT\_WCLR Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-6 | Reserved | R    | 0h    |   |
| 5    | PRE      | R/W  | 1h    | Prescaler enable/disable configuration<br>0x0 = Prescaler disabled<br>0x1 = Prescaler enabled   |
| 4-2  | PTV      | R/W  | 0h    | Prescaler value The timer counter is prescaled with the value:<br>$2^{**}PTV$ .<br>Example: PTV = 3 then counter increases value if started after 8 functional clock periods.<br>On reset, it is loaded from PI_PTV_RESET_VALUE input port. |
| 1-0  | Reserved | R    | 0h    |   |

#### 20.4.4.1.7 WDT\_WCRR Register (offset = 28h) [reset = 0h]

WDT\_WCRR

WDT\_WCRR is shown in [Figure 20-100](#) and described in [Table 20-117](#).

The WDT\_WCRR register holds the value of the internal counter.

**Figure 20-100. WDT\_WCRR Register**

|               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31            | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER_COUNTER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-117. WDT\_WCRR Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description                         |
|------|---------------|------|-------|-------------------------------------|
| 31-0 | TIMER_COUNTER | R/W  | 0h    | Value of the timer counter register |

#### **20.4.4.1.8 WDT\_WLDR Register (offset = 2Ch) [reset = 0h]**

WDT\_WLDR

WDT\_WLDR is shown in [Figure 20-101](#) and described in [Table 20-118](#).

The WDT\_WLDR register holds the timer load value.

**Figure 20-101. WDT\_WLDR Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER_LOAD |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-118. WDT\_WLDR Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                      |
|------|------------|------|-------|----------------------------------|
| 31-0 | TIMER_LOAD | R/W  | 0h    | Value of the timer load register |

#### 20.4.4.1.9 WDT\_WTGR Register (offset = 30h) [reset = 0h]

WDT\_WTGR

WDT\_WTGR is shown in [Figure 20-102](#) and described in [Table 20-119](#).

Writing a different value than the one already written in this register does a watchdog counter reload.

**Figure 20-102. WDT\_WTGR Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TTGR_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-119. WDT\_WTGR Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                   |
|------|------------|------|-------|-------------------------------|
| 31-0 | TTGR_VALUE | R/W  | 0h    | Value of the trigger register |

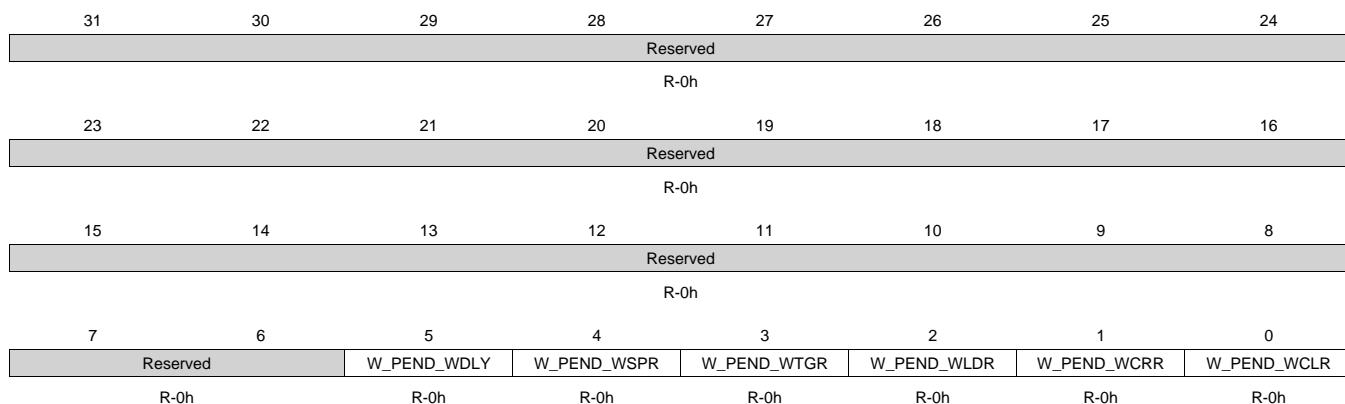
#### 20.4.4.1.10 WDT\_WWPS Register (offset = 34h) [reset = 0h]

##### WDT\_WWPS

WDT\_WWPS is shown in [Figure 20-103](#) and described in [Table 20-120](#).

This register contains the write posting bits for all writeable functional registers.

**Figure 20-103. WDT\_WWPS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-120. WDT\_WWPS Register Field Descriptions**

| Bit  | Field       | Type | Reset | Description  |
|------|-------------|------|-------|--|
| 31-6 | Reserved    | R    | 0h    |  |
| 5    | W_PEND_WDLY | R    | 0h    | Write pending for register WDLY<br>0x0 = No register write pending<br>0x1 = Register write pending |
| 4    | W_PEND_WSPR | R    | 0h    | Write pending for register WSPR<br>0x0 = No register write pending<br>0x1 = Register write pending |
| 3    | W_PEND_WTGR | R    | 0h    | Write pending for register WTGR<br>0x0 = No register write pending<br>0x1 = Register write pending |
| 2    | W_PEND_WLDR | R    | 0h    | Write pending for register WLDR<br>0x0 = No register write pending<br>0x1 = Register write pending |
| 1    | W_PEND_WCRR | R    | 0h    | Write pending for register WCRR<br>0x0 = No register write pending<br>0x1 = Register write pending |
| 0    | W_PEND_WCLR | R    | 0h    | Write pending for register WCLR<br>0x0 = No register write pending<br>0x1 = Register write pending |

#### 20.4.4.1.11 WDT\_WDLY Register (offset = 44h) [reset = 0h]

WDT\_WDLY

WDT\_WDLY is shown in [Figure 20-104](#) and described in [Table 20-121](#).

The WDT\_WDLY register holds the delay value that controls the internal pre-overflow event detection.

**Figure 20-104. WDT\_WDLY Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| WDLY_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-121. WDT\_WDLY Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                 |
|------|------------|------|-------|-----------------------------|
| 31-0 | WDLY_VALUE | R/W  | 0h    | Value of the delay register |

#### **20.4.4.1.12 WDT\_WSPR Register (offset = 48h) [reset = 0h]**

WDT\_WSPR

WDT\_WSPR is shown in [Figure 20-105](#) and described in [Table 20-122](#).

The WDT\_WSPR register holds the start-stop value that controls the internal start-stop FSM.

**Figure 20-105. WDT\_WSPR Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WSPR_VALUE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-122. WDT\_WSPR Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description                      |
|------|------------|------|-------|----------------------------------|
| 31-0 | WSPR_VALUE | R/W  | 0h    | Value of the start-stop register |

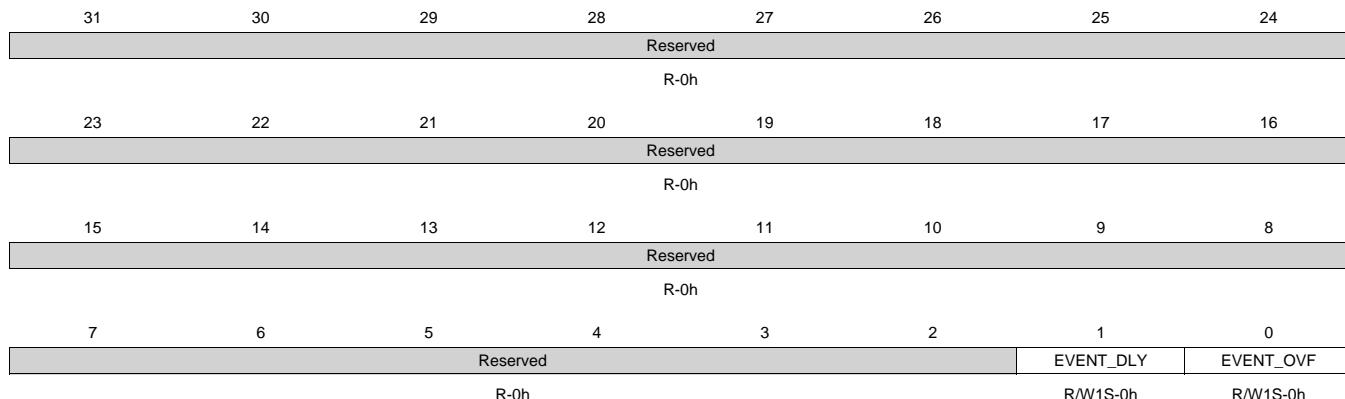
**20.4.4.1.13 WDT\_WIRQSTATRAW Register (offset = 54h) [reset = 0h]**

WDT\_WIRQSTATRAW

WDT\_WIRQSTATRAW is shown in [Figure 20-106](#) and described in [Table 20-123](#).

IRQ unmasked status, status set per-event raw interrupt status vector, line 0. Raw status is set even if event is not enabled. Write 1 to set the (raw) status, mostly for debug.

**Figure 20-106. WDT\_WIRQSTATRAW Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-123. WDT\_WIRQSTATRAW Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-2 | Reserved  | R     | 0h    |   |
| 1    | EVENT_DLY | R/W1S | 0h    | Settable raw status for delay event<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Set event (debug)<br>0x1(R) = Event pending    |
| 0    | EVENT_OVF | R/W1S | 0h    | Settable raw status for overflow event<br>0x0(W) = No action<br>0x0(R) = No event pending<br>0x1(W) = Set event (debug)<br>0x1(R) = Event pending |

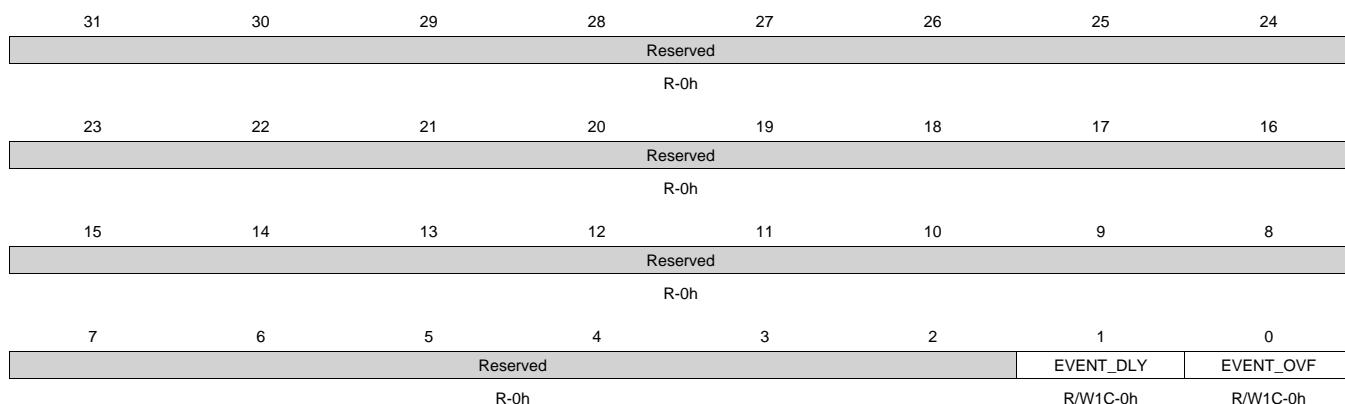
#### **20.4.4.1.14 WDT\_WIRQSTAT Register (offset = 58h) [reset = 0h]**

##### **WDT\_WIRQSTAT**

WDT\_WIRQSTAT is shown in [Figure 20-107](#) and described in [Table 20-124](#).

IRQ masked status, status clear per-event enabled interrupt status vector, line 0. Enabled status is not set unless event is enabled. Write 1 to clear the status after interrupt has been serviced (raw status gets cleared, that is, even if not enabled).

**Figure 20-107. WDT\_WIRQSTAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-124. WDT\_WIRQSTAT Register Field Descriptions**

| Bit  | Field     | Type  | Reset | Description   |
|------|-----------|-------|-------|---|
| 31-2 | Reserved  | R     | 0h    |   |
| 1    | EVENT_DLY | R/W1C | 0h    | Clearable, enabled status for delay event<br>0x0(W) = No action<br>0x0(R) = No (enabled) event pending<br>0x1(W) = Clear (raw) event<br>0x1(R) = Event pending    |
| 0    | EVENT_OVF | R/W1C | 0h    | Clearable, enabled status for overflow event<br>0x0(W) = No action<br>0x0(R) = No (enabled) event pending<br>0x1(W) = Clear (raw) event<br>0x1(R) = Event pending |

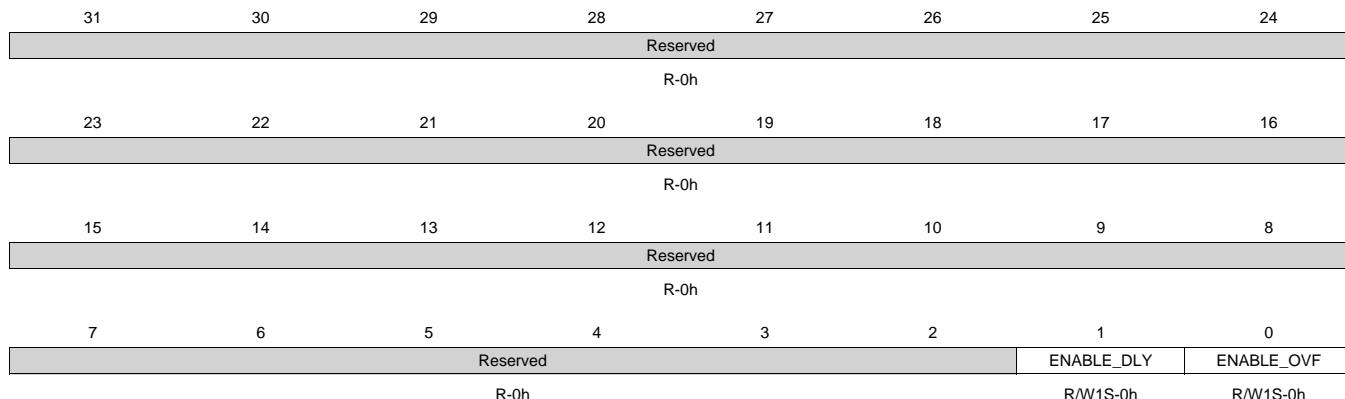
**20.4.4.1.15 WDT\_WIRQENSET Register (offset = 5Ch) [reset = 0h]**

WDT\_WIRQENSET

WDT\_WIRQENSET is shown in [Figure 20-108](#) and described in [Table 20-125](#).

IRQ enable set per-event interrupt enable bit vector, line 0. Write 1 to set (enable interrupt). Readout equal to corresponding \_CLR register.

**Figure 20-108. WDT\_WIRQENSET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

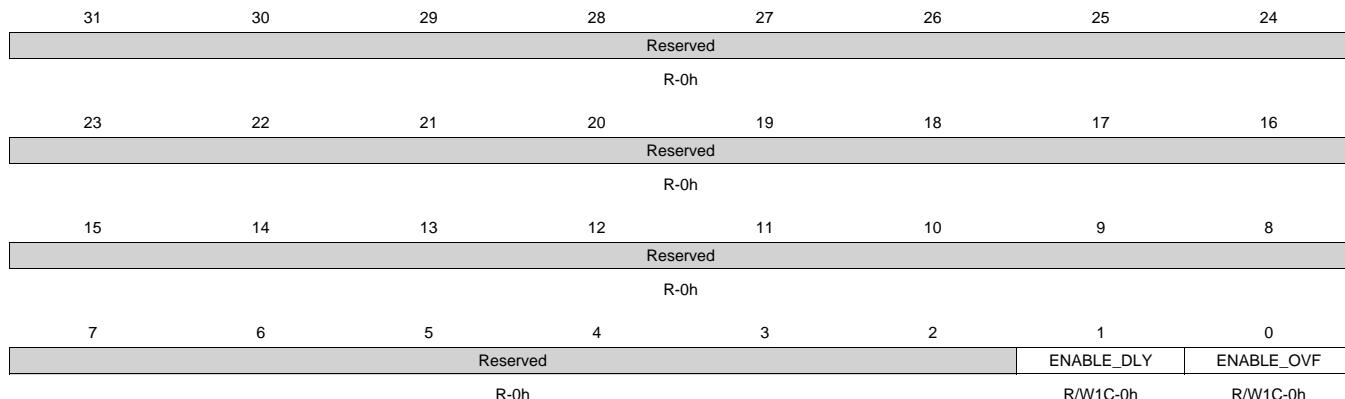
**Table 20-125. WDT\_WIRQENSET Register Field Descriptions**

| Bit  | Field      | Type  | Reset | Description   |
|------|------------|-------|-------|---|
| 31-2 | Reserved   | R     | 0h    |   |
| 1    | ENABLE_DLY | R/W1S | 0h    | Enable for delay event<br>0x0(W) = No action<br>0x0(R) = Interrupt disabled (masked)<br>0x1(W) = Enable interrupt.<br>0x1(R) = Interrupt enabled    |
| 0    | ENABLE_OVF | R/W1S | 0h    | Enable for overflow event<br>0x0(W) = No action<br>0x0(R) = Interrupt disabled (masked)<br>0x1(W) = Enable interrupt.<br>0x1(R) = Interrupt enabled |

**20.4.4.1.16 WDT\_WIRQENCLR Register (offset = 60h) [reset = 0h]**
**WDT\_WIRQENCLR**

 WDT\_WIRQENCLR is shown in [Figure 20-109](#) and described in [Table 20-126](#).

IRQ enable clear per-event interrupt enable bit vector, line 0. Write 1 to clear (disable interrupt). Readout equal to corresponding \_SET register.

**Figure 20-109. WDT\_WIRQENCLR Register**


LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 20-126. WDT\_WIRQENCLR Register Field Descriptions**

| Bit  | Field      | Type  | Reset | Description  |
|------|------------|-------|-------|--|
| 31-2 | Reserved   | R     | 0h    |  |
| 1    | ENABLE_DLY | R/W1C | 0h    | Enable for delay event<br>0x0(W) = No action<br>0x0(R) = Interrupt disabled (masked)<br>0x1(W) = Disable interrupt.<br>0x1(R) = Interrupt enabled    |
| 0    | ENABLE_OVF | R/W1C | 0h    | Enable for overflow event<br>0x0(W) = No action<br>0x0(R) = Interrupt disabled (masked)<br>0x1(W) = Disable interrupt.<br>0x1(R) = Interrupt enabled |

This chapter describes the I2C of the device.

| Topic                                    | Page        |
|--|-------------|
| <b>21.1 Introduction .....</b>           | <b>3681</b> |
| <b>21.2 Integration .....</b>            | <b>3682</b> |
| <b>21.3 Functional Description .....</b> | <b>3684</b> |
| <b>21.4 I2C Registers .....</b>          | <b>3698</b> |

## 21.1 Introduction

The multi-master I<sup>2</sup>C peripheral provides an interface between a CPU and any I<sup>2</sup>C-bus-compatible device that connects via the I<sup>2</sup>C serial bus. External components attached to the I<sup>2</sup>C bus can serially transmit/receive up to 8-bit data to/from the CPU device through the two-wire I<sup>2</sup>C interface.

The I<sup>2</sup>C bus is a multi-master bus. The I<sup>2</sup>C controller does support the multi-master mode that allows more than one device capable of controlling the bus to be connected to it. Each I<sup>2</sup>C device is recognized by a unique address and can operate as either transmitter or receiver, according to the function of the device. In addition to being a transmitter or receiver, a device connected to the I<sup>2</sup>C bus can also be considered as master or slave when performing data transfers. Note that a master device is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave.

### 21.1.1 I<sup>2</sup>C Features

The general features of the I<sup>2</sup>C controller are:

- Compliant with Philips I<sup>2</sup>C specification version 2.1
- Supports OmniVision Serial Camera Control Bus Protocol (SCCB)
- Supports standard mode (up to 100K bits/s) and fast mode (up to 400K bits/s).
- Multimaster transmitter/slave receiver mode
- Multimaster receiver/slave transmitter mode
- Combined master transmit/receive and receive/transmit modes
- 7-bit and 10-bit device addressing modes
- Built-in 32-byte FIFO for buffered read or writes in each module
- Programmable clock generation
- Two DMA channels, one interrupt line

### 21.1.2 Unsupported I<sup>2</sup>C Features

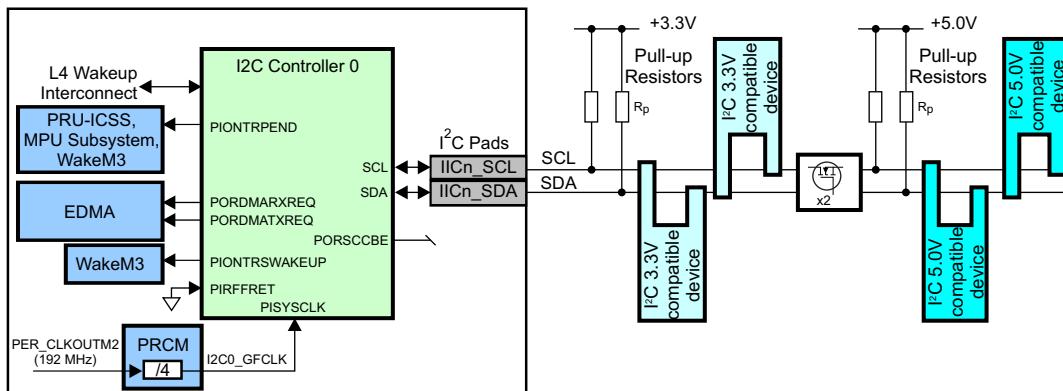
The I<sup>2</sup>C module features not supported in this device are shown in [Table 21-1](#).

**Table 21-1. Unsupported I<sup>2</sup>C Features**

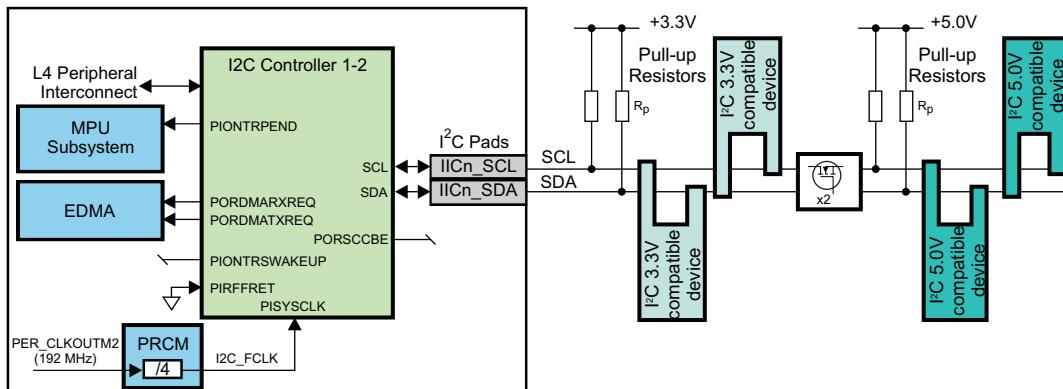
| Feature                         | Reason                     |
|---------------------------------|----------------------------|
| SCCB Protocol                   | SCCB signal not pinned out |
| High Speed (3.4 MBPS) operation | Not required               |

## 21.2 Integration

This device includes three instantiations of the I<sup>2</sup>C module. This peripheral implements the multi-master I<sup>2</sup>C bus which allows serial transfer of 8-bit data to/from other I<sup>2</sup>C master/slave devices through a two-wire interface. There are three I<sup>2</sup>C modules instantiations called I<sup>2</sup>C0, I<sup>2</sup>C1, and I<sup>2</sup>C2. The I<sup>2</sup>C0 module is located in the Wake-up power domain. Figure 21-1 and Figure 21-2 show examples of a system with multiple I<sup>2</sup>C-compatible devices.



**Figure 21-1. I<sup>2</sup>C0 Integration and Bus Application**



**Figure 21-2. I<sup>2</sup>C(1-2) Integration and Bus Application**

### 21.2.1 I<sup>2</sup>C Connectivity Attributes

The general connectivity attributes for the I<sup>2</sup>C module are shown in Table 21-2 and Table 21-3.

**Table 21-2. I<sup>2</sup>C0 Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Wakeup Domain   |
| Clock Domain        | PD_WKUP_L4_WKUP_GCLK (Interface/OCP)<br>PD_WKUP_I2C0_GFCLK (Func)         |
| Reset Signals       | WKUP_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle / Wakeup   |
| Interrupt Requests  | 1 interrupt to MPU Subsystem (I <sup>2</sup> C0INT), PRU-ICSS, and WakeM3 |
| DMA Requests        | 2 DMA requests to EDMA (I <sup>2</sup> CTXEVTO, I <sup>2</sup> CRXEVTO)   |
| Physical Address    | L4 Wakeup slave port  |

**Table 21-3. I2C(1–2) Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L4LS_GCLK (Interface/OCP)<br>PD_PER_I2C_FCLK (Func)   |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 1 interrupt per instance to MPU Subsystem (I2C1INT, I2C2INT) |
| DMA Requests        | 2 DMA requests per instance to EDMA (I2CTXEVTx, I2CRXEVTx)   |
| Physical Address    | L4 Peripheral slave port                                     |

### 21.2.2 I2C Clock and Reset Management

The I2C controllers have separate bus interface and functional clocks. During power-down mode, the I2Cx\_SCL and I2Cx\_SDA are configured as inputs.

**Table 21-4. I2C Clock Signals**

| Clock Signal                  | Max Freq | Reference / Source | Comments                          |
|-------------------------------|----------|--------------------|-----------------------------------|
| <b>I2C0 Clock Signals</b>     |          |                    |                                   |
| PIOCPCLK<br>Interface clock   | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_wkup_l4_wkup_gclk<br>From PRCM |
| PISYSCLK<br>Functional clock  | 48 MHz   | PER_CLKOUTM2 / 4   | pd_wkup_i2c0_gfclk<br>From PRCM   |
| <b>I2C(1-2) Clock Signals</b> |          |                    |                                   |
| PIOCPCLK<br>Interface clock   | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>From PRCM     |
| PISYSCLK<br>Functional clock  | 48 MHz   | PER_CLKOUTM2 / 4   | pd_per_ic2_fclk<br>From PRCM      |

### 21.2.3 I2C Pin List

The external signals (I2Cx\_SDA, I2Cx\_SCL) on the device use standard LVC MOS I/Os and may not meet full compliance with the I2C specifications for Fast-mode devices for slope control and input filtering (spike suppression) to improve the EMC behavior.

**Table 21-5. I2C Pin List**

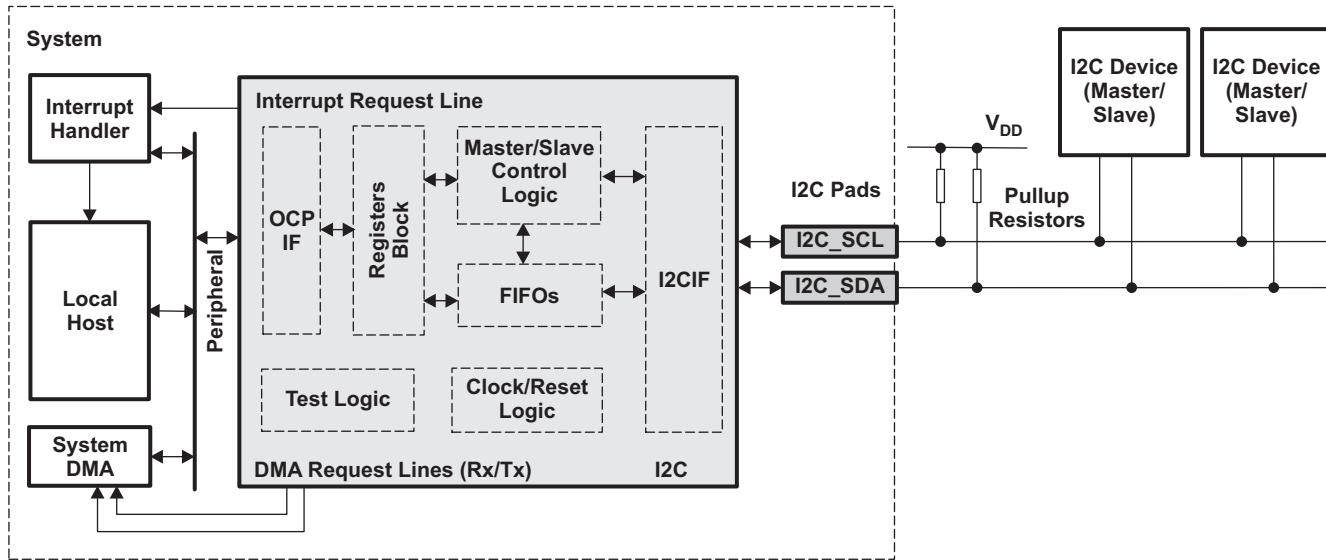
| Pin      | Type | Description                   |
|----------|------|-------------------------------|
| I2Cx_SCL | I/OD | I2C serial clock (open drain) |
| I2Cx_SDA | I/OD | I2C serial data (open drain)  |

## 21.3 Functional Description

### 21.3.1 Functional Block Diagram

Figure 21-3 shows an example of a system with multiple I2C compatible devices in which the I2C serial ports are all connected together for a two-way transfer from one device to other devices.

**Figure 21-3. I2C Functional Block Diagram**



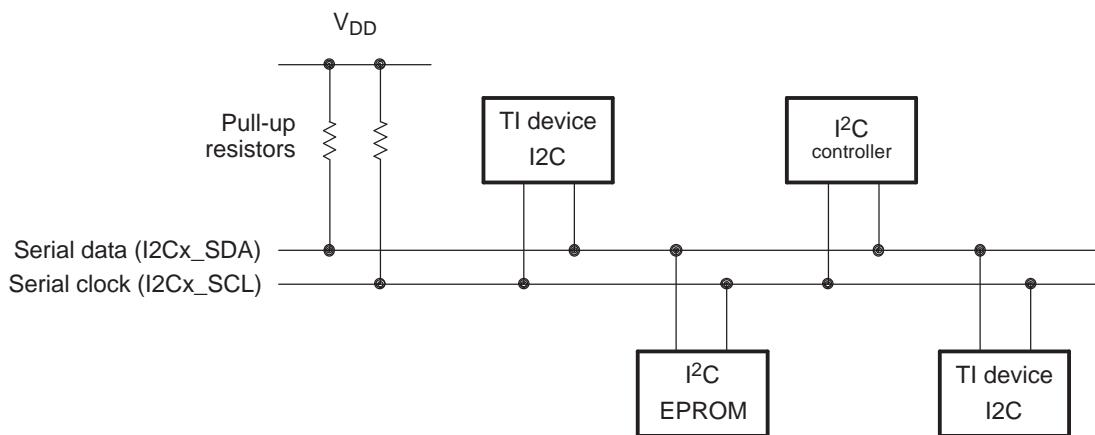
The I2C peripheral consists of the following primary blocks:

- A serial interface: one data pin (I2C\_SDA) and one clock pin (I2C\_SCL).
- Data registers to temporarily hold receive data and transmit data traveling between the I2C\_SDA pin and the CPU or the DMA controller.
- Control and status registers
- A peripheral data bus interface to enable the CPU and the DMA controller to access the I2C peripheral registers.
- A clock synchronizer to synchronize the I2C input clock (from the processor clock generator) and the clock on the I2C\_SCL pin, and to synchronize data transfers with masters of different clock speeds.
- A prescaler to divide down the input clock that is driven to the I2C peripheral
- A noise filter on each of the two pins, I2C\_SDA and I2C\_SCL
- An arbitrator to handle arbitration between the I2C peripheral (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- DMA event generation logic to send an interrupt to the CPU upon reception and data transmission of data.

### 21.3.2 I2C Master/Slave Controller Signals

Data is communicated to devices interfacing with the I2C via the serial data line (SDA) and the serial clock line (SCL). These two wires can carry information between a device and others connected to the I2C bus. Both SDA and SCL are bi-directional pins. They must be connected to a positive supply voltage via a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open drain to perform the required wired-AND function.

An example of multiple I2C modules that are connected for a two-way transfer from one device to other devices is shown in [Figure 21-4](#).

**Figure 21-4. Multiple I<sup>2</sup>C Modules Connected**

**Table 21-6. Signal Pads**

| Name                 | I <sup>2</sup> C Mode  |  |
|----------------------|------------------------|--|
|                      | Default Operating Mode | Description  |
| I <sup>2</sup> C_SCL | In/ Out                | I <sup>2</sup> C serial CLK line<br>Open-drain output buffer. Requires external pull-up resistor ( $R_p$ ).  |
| I <sup>2</sup> C_SDA | In/ Out                | I <sup>2</sup> C serial data line<br>Open-drain output buffer. Requires external pull-up resistor ( $R_p$ ). |

### 21.3.3 I<sup>2</sup>C Reset

The I<sup>2</sup>C module can be reset in the following three ways:

- A system reset (PIRSTNA = 0). A device reset causes the system reset. All registers are reset to power up reset values.
- A software reset by setting the SRST bit in the I<sup>2</sup>C\_SYSC register. This bit has exactly the same action on the module logic as the system bus reset. All registers are reset to power up reset values.
- The I<sup>2</sup>C\_EN bit in the I<sup>2</sup>C\_CON register can be used to hold the I<sup>2</sup>C module in reset. When the system bus reset is removed (PIRSTNA = 1), I<sup>2</sup>C\_EN = 0 keeps the functional part of I<sup>2</sup>C module in reset state and all configuration registers can be accessed. I<sup>2</sup>C\_EN = 0 does not reset the registers to power up reset values.

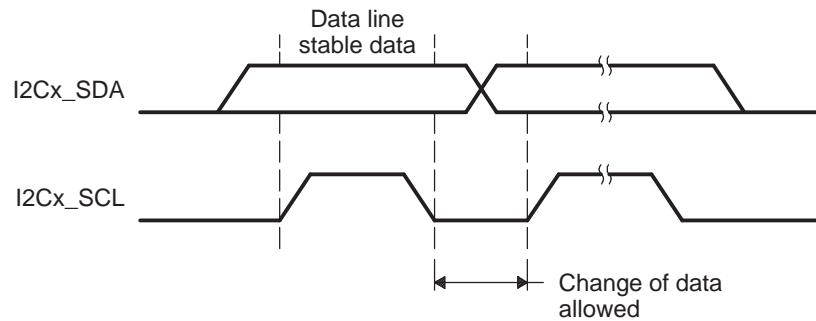
**Table 21-7. Reset State of I<sup>2</sup>C Signals**

| Pin | I/O/Z <sup>(1)</sup> | System Reset   | I <sup>2</sup> C Reset    |
|-----|----------------------|----------------|---------------------------|
|     |                      |                | (I <sup>2</sup> C_EN = 0) |
| SDA | I/O/Z                | High impedance | High impedance            |
| SCL | I/O/Z                | High impedance | High impedance            |

<sup>(1)</sup> I = Input, O = Output, Z = High impedance

### 21.3.4 Data Validity

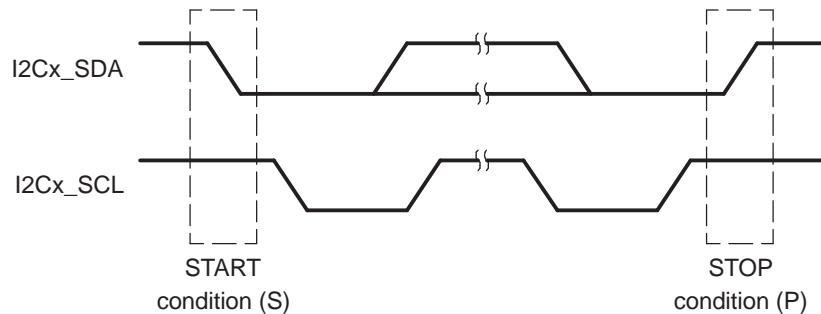
The data on the SDA line must be stable during the high period of the clock. The high and low states of the data line can only change when the clock signal on the SCL line is LOW.

**Figure 21-5. Bit Transfer on the I<sub>2</sub>C Bus**

### 21.3.5 START & STOP Conditions

- The I2C module generates START and STOP conditions when it is configured as a master.
- START condition is a high-to-low transition on the SDA line while SCL is high.
  - STOP condition is a low-to-high transition on the SDA line while SCL is high.
  - The bus is considered to be busy after the START condition ( $BB = 1$ ) and free after the STOP condition ( $BB = 0$ ).

**Figure 21-6. Start and Stop Condition Events**

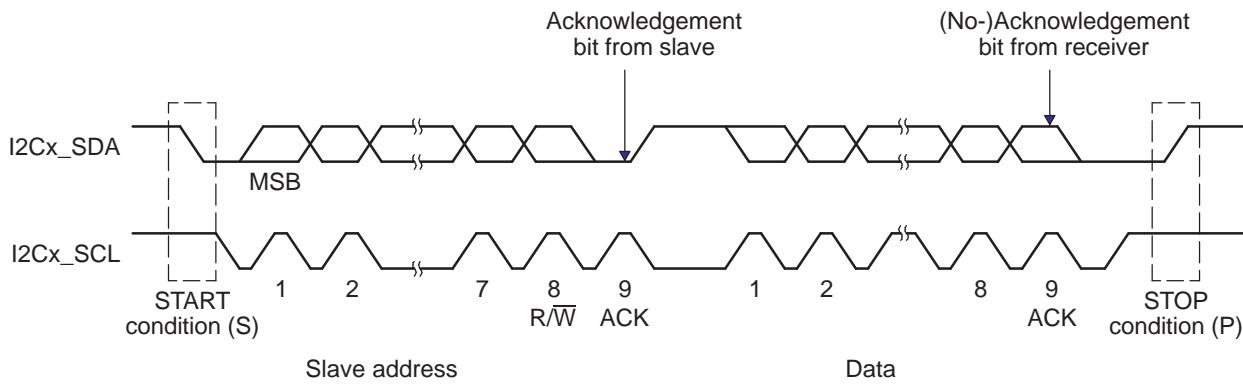


### 21.3.6 I2C Operation

#### 21.3.6.1 Serial Data Formats

The I2C controller operates in 8-bit word data format (byte write access supported for the last access). Each byte put on the SDA line is 8 bits long. The number of bytes that can be transmitted or received is restricted by the value programmed in the DCOUNT register. The data is transferred with the most significant bit (MSB) first. Each byte is followed by an acknowledge bit from the I2C module if it is in receiver mode.

**Figure 21-7. I2C Data Transfer**



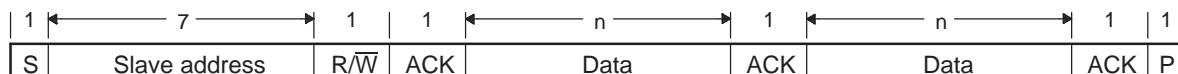
The I2C module supports two data formats, as shown in [Figure 21-8](#):

- 7-bit/10-bit addressing format
- 7-bit/10-bit addressing format with repeated start condition

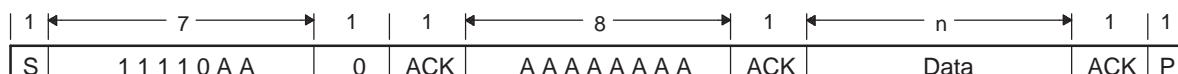
The first byte after a start condition (S) always consists of 8 bits. In the acknowledge mode, an extra bit dedicated for acknowledgment is inserted after each byte. In the addressing formats with 7-bit addresses, the first byte is composed of 7 MSB slave address bits and 1 LSB R/nW bit. In the addressing formats with 10-bit addresses, the first byte is composed of 7 MSB slave address bits, such as 11110XX, where XX is the two MSB of the 10-bit addresses, and 1 LSB R/nW bit, which is 0 in this case.

The least significant R/nW of the address byte indicates the direction of transmission of the following data bytes. If R/nW is 0, the master writes data into the selected slave; if it is 1, the master reads data out of the slave.

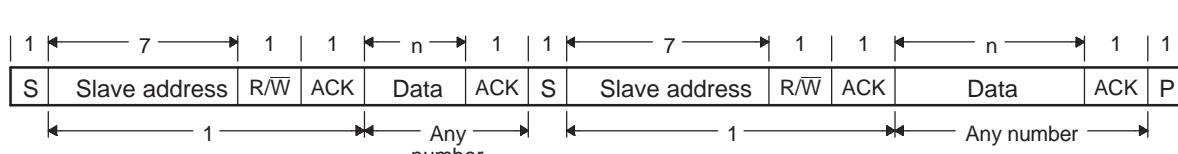
**Figure 21-8. I<sub>2</sub>C Data Transfer Formats**



7-Bit Addressing Format



10-Bit Addressing Format



7-Bit Addressing Format With Repeated START Condition

### 21.3.6.2 Master Transmitter

In this mode, data assembled in one of the previously described data formats is shifted out on the serial data line SDA in synch with the self-generated clock pulses on the serial clock line SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (XUDF) after a byte has been transmitted.

### 21.3.6.3 Master Receiver

This mode can only be entered from the master transmitter mode. With either of the address formats (Figure 21-8 (a), (b), and (c)), the master receiver is entered after the slave address byte and bit R/W<sub>\_</sub> has been transmitted, if R/W<sub>\_</sub> is high. Serial data bits received on bus line SDA are shifted in synch with the self-generated clock pulses on SCL. The clock pulses are inhibited and SCL held low when the intervention of the processor is required (ROVR) after a byte has been transmitted. At the end of a transfer, it generates the stop condition.

### 21.3.6.4 Slave Transmitter

This mode can only be entered from the slave receiver mode. With either of the address formats (Figure 21-8 (a), (b), and (c)), the slave transmitter is entered if the slave address byte is the same as its own address and bit R/W<sub>\_</sub> has been transmitted, if R/W<sub>\_</sub> is high. The slave transmitter shifts the serial data out on the data line SDA in synch with the clock pulses that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the CPU is required (XUDF).

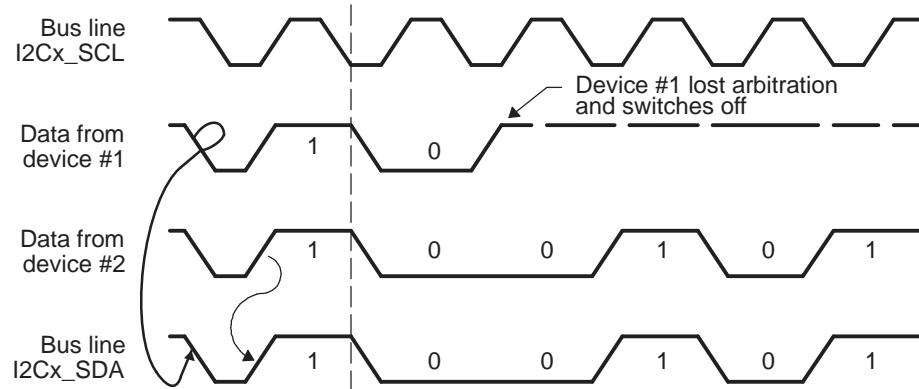
### 21.3.6.5 Slave Receiver

In this mode, serial data bits received on the bus line SDA are shifted-in in synch with the clock pulses on SCL that are generated by the master device. It does not generate the clock but it can hold clock line SCL low while intervention of the CPU is required (ROVR) following the reception of a byte.

### 21.3.7 Arbitration

If two or more master transmitters start a transmission on the same bus almost simultaneously, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial bus by the competing transmitters. When a transmitter senses that a high signal it has presented on the bus has been overruled by a low signal, it switches to the slave receiver mode, sets the arbitration lost (AL) flag, and generates the arbitration lost interrupt. [Figure 21-9](#) shows the arbitration procedure between two devices. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

**Figure 21-9. Arbitration Procedure Between Two Master Transmitters**



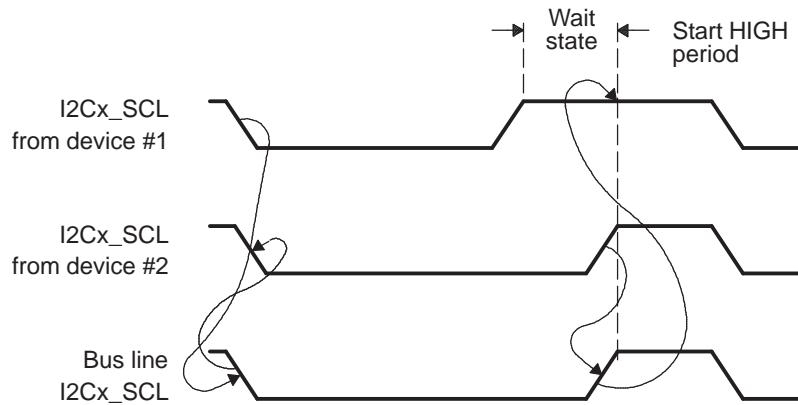
### 21.3.8 I<sup>2</sup>C Clock Generation and I<sup>2</sup>C Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more master devices and the clock must be synchronized so that the data output can be compared. The wired-AND property of the clock line means that a device that first generates a low period of the clock line overrules the other devices. At this high/low transition, the clock generators of the other devices are forced to start generation of their own low period. The clock line is then held low by the device with the longest low period, while the other devices that finish their low periods must wait for the clock line to be released before starting their high periods. A synchronized signal on the clock line is thus obtained, where the slowest device determines the length of the low period and the fastest the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the WAIT-state. In this way a slave can slow down a fast master and the slow device can create enough time to store a received byte or to prepare a byte to be transmitted (Clock Stretching). [Figure 21-10](#) illustrates the clock synchronization.

**Note:** If the SCL or SDA lines are stuck low, the Bus Clear operation is supported. If the clock line (SCL) is stuck low, the preferred procedure is to reset the bus using the hardware reset signal if your I<sup>2</sup>C devices have hardware reset inputs. If the I<sup>2</sup>C devices do not have hardware reset inputs, cycle power to the devices to activate the mandatory internal power-on reset (POR) circuit. If the data line (SDA) is stuck low, the master should send nine clock pulses. The device that held the bus low should release it sometime within those nine clocks. If not, use the hardware reset or cycle power to clear the bus.

**Figure 21-10. Synchronization of Two I<sub>2</sub>C Clock Generators**



### 21.3.9 Prescaler (SCLK/ICLK)

The I<sub>2</sub>C module is operated with a functional clock (SCLK) frequency that can be in a range of 12-100 MHz, according to I<sub>2</sub>C mode that must be used (an internal ~24 MHz clock (ICLK) is recommended in case of F/S operation mode e). Note that the frequency of the functional clock influences directly the I<sub>2</sub>C bus performance and timings.

The internal clock used for I<sub>2</sub>C logic - ICLK - is generated via the I<sub>2</sub>C prescaler block. The prescaler consists of a 4-bit register - I<sub>2</sub>C\_PSC, and is used to divide the system clock (SCLK) to obtain the internal required clock for the I<sub>2</sub>C module.

### 21.3.10 Noise Filter

The noise filter is used to suppress any noise that is 50 ns or less, in the case of F/S mode of operation. It is designed to suppress noise with one ICLK. The noise filter is always one ICLK cycle, regardless of the bus speed. For FS mode (prescaler = 4, ICLK = 24 MHz), the maximum width of the suppressed spikes is 41.6 ns. To ensure a correct filtering, the prescaler must be programmed accordingly.

### 21.3.11 I<sub>2</sub>C Interrupts

The I<sub>2</sub>C module generates 12 types of interrupt: addressed as slave, bus free (stop condition detected), access error, start condition, arbitration-lost, noacknowledge, general call, registers-ready-for-access, receive and transmit data, receive and transmit draining. These 12 interrupts are accompanied with 12 interrupt masks and flags defined in the I<sub>2</sub>C\_IRQENABLE\_SET and respectively I<sub>2</sub>C\_IRQSTATUS\_RAW registers. Note that all these 12 interrupt events are sharing the same hardware interrupt line.

- Addressed As Slave interrupt (AAS) is generated to inform the Local Host that an external master addressed the module as a slave. When this interrupt occurs, the CPU can check the I<sub>2</sub>C\_ACTOA status register to check which of the 4 own addresses was used by the external master to access the module.
- Bus Free interrupt (BF) is generated to inform the Local Host that the I<sub>2</sub>C bus became free (when a Stop Condition is detected on the bus) and the module can initiate his own I<sub>2</sub>C transaction.
- Start Condition interrupt (STC) is generated after the module being in idle mode have detected (synchronously or asynchronously) a possible Start Condition on the bus (signalized with WakeUp).
- Access Error interrupt (AERR) is generated if a Data read access is performed while RX FIFO is empty or a Data write access is performed while TX FIFO is full.
- Arbitration lost interrupt (AL) is generated when the I<sub>2</sub>C arbitration procedure is lost.
- No-acknowledge interrupt (NACK) is generated when the master I<sub>2</sub>C does not receive acknowledge from the receiver.
- General call interrupt (GC) is generated when the device detects the address of all zeros (8 bits).
- Registers-ready-for-access interrupt (ARDY) is generated by the I<sub>2</sub>C when the previously programmed address, data, and command have been performed and the status bits have been updated. This

interrupt is used to let the CPU know that the I2C registers are ready for access.

- Receive interrupt/status (RRDY) is generated when there is received data ready to be read by the CPU from the I2C\_DATA register (see the FIFO Management subsection for a complete description of required conditions for interrupt generation). The CPU can alternatively poll this bit to read the received data from the I2C\_DATA register.
- Transmit interrupt/status (XRDY) is generated when the CPU needs to put more data in the I2C\_DATA register after the transmitted data has been shifted out on the SDA pin (see the FIFO Management subsection for a complete description of required conditions for interrupt generation). The CPU can alternatively poll this bit to write the next transmitted data into the I2C\_DATA register.
- Receive draining interrupt (RDR) is generated when the transfer length is not a multiple of threshold value, to inform the CPU that it can read the amount of data left to be transferred and to enable the draining mechanism. (see the Draining Feature subsection for additional details).
- Transmit draining interrupt (XDR) is generated when the transfer length is not a multiple of threshold value, to inform the CPU that it can read the amount of data left to be written and to enable the draining mechanism. (see the Draining Feature subsection for additional details).

When the interrupt signal is activated, the Local Host must read the I2C\_IRQSTATUS\_RAW register to define the type of the interrupt, process the request, and then write into these registers the correct value to clear the interrupt flag.

### **21.3.12 DMA Events**

The I2C module can generate two DMA requests events, read (I2C\_DMA\_RX) and write (I2C\_DMA\_TX) that can be used by the DMA controller to synchronously read received data from the I2C\_DATA or write transmitted data to the I2C\_DATA register. The DMA read and write requests are generated in a similar manner as RRDY and XRDY, respectively.

The I2C DMA request signals (I2C\_DMA\_TX and I2C\_DMA\_RX) are activated according to the FIFO Management subsection.

### **21.3.13 Interrupt and DMA Events**

I2C has two DMA channels (Tx and Rx).

I2C has one interrupt line for all the interrupt requests.

For the event and interrupt numbers, see *AM335x ARM Cortex-A8 Microprocessors (MPUs)* (literature number [SPRS717](#)).

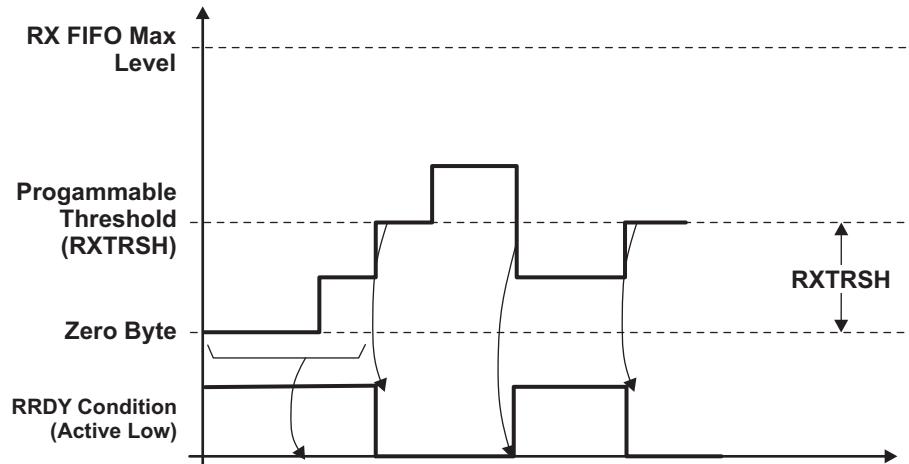
### **21.3.14 FIFO Management**

The I2C module implements two internal 32-bytes FIFOs with dual clock for RX and TX modes. The depth of the FIFOs can be configured at integration via a generic parameter which will also be reflected in I2C\_IRQSTATUS\_RAW.FIFODEPTH register.

#### **21.3.14.1 FIFO Interrupt Mode Operation**

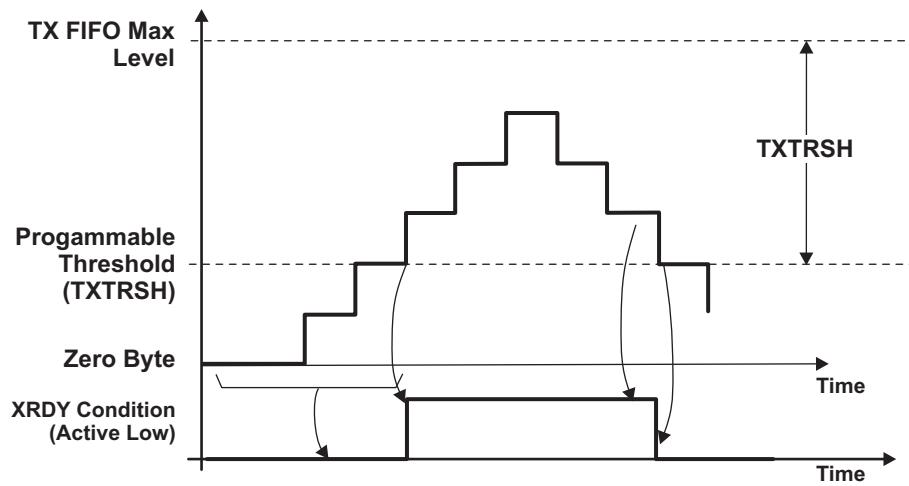
In FIFO interrupt mode (relevant interrupts enabled via I2C\_IRQENABLE\_SET register), the processor is informed of the status of the receiver and transmitter by an interrupt signal. These interrupts are raised when receive/transmit FIFO threshold (defined by I2C\_BUF.TXTRSH or I2C\_BUF.RXTRSH) are reached; the interrupt signals instruct the Local Host to transfer data to the destination (from the I2C module in receive mode and/or from any source to the I2C FIFO in transmit mode).

[Figure 21-11](#) and [Figure 21-12](#), respectively, illustrate receive and transmit operations from FIFO management point of view.

**Figure 21-11. Receive FIFO Interrupt Request Generation**


Note that in [Figure 21-11](#), the RRDY Condition illustrates that the condition for generating a RRDY interrupt is achieved. The interrupt request is generated when this signal is active, and it can be cleared only by the CPU by writing a 1 in the corresponding interrupt flag. If the condition is still present after clearing the previous interrupt, another interrupt request will be generated.

In receive mode, RRDY interrupt is not generated until the FIFO reaches its receive threshold. Once low, the interrupt can only be de-asserted when the Local Host has handled enough bytes to make the FIFO level below threshold. For each interrupt, the Local Host can be configured to read an amount of bytes equal with the value of the RX FIFO threshold + 1.

**Figure 21-12. Transmit FIFO Interrupt Request Generation**


Note that in the figure above, the XRDY Condition illustrates that the condition for generating a XRDY interrupt is achieved. The interrupt request is generated when this condition is achieved (when TX FIFO is empty, or the TX FIFO threshold is not reached and there are still data bytes to be transferred in the TX FIFO), and it can be cleared only by the CPU by writing a 1 in the corresponding interrupt flag after transmitting the configured number of bytes. If the condition is still present after clearing the previous interrupt, another interrupt request will be generated.

Note that in interrupt mode, the module offers two options for the CPU application to handle the interrupts:

- When detecting an interrupt request (XRDY or RRDY type), the CPU can write/read one data byte to/from the FIFO and then clear the interrupt. The module will not reassert the interrupt until the

interrupt condition is not met.

- When detecting an interrupt request (XRDY or RRDY type), the CPU can be programmed to write/read the amount of data bytes specified by the corresponding FIFO threshold (I2C\_BUF.TXTRSH + 1 or I2C\_BUF.RXTRSH + 1). In this case, the interrupt condition will be cleared and the next interrupt will be asserted again when the XRDY or RRDY condition will be again met.

If the second interrupt serving approach is used, an additional mechanism (draining feature) is implemented for the case when the transfer length is not a multiple of FIFO threshold (see the Draining Feature subsection).

In slave TX mode, the draining feature cannot be used, since the transfer length is not known at the configuration time, and the external master can end the transfer at any point by not acknowledging one data byte.

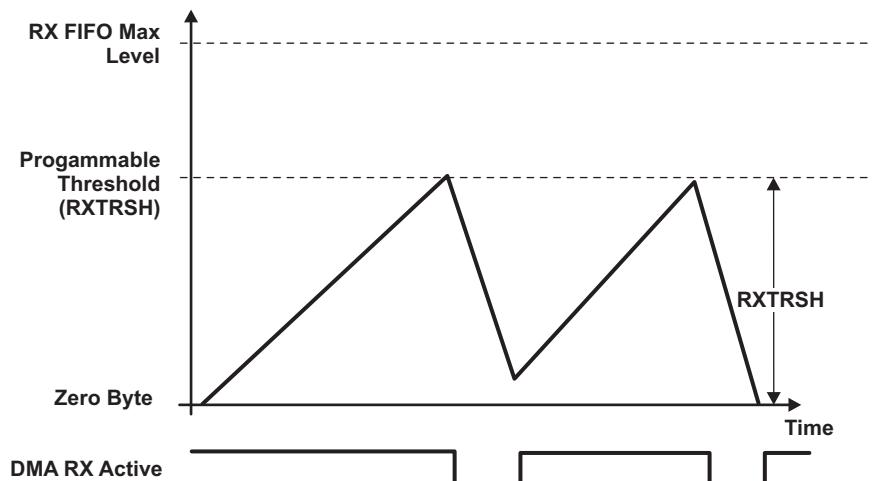
#### 21.3.14.2 FIFO Polling Mode Operation

In FIFO polled mode (I2C\_IRQENABLE\_SET.XRDY\_IE and I2C\_IRQENABLE\_SET.RRDY\_IE disabled and DMA disabled), the status of the module (receiver or transmitter) can be checked by polling the XRDY and RRDY status registers (I2C\_IRQSTATUS\_RAW) (RDR and XDR can also be polled if draining feature must be used). The XRDY and RRDY flags are accurately reflecting the interrupt conditions mentioned in Interrupt Mode. This mode is an alternative to the FIFO interrupt mode of operation, where the status of the receiver and transmitter is automatically known by means of interrupts sent to the CPU.

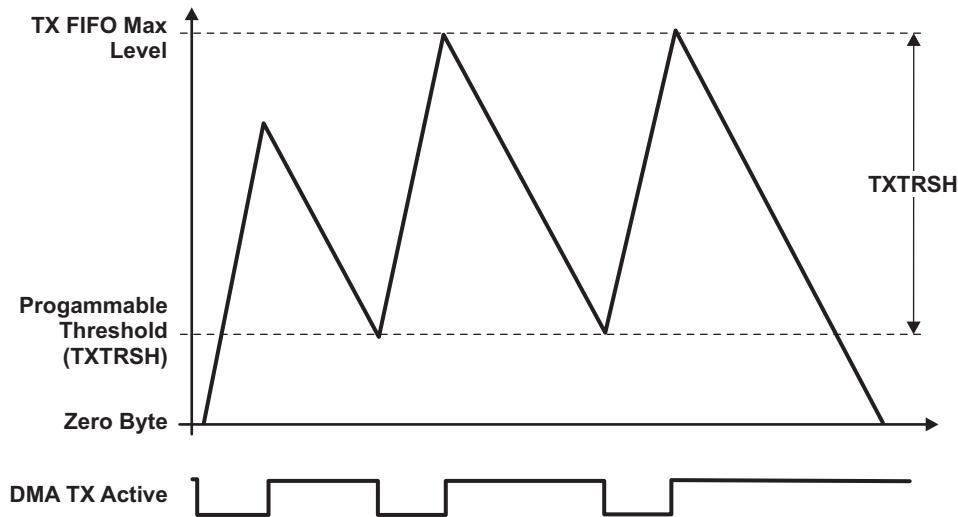
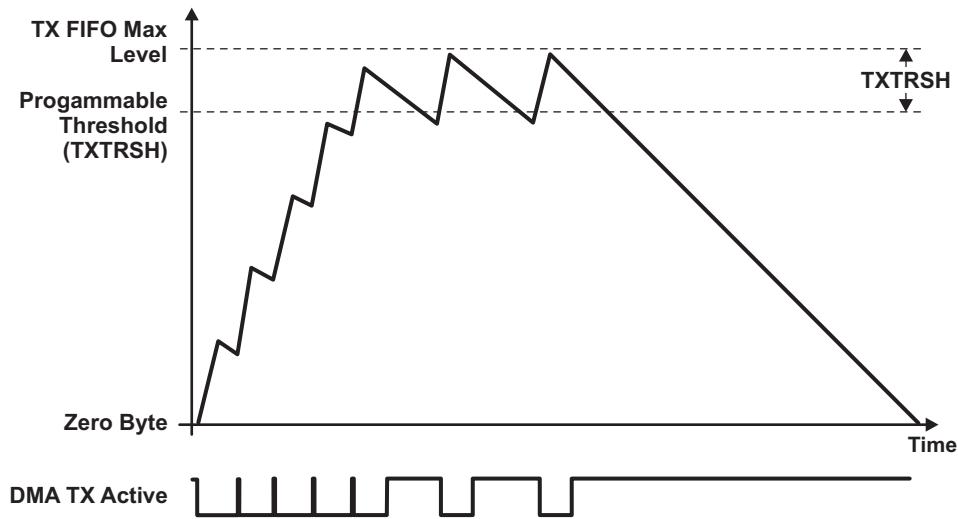
#### 21.3.14.3 FIFO DMA Mode Operation

In receive mode, a DMA request is generated as soon as the receive FIFO exceeds its threshold level defined in the threshold level register (I2C\_BUF.RXTRSH +1). This request should be de-asserted when the number of bytes defined by the threshold level has been read by the DMA, by setting the I2C\_DMARXENABLE\_CLR.DMARX\_ENABLE\_CLEAR field.

**Figure 21-13. Receive FIFO DMA Request Generation**



In transmit mode, a DMA request is automatically asserted when the transmit FIFO is empty. This request should be de-asserted when the number of bytes defined by the number in the threshold register (I2C\_BUF.TXTHRS+1) has been written in the FIFO by the DMA, by setting the I2C\_DMATXENABLE\_CLR.DMATX\_ENABLE\_CLEAR field. If an insufficient number of characters are written, then the DMA request will remain active. [Figure 21-14](#) and [Figure 21-15](#) illustrate DMA TX transfers with different values for TXTRSH.

**Figure 21-14. Transmit FIFO DMA Request Generation (High Threshold)**

**Figure 21-15. Transmit FIFO DMA Request Generation (Low Threshold)**


Note that also in DMA mode it is possible to have a transfer whose length is not multiple of the configured FIFO threshold. In this case, the DMA draining feature is also used for transferring the additional bytes of the transfer (see the Draining Feature subsection for additional details).

According to the desired operation mode, the programmer must set the FIFO thresholds according to the following table (note that only the interface/OCP side thresholds can be programmed; the I2C side thresholds are default equals to 1). Note that the thresholds must be set consistent with the DMA channel length.

In I2C Slave RX Mode, the Local Host can program the RX threshold with the desired value, and use the FIFO draining feature at the end of the I2C transfer to extract from the FIFO the remaining bytes if the threshold is not reached (see the Draining Feature subsection for additional details).

Note that in I2C Slave TX Mode, the TX FIFO threshold should be set to 1 (I2C\_BUF.TXTRSH=0, default value), since the length of the transfer may not be known at configuration time. In this way, the interrupt (or accordingly, DMA) requests will be generated for each byte requested by the remote I2C master to be transferred over the I2C bus. This configuration will prevent the I2C core to request additional data from the CPU or from the DMA controller (using IRQ or DMA), data that will eventually not be extracted from the FIFO by the external master (which can use not acknowledge at any time to end the transfer). If the TX threshold is not set to 1, the module will generate an interrupt or assert a DMA only when the external master requests a byte and the FIFO is empty. However, in this case the TX FIFO will require to be cleared at the end of the transfer.

The I2C module offers the possibility to the user to clear the RX or TX FIFO. This is achieved through I2C\_BUF.RXFIFO\_CLR and I2C\_BUF.TXFIFO\_CLR registers, which act like software reset for the FIFOs. In DMA mode, these bits will also reset the DMA state machines.

The FIFO clearing feature can be used when the module is configured as a transmitter, the external receiver responds with a NACK in the middle of the transfer, and there is still data in TX FIFO waiting to be transferred.

On the Functional (I2C) domain, the thresholds can always be considered equal to 1. This means that the I2C Core can start transferring data on the I2C bus whenever it has data in the FIFOs (FIFO is not empty).

#### 21.3.14.4 Draining Feature

The Draining Feature is implemented by the I2C core for handling the end of the transfers whose length is not a multiple of FIFO threshold value, and offers the possibility to transfer the remaining amount of bytes (since the threshold is not reached).

Note that this feature prevents the CPU or the DMA controller to attempt more FIFO accesses than necessary (for example, to generate at the end of a transfer a DMA RX request having in the FIFO less bytes than the configured DMA transfer length). Otherwise, an Access Error interrupt will be generated (see I2C\_IRQSTATUS\_RAW.AERR interrupt).

The Draining mechanism will generate an interrupt (I2C\_IRQSTATUS\_RAW.RDR or I2C\_IRQSTATUS\_RAW.XDR) at the end of the transfer informing the CPU that it needs to check the amount of data left to be transferred (I2C\_BUFSTAT.TXSTAT or RXSTAT) and to enable the Draining Feature of the DMA controller if DMA mode is enabled (by re-configuring the DMA transfer length according to this value), or perform only the required number of data accesses, if DMA mode is disabled.

In receiving mode (master or slave), if the RX FIFO threshold is not reached but the transfer was ended on the I2C bus and there is still data left in the FIFO (less than the threshold), the receive draining interrupt (I2C\_IRQSTATUS\_RAW.RDR) will be asserted to inform the local host that it can read the amount of data in the FIFO (I2C\_BUFSTAT.RXSTAT). The CPU will perform a number of data read accesses equal with RXSTAT value (if interrupt or polling mode) or re-configure the DMA controller with the required value in order to drain the FIFO.

In master transmit mode, if the TX FIFO threshold is not reached but the amount of data remaining to be written in the FIFO is less than TXTRSH, the transmit draining interrupt (I2C\_IRQSTATUS\_RAW.XDR) will be asserted to inform the local host that it can read the amount of data remained to be written in the TX FIFO (I2C\_BUFSTAT.TXSTAT). The CPU will need to write the required number of data bytes (specified by TXSTAT value) or re-configure the DMA controller with the required value in order to transfer the last bytes to the FIFO.

Note that in master mode, the CPU can alternatively skip the checking of TXSTAT and RXSTAT values since it can obtain internally this information (by computing DATACOUNT modulo TX/RXTHRESH).

The draining feature is disabled by default, and it can be enabled using I2C\_IRQENABLE\_SET.XDR\_IE or I2C\_IRQENABLE\_SET.RDR\_IE registers (default disabled) only for the transfers with length not equal with the threshold value.

#### 21.3.15 How to Program I2C

##### 21.3.15.1 Module Configuration Before Enabling the Module

1. Program the prescaler to obtain an approximately 12-MHz I2C module clock (I2C\_PSC = x; this value is to be calculated and is dependent on the System clock frequency).
2. Program the I2C clock to obtain 100 Kbps or 400 Kbps (SCLL = x and SCLH = x; these values are to be calculated and are dependent on the System clock frequency).
3. Configure its own address (I2C\_OA = x) - only in case of I2C operating mode (F/S mode).
4. Take the I2C module out of reset (I2C\_CON:I2C\_EN = 1).

##### 21.3.15.2 Initialization Procedure

1. Configure the I2C mode register (I2C\_CON) bits.
2. Enable interrupt masks (I2C\_IRQENABLE\_SET), if using interrupt for transmit/receive data.
3. Enable the DMA (I2C\_BUF and I2C\_DMA/RX/TX/ENABLE\_SET) and program the DMA controller) - only in case of I2C operating mode (F/S mode), if using DMA for transmit/receive data.

##### 21.3.15.3 Configure Slave Address and DATA Counter Registers

In master mode, configure the slave address (I2C\_SA = x) and the number of byte associated with the transfer (I2C\_CNT = x).

#### 21.3.15.4 Initiate a Transfer

Poll the bus busy (BB) bit in the I2C status register (I2C\_IRQSTATUS\_RAW). If it is cleared to 0 (bus not busy), configure START/STOP (I2C\_CON: STT / I2C\_CON: STP condition to initiate a transfer) - only in case of I2C operating mode (F/S mode).

#### 21.3.15.5 Receive Data

Poll the receive data ready interrupt flag bit (RRDY) in the I2C status register (I2C\_IRQSTATUS\_RAW), use the RRDY interrupt (I2C\_IRQENABLE\_SET.RRDY\_IE set) or use the DMA RX (I2C\_BUF.RDMA\_EN set together with I2C\_DMARXENABLE\_SET) to read the received data in the data receive register (I2C\_DATA). Use draining feature (I2C\_IRQSTATUS\_RAW.RDR enabled by I2C\_IRQENABLE\_SET.RDR\_IE) if the transfer length is not equal with FIFO threshold.

#### 21.3.15.6 Transmit Data

Poll the transmit data ready interrupt flag bit (XRDY) in the I2C status register (I2C\_IRQSTATUS\_RAW), use the XRDY interrupt (I2C\_IRQENABLE\_SET.XRDY\_IE set) or use the DMA TX (I2C\_BUF.XDMA\_EN set together with I2C\_DMATXENABLE\_SET) to write data into the data transmit register (I2C\_DATA). Use draining feature (I2C\_IRQSTATUS\_RAW.XDR enabled by I2C\_IRQENABLE\_SET.XDR\_IE) if the transfer length is not equal with FIFO threshold.

## 21.4 I2C Registers

---

**NOTE:** All bits defined as reserved must be written by software with 0s, for preserving future compatibility. When read, any reserved bit returns 0. Also, note that it is good software practice to use complete mask patterns for setting or testing individually bit fields within a register.

---

### 21.4.1 I2C Registers

Table 21-8 lists the memory-mapped registers for the I2C. All register offset addresses not listed in Table 21-8 should be considered as reserved locations and the register contents should not be modified.

**Table 21-8. I2C REGISTERS**

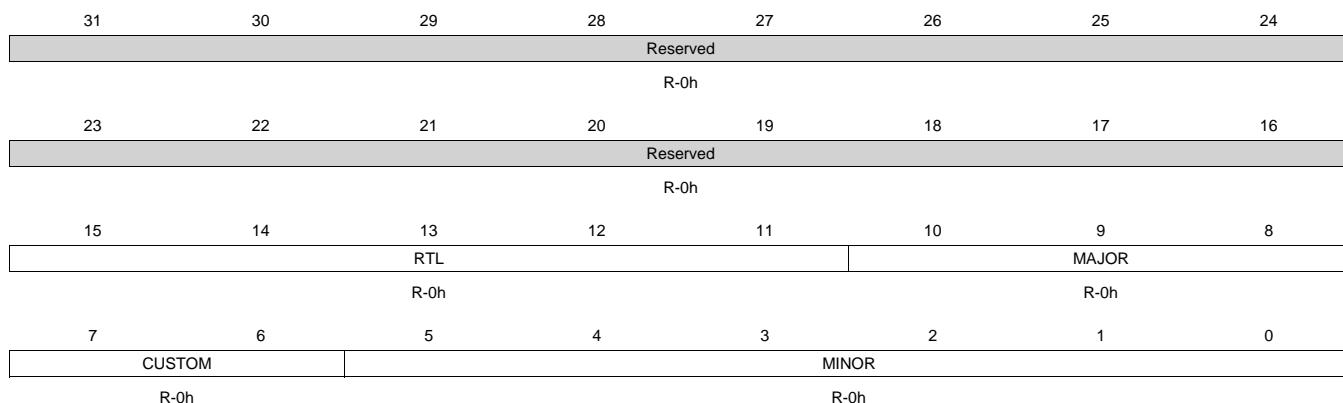
| Offset | Acronym             | Register Name | Section           |
|--------|---------------------|---------------|-------------------|
| 00h    | I2C_REVNB_LO        |               | Section 21.4.1.1  |
| 04h    | I2C_REVNB_HI        |               | Section 21.4.1.2  |
| 10h    | I2C_SYSC            |               | Section 21.4.1.3  |
| 24h    | I2C_IRQSTATUS_RAW   |               | Section 21.4.1.4  |
| 28h    | I2C_IRQSTATUS       |               | Section 21.4.1.5  |
| 2Ch    | I2C_IRQENABLE_SET   |               | Section 21.4.1.6  |
| 30h    | I2C_IRQENABLE_CLR   |               | Section 21.4.1.7  |
| 34h    | I2C_WE              |               | Section 21.4.1.8  |
| 38h    | I2C_DMARXENABLE_SET |               | Section 21.4.1.9  |
| 3Ch    | I2C_DMATXENABLE_SET |               | Section 21.4.1.10 |
| 40h    | I2C_DMARXENABLE_CLR |               | Section 21.4.1.11 |
| 44h    | I2C_DMATXENABLE_CLR |               | Section 21.4.1.12 |
| 48h    | I2C_DMARXWAKE_EN    |               | Section 21.4.1.13 |
| 4Ch    | I2C_DMATXWAKE_EN    |               | Section 21.4.1.14 |
| 90h    | I2C_SYSS            |               | Section 21.4.1.15 |
| 94h    | I2C_BUF             |               | Section 21.4.1.16 |
| 98h    | I2C_CNT             |               | Section 21.4.1.17 |
| 9Ch    | I2C_DATA            |               | Section 21.4.1.18 |
| A4h    | I2C_CON             |               | Section 21.4.1.19 |
| A8h    | I2C_OA              |               | Section 21.4.1.20 |
| ACh    | I2C_SA              |               | Section 21.4.1.21 |
| B0h    | I2C_PSC             |               | Section 21.4.1.22 |
| B4h    | I2C_SCLL            |               | Section 21.4.1.23 |
| B8h    | I2C_SCLH            |               | Section 21.4.1.24 |
| BCh    | I2C_SYSTEST         |               | Section 21.4.1.25 |
| C0h    | I2C_BUFWSTAT        |               | Section 21.4.1.26 |
| C4h    | I2C_OA1             |               | Section 21.4.1.27 |
| C8h    | I2C_OA2             |               | Section 21.4.1.28 |
| CCh    | I2C_OA3             |               | Section 21.4.1.29 |
| D0h    | I2C_ACTOA           |               | Section 21.4.1.30 |
| D4h    | I2C_SBLOCK          |               | Section 21.4.1.31 |

### 21.4.1.1 I2C\_REVNB\_LO Register (offset = 00h) [reset = 0h]

I2C\_REVNB\_LO is shown in [Figure 21-16](#) and described in [Table 21-9](#).

This read-only register contains the hard-coded revision number of the module. A write to this register has no effect. I2C controller with interrupt using interrupt vector register (I2C\_IV) is revision 1.x. I2C controller with interrupt using status register bits (I2C\_IRQSTATUS\_RAW) is revision 2.x.

**Figure 21-16. I2C\_REVNB\_LO Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-9. I2C\_REVNB\_LO Register Field Descriptions**

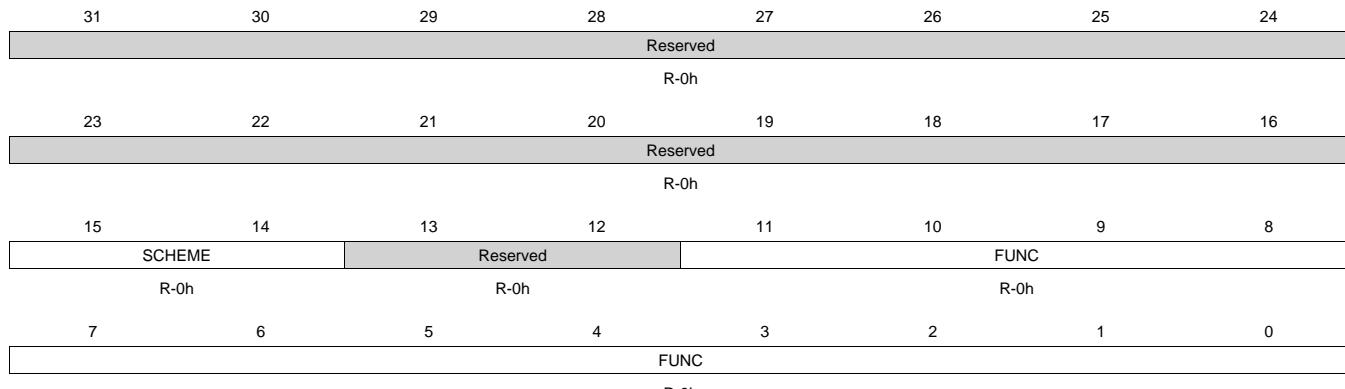
| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15-11 | RTL      | R    | 0h    | RTL version.   |
| 10-8  | MAJOR    | R    | 0h    | Major Revision.<br>This field changes when there is a major feature change.<br>This field does not change due to bug fix, or minor feature change.             |
| 7-6   | CUSTOM   | R    | 0h    | Indicates a special version for a particular device.<br>Consequence of use may avoid use of standard Chip Support Library (CSL) / Drivers.<br>0 if non-custom. |
| 5-0   | MINOR    | R    | 0h    | Minor Revision This field changes when features are scaled up or down.<br>This field does not change due to bug fix, or major feature change.                  |

### 21.4.1.2 I2C\_REVNB\_HI Register (offset = 04h) [reset = 0h]

I2C\_REVNB\_HI is shown in [Figure 21-17](#) and described in [Table 21-10](#).

A reset has no effect on the value returned.

**Figure 21-17. I2C\_REVNB\_HI Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-10. I2C\_REVNB\_HI Register Field Descriptions**

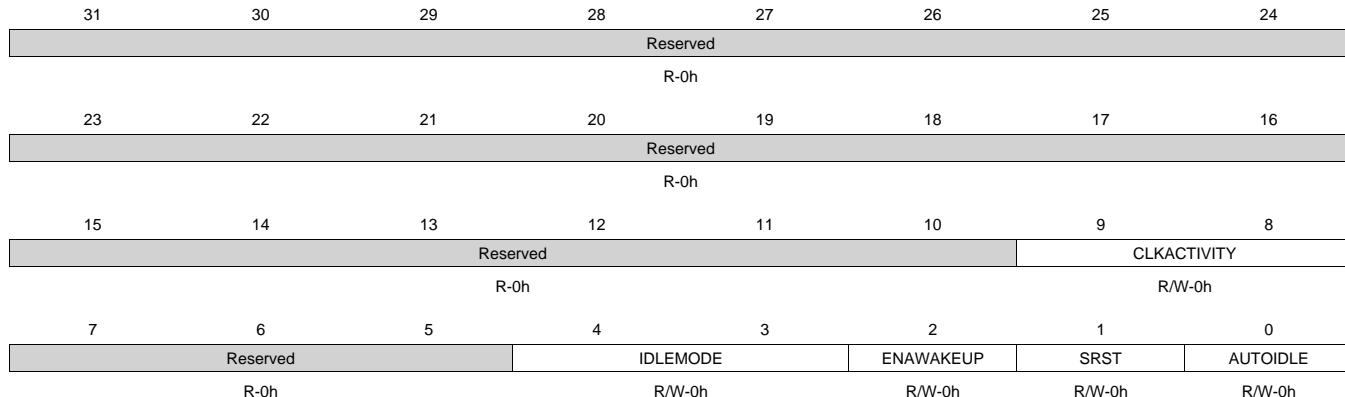
| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15-14 | SCHEME   | R    | 0h    | Used to distinguish between old Scheme and current.<br>Spare bit to encode future schemes. |
| 13-12 | Reserved | R    | 0h    |  |
| 11-0  | FUNC     | R    | 0h    | Function: Indicates a software compatible module family                                    |

### 21.4.1.3 I2C\_SYSC Register (offset = 10h) [reset = 0h]

I2C\_SYSC is shown in [Figure 21-18](#) and described in [Table 21-11](#).

This register allows controlling various parameters of the peripheral interface.

**Figure 21-18. I2C\_SYSC Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-11. I2C\_SYSC Register Field Descriptions**

| Bit   | Field       | Type | Reset | Description  |
|-------|-------------|------|-------|--|
| 31-10 | Reserved    | R    | 0h    |  |
| 9-8   | CLKACTIVITY | R/W  | 0h    | <p>Clock Activity selection bits.<br/>           Those bits (one bit for each clock signal present on the boundary of the module) are set to 1 to disable external clock gating mechanism in Idle Mode.<br/>           Values after reset are low (for both 2 bits).<br/>           Note: If the System (functional) Clock is cut-off, the module will assert a WakeUp event when it asynchronously detects a Start Condition on the I2C Bus.<br/>           Note that in this case the first transfer will not be taken into account by the module (NACK will be detected by the external master).<br/>           0x0 = Both clocks can be cut off<br/>           0x1 = Only Interface/OCP clock must be kept active; system clock can be cut off<br/>           0x2 = Only system clock must be kept active; Interface/OCP clock can be cut off<br/>           0x3 = Both clocks must be kept active</p> |
| 7-5   | Reserved    | R    | 0h    |  |
| 4-3   | IDLEMODE    | R/W  | 0h    | <p>Idle Mode selection bits.<br/>           These two bits are used to select one of the idle mode operation mechanisms.<br/>           Value after reset is 00 (Force Idle).<br/>           0x1 = No Idle mode<br/>           0x2 = Smart Idle mode<br/>           0x3 = Reserved</p>   |
| 2     | ENAWAKEUP   | R/W  | 0h    | <p>Enable Wakeup control bit.<br/>           When this bit is set to 1, the module enables its own wakeup mechanism.<br/>           Value after reset is low.<br/>           0x0 = Wakeup mechanism is disabled<br/>           0x1 = Wakeup mechanism is enabled</p>   |

**Table 21-11. I2C\_SYSC Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description   |
|-----|----------|------|-------|---|
| 1   | SRST     | R/W  | 0h    | <p>SoftReset bit.<br/> When this bit is set to 1, entire module is reset as for the hardware reset.<br/> This bit is automatically cleared to 0 by the core and it is only reset by the hardware reset.<br/> During reads, it always returns 0.<br/> Value after reset is low.<br/> 0x0 = Normal mode<br/> 0x1 = The module is reset</p>  |
| 0   | AUTOIDLE | R/W  | 0h    | <p>Autoidle bit.<br/> When this bit is set to 1, the module activates its own idle mode mechanism.<br/> By evaluating its internal state, the module can decide to gate part of his internal clock tree in order to improve the overall power consumption.<br/> Value after reset is high.<br/> 0x0 = Auto Idle mechanism is disabled<br/> 0x1 = Auto Idle mechanism is enabled</p> |

#### 21.4.1.4 I2C\_IRQSTATUS\_RAW Register (offset = 24h) [reset = 0h]

I2C\_IRQSTATUS\_RAW is shown in [Figure 21-19](#) and described in [Table 21-12](#).

This register provides core status information for interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit will set it to 1, that is, trigger the IRQ (mostly for debug). Writing a 0 will have no effect, that is, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line.

**Figure 21-19. I2C\_IRQSTATUS\_RAW Register**

|          |        |        |        |        |        |        |        |
|----------|--------|--------|--------|--------|--------|--------|--------|
| 31       | 30     | 29     | 28     | 27     | 26     | 25     | 24     |
| Reserved |        |        |        |        |        |        |        |
| R-0h     |        |        |        |        |        |        |        |
| 23       | 22     | 21     | 20     | 19     | 18     | 17     | 16     |
| Reserved |        |        |        |        |        |        |        |
| R-0h     |        |        |        |        |        |        |        |
| 15       | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| Reserved | XDR    | RDR    | BB     | ROVR   | XUDF   | AAS    | BF     |
| R-0h     | R/W-0h | R/W-0h | R-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| AERR     | STC    | GC     | XRDY   | RRDY   | ARDY   | NACK   | AL     |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-15 | Reserved | R    | 0h    |  |
| 14    | XDR      | R/W  | 0h    | <p>Transmit draining IRQ status.<br/>I2C Master Transmit mode only.</p> <p>This read/clear only bit is set to 1 when the module is configured as a master transmitter, the TX FIFO level is below the configured threshold (TXTRSH) and the amount of data still to be transferred is less than TXTRSH.</p> <p>When this bit is set to 1 by the core, CPU must read the I2C_BUFSTAT.TXSTAT register in order to check the amount of data that need to be written in the TX FIFO.</p> <p>Then, according to the mode set (DMA or interrupt), the CPU can enable the DMA draining feature of the DMA controller with the number of data bytes to be transferred (I2C_BUFSTAT.TXSTAT), or generate write data accesses according to this value (IRQ mode).</p> <p>The interrupt needs to be cleared after the DMA controller was reconfigured (if DMA mode enabled), or before generating data accesses to the FIFO (if IRQ mode enabled).</p> <p>If the corresponding interrupt was enabled, an interrupt is signaled to the local host.</p> <p>The CPU can also poll this bit.</p> <p>For more details about TDR generation, refer to the FIFO Management subsection.</p> <p>The CPU can only clear this bit by writing a 1 into this register.</p> <p>A write 0 has no effect.</p> <p>Value after reset is low.</p> <p>0x0 = Transmit draining inactive</p> <p>0x1 = Transmit draining enabled</p> |

Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions (continued)

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 13  | RDR   | R/W  | 0h    | <p>Receive draining IRQ status.<br/>I2C Receive mode only.</p> <p>This read/clear only bit is set to 1 when the module is configured as a receiver, a stop condition was received on the bus and the RX FIFO level is below the configured threshold (RXTRSH). When this bit is set to 1 by the core, CPU must read the I2C_BUFSTAT.RXSTAT register in order to check the amount of data left to be transferred from the FIFO.</p> <p>Then, according to the mode set (DMA or interrupt), the CPU needs to enable the draining feature of the DMA controller with the number of data bytes to be transferred (I2C_BUFSTAT.RXSTAT), or generate read data accesses according to this value (IRQ mode). The interrupt needs to be cleared after the DMA controller was reconfigured (if DMA mode enabled), or before generating data accesses to the FIFO (if IRQ mode enabled).</p> <p>If the corresponding interrupt was enabled, an interrupt is signaled to the local host.</p> <p>The CPU can also poll this bit.</p> <p>For more details about RDR generation, refer to the FIFO Management subsection.</p> <p>The CPU can only clear this bit by writing a 1 into this register. A write 0 has no effect.</p> <p>Value after reset is low.</p> <p>0x0 = Receive draining inactive<br/>0x1 = Receive draining enabled</p> |
| 12  | BB    | R    | 0h    | <p>This read-only bit indicates the state of the serial bus. In slave mode, on reception of a start condition, the device sets BB to 1.</p> <p>BB is cleared to 0 after reception of a stop condition.</p> <p>In master mode, the software controls BB.</p> <p>To start a transmission with a start condition, MST, TRX, and STT must be set to 1 in the I2C_CON register.</p> <p>To end a transmission with a stop condition, STP must be set to 1 in the I2C_CON register.</p> <p>When BB = 1 and STT = 1, a restart condition is generated.</p> <p>Value after reset is low.</p> <p>0x0 = Bus is free<br/>0x1 = Bus is occupied</p>  |
| 11  | ROVR  | R/W  | 0h    | <p>Receive overrun status.</p> <p>Writing into this bit has no effect.</p> <p>I2C receive mode only.</p> <p>This read-only bit indicates whether the receiver has experienced overrun.</p> <p>Overrun occurs when the shift register is full and the receive FIFO is full.</p> <p>An overrun condition does not result in a data loss the peripheral is just holding the bus (low on SCL) and prevents other bytes from being received.</p> <p>ROVR is set to 1 when the I2C has recognized an overrun.</p> <p>ROVR is clear when reading I2C_DATA register, or when resetting the I2C (I2C_CON:I2C_EN = 0).</p> <p>Value after reset is low.</p> <p>0x0 = Normal operation<br/>0x1 = Receiver overrun</p>  |

**Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 10  | XUDF  | R/W  | 0h    | <p>Transmit underflow status.<br/>Writing into this bit has no effect.<br/>I2C transmit mode only.</p> <p>This read-only bit indicates whether the transmitter has experienced underflow.</p> <p>In master transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (DCOUNT 0).</p> <p>In slave transmit mode, underflow occurs when the shift register is empty, the transmit FIFO is empty, and there are still some bytes to transmit (read request from external I2C master).</p> <p>XUDF is set to 1 when the I2C has recognized an underflow.<br/>The core holds the line till the underflow cause has disappeared.</p> <p>XUDF is clear when writing I2C_DATA register or resetting the I2C (I2C_CON:I2C_EN = 0).</p> <p>Value after reset is low.</p> <p>0x0 = Normal operation<br/>0x1 = Transmit underflow</p> |
| 9   | AAS   | R/W  | 0h    | <p>Address recognized as slave IRQ status.<br/>I2C mode only.</p> <p>This read only bit is set to 1 by the device when it has recognized its own slave address (or one of the alternative own addresses), or an address of all zeros (8 bits).</p> <p>When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.</p> <p>This bit can be cleared in 2 ways: One way is if the interrupt was enabled, it will be cleared by writing 1 into this register (writing 0 has no effect).</p> <p>The other way is if the interrupt was not enabled, the AAS bit is reset to 0 by restart or stop.</p> <p>Value after reset is low.</p> <p>0x0 = No action<br/>0x1 = Address recognized</p>  |
| 8   | BF    | R/W  | 0h    | <p>I2C mode only.</p> <p>This read only bit is set to 1 by the device when the I2C bus became free (after a transfer is ended on the bus stop condition detected).<br/>This interrupt informs the Local Host that it can initiate its own I2C transfer on the bus.</p> <p>When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.</p> <p>The CPU can only clear this bit by writing a 1 into this register.<br/>Writing 0 has no effect.</p> <p>Value after reset is low.</p> <p>0x0 = No action<br/>0x1 = Bus Free</p>  |
| 7   | AERR  | R/W  | 0h    | <p>Access Error IRQ status.<br/>I2C mode only.</p> <p>This read/clear only bit is set to 1 by the device if an Interface/OCP write access is performed to I2C_DATA while the TX FIFO is full or if an Interface/OCP read access is performed to the I2C_DATA while the RX FIFO is empty.</p> <p>Note that, when the RX FIFO is empty, a read access will return to the previous read data value.</p> <p>When the TX FIFO is full, a write access is ignored.<br/>In both events, the FIFO pointers will not be updated.</p> <p>When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.</p> <p>The CPU can only clear this bit by writing a 1 into this register.<br/>Writing 0 has no effect.</p> <p>Value after reset is low.</p> <p>0x0 = No action<br/>0x1 = Access Error</p>   |

Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions (continued)

| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 6   | STC   | R/W  | 0h    | <p>Start Condition IRQ status.<br/>I2C mode only.</p> <p>This read/clear only bit is set to 1 by the device if previously the module was in idle mode and a start condition was asynchronously detected on the I2C Bus and signalized with an Wakeup (if the I2C_SYSC.ClockActivity allows the system clock to be cut-off). When the Active Mode will be restored and the interrupt generated, this bit will indicate the reason of the wakeup.</p> <p>Note</p> <p>1: The corresponding interrupt for this bit should be enabled only if the module was configured to allow the possibility of cutting-off the system clock while in Idle State (I2C_SYSC.ClockActivity = 00 or 01).</p> <p>2: The first transfer (corresponding to the detected start condition) will be lost (not taken into account by the module) and it will be used only for generating the WakeUp enable for restoring the Active Mode of the module.</p> <p>On the I2C line, the external master which generated the transfer will detect this behavior as a not acknowledge to the address phase and will possibly restart the transfer.</p> <p>The CPU can only clear this bit by writing a 1 into this register.</p> <p>Writing 0 has no effect.</p> <p>Value after reset is low.</p> <p>0x0 = No action</p> <p>0x1 = Start Condition detected</p> |
| 5   | GC    | R/W  | 0h    | <p>General call IRQ status.<br/>Set to '1' by core when General call address detected and interrupt signaled to MPUSS.</p> <p>Write '1' to clear.</p> <p>I2C mode only.</p> <p>This read/clear only bit is set to 1 by the device if it detects the address of all zeros (8 bits) (general call).</p> <p>When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.</p> <p>The CPU can only clear this bit by writing a 1 into this register.</p> <p>Writing 0 has no effect.</p> <p>Note: When this bit is set to 1, AAS also reads as 1.</p> <p>Value after reset is low.</p> <p>0x0 = No general call detected</p> <p>0x1 = General call address detected</p>   |

**Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 4   | XRDY  | R/W  | 0h    | <p>Transmit data ready IRQ status.<br/> Set to '1' by core when transmitter and when new data is requested.<br/> When set to '1' by core, an interrupt is signaled to MPUSS.<br/> Write '1' to clear.</p> <p>Transmit mode only (I2C mode).<br/> This read/clear only bit (XRDY) is set to 1 when the I2C peripheral is a master or slave transmitter, the CPU needs to send data through the I2C bus, and the module (transmitter) requires new data to be served.</p> <p>Note that a master transmitter requests new data if the FIFO TX level is below the threshold (TXTRSH) and the required amount of data remained to be transmitted (I2C_BUFSTAT.TXSTAT) is greater than the threshold.</p> <p>A slave transmitter requests new data when the FIFO TX level is below the threshold (if TXTRSH &gt; 1), or anytime there is a read request from external master (for each acknowledge received from the master), if TXTRSH = 1.</p> <p>When this bit is set to 1 by the core, an interrupt is signaled to the local host if the interrupt was enabled.</p> <p>The CPU can also poll this bit (refer to the FIFO Management subsection for details about XRDY generation).</p> <p>The CPU can only clear this bit by writing a 1 into this register.</p> <p>Writing 0 has no effect.</p> <p>Note: If the DMA transmit mode is enabled (I2C_BUF.XDMA_EN is set, together with I2C_DMATXENABLE_SET), this bit is forced to 0 and no interrupt will be generated instead, a DMA TX request to the main DMA controller of the system is generated.</p> <p>Value after reset is low.</p> <p>0x0 = Transmission ongoing<br/> 0x1 = Transmit data ready</p> |
| 3   | RRDY  | R/W  | 0h    | <p>Receive mode only (I2C mode).<br/> This read/clear only RRDY is set to 1 when the RX FIFO level is above the configured threshold (RXTRSH).<br/> When this bit is set to 1 by the core, CPU is able to read new data from the I2C_DATA register.</p> <p>If the corresponding interrupt was enabled, an interrupt is signaled to the local host.</p> <p>The CPU to read the received data in I2C_DATA register can also poll this bit (refer to the FIFO Management subsection for details about RRDY generation).</p> <p>The CPU can only clear this bit by writing a 1 into this register.</p> <p>A write 0 has no effect.</p> <p>If the DMA receive mode is enabled (I2C_BUF.RDMA_EN is set, together with I2C_DMARXENABLE_SET), this bit is forced to 0 and no interrupt will be generated instead a DMA RX request to the main DMA controller of the system is generated.</p> <p>Value after reset is low.</p> <p>0x0 = Receive FIFO threshold not reached<br/> 0x1 = Receive data ready for read (RX FIFO threshold reached)</p>   |

**Table 21-12. I2C\_IRQSTATUS\_RAW Register Field Descriptions (continued)**

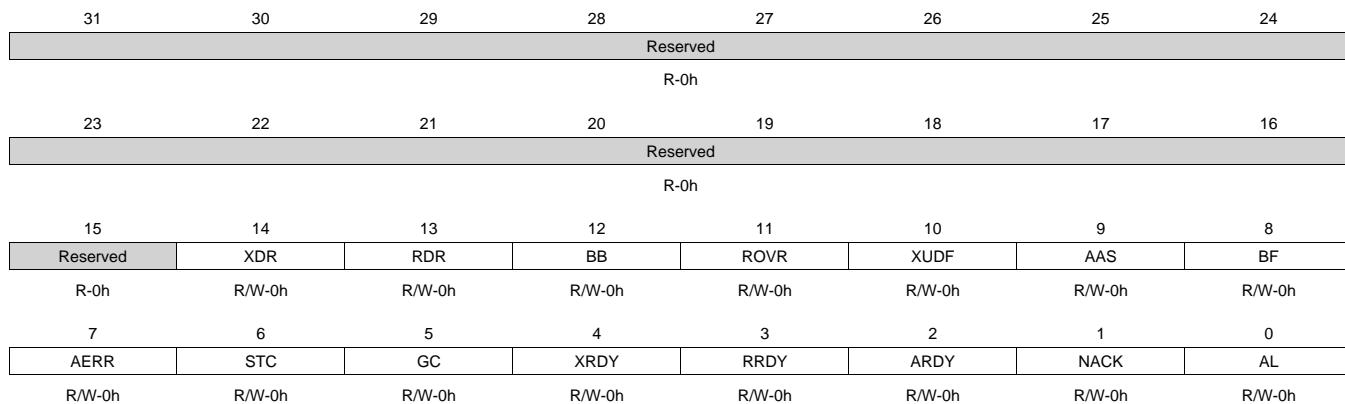
| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 2   | ARDY  | R/W  | 0h    | <p>I2C mode only.<br/> This read/clear only bit, when set to 1, indicates that the previously programmed data and command (receive or transmit, master or slave) has been performed and status bit has been updated.<br/> The CPU uses this flag to let it know that the I2C registers are ready to be accessed again.<br/> The CPU can only clear this bit by writing a 1 into this register.<br/> A write 0 has no effect.<br/> Mode: I2C Master transmit, Others: STP = 1, ARDY Set Condition: DCOUNT = 0.<br/> Mode: I2C Master receive, Others: STP = 1, ARDY Set Condition: DCOUNT = 0 and receiver FIFO empty Mode: I2C Master transmit, Others: STP = 0, ARDY Set Condition: DCOUNT passed 0 Mode: I2C Master receive, Others: STP = 0, ARDY Set Condition: DCOUNT passed 0 and receiver FIFO empty Mode: I2C Master transmit, Others: n/a, ARDY Set Condition: Stop or restart condition received from master Mode: I2C Slave receive, Others: n/a, ARDY Set Condition: Stop or restart condition and receiver FIFO empty Value after reset is low.<br/> 0x0 = No action<br/> 0x1 = Access ready</p> |
| 1   | NACK  | R/W  | 0h    | <p>No acknowledgment IRQ status.<br/> Bit is set when No Acknowledge has been received, an interrupt is signaled to MPUSS.<br/> Write '1' to clear this bit.<br/> I2C mode only.<br/> The read/clear only No Acknowledge flag bit is set when the hardware detects No Acknowledge has been received.<br/> When a NACK event occurs on the bus, this bit is set to 1, the core automatically ends the transfer and clears the MST/STP bits in the I2C_CON register and the I2C becomes a slave.<br/> Clearing the FIFOs from remaining data might be required.<br/> The CPU can only clear this bit by writing a 1 into this register.<br/> Writing 0 has no effect.<br/> Value after reset is low.<br/> 0x0 = Normal operation<br/> 0x1 = Not Acknowledge detected</p>  |
| 0   | AL    | R/W  | 0h    | <p>Arbitration lost IRQ status.<br/> This bit is automatically set by the hardware when it loses the Arbitration in master transmit mode, an interrupt is signaled to MPUSS.<br/> During reads, it always returns 0.<br/> I2C mode only.<br/> The read/clear only Arbitration Lost flag bit is set to 1 when the device (configured in master mode) detects it has lost an arbitration (in Address Phase).<br/> This happens when two or more masters initiate a transfer on the I2C bus almost simultaneously or when the I2C attempts to start a transfer while BB (bus busy) is 1.<br/> When this is set to 1 due to arbitration lost, the core automatically clears the MST/STP bits in the I2C_CON register and the I2C becomes a slave receiver.<br/> The CPU can only clear this bit by writing a 1 to this register.<br/> Writing 0 has no effect.<br/> Value after reset is low.<br/> 0x0 = Normal operation<br/> 0x1 = Arbitration lost detected</p>  |

### 21.4.1.5 I2C\_IRQSTATUS Register (offset = 28h) [reset = 0h]

I2C\_IRQSTATUS is shown in [Figure 21-20](#) and described in [Table 21-13](#).

This register provides core status information for interrupt handling, showing all active and enabled events and masking the others. The fields are read-write. Writing a 1 to a bit will clear it to 0, that is, clear the IRQ. Writing a 0 will have no effect, that is, the register value will not be modified. Only enabled, active events will trigger an actual interrupt request on the IRQ output line. For all the internal fields of the I2C\_IRQSTATUS register, the descriptions given in the I2C\_IRQSTATUS\_RAW subsection are valid.

**Figure 21-20. I2C\_IRQSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-13. I2C\_IRQSTATUS Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-15 | Reserved | R    | 0h    |  |
| 14    | XDR      | R/W  | 0h    | Transmit draining IRQ enabled status.<br>0x0 = Transmit draining inactive<br>0x1 = Transmit draining enabled                     |
| 13    | RDR      | R/W  | 0h    | Receive draining IRQ enabled status.<br>0x0 = Receive draining inactive<br>0x1 = Receive draining enabled                        |
| 12    | BB       | R/W  | 0h    | Bus busy enabled status.<br>Writing into this bit has no effect.<br>0x0 = Bus is free<br>0x1 = Bus is occupied                   |
| 11    | ROVR     | R/W  | 0h    | Receive overrun enabled status.<br>Writing into this bit has no effect.<br>0x0 = Normal operation<br>0x1 = Receiver overrun      |
| 10    | XUDF     | R/W  | 0h    | Transmit underflow enabled status.<br>Writing into this bit has no effect.<br>0x0 = Normal operation<br>0x1 = Transmit underflow |
| 9     | AAS      | R/W  | 0h    | Address recognized as slave IRQ enabled status.<br>0x0 = No action<br>0x1 = Address recognized                                   |
| 8     | BF       | R/W  | 0h    | Bus Free IRQ enabled status.<br>0x0 = No action<br>0x1 = Bus free  |

**Table 21-13. I2C\_IRQSTATUS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 7   | AERR  | R/W  | 0h    | Access Error IRQ enabled status.<br>0x0 = No action<br>0x1 = Access error  |
| 6   | STC   | R/W  | 0h    | Start Condition IRQ enabled status.<br>0x0 = No action<br>0x1 = Start condition detected   |
| 5   | GC    | R/W  | 0h    | General call IRQ enabled status.<br>Set to '1' by core when General call address detected and interrupt signaled to MPUSS.<br>Write '1' to clear.<br>0x0 = No general call detected<br>0x1 = General call address detected   |
| 4   | XRDY  | R/W  | 0h    | Transmit data ready IRQ enabled status.<br>Set to '1' by core when transmitter and when new data is requested.<br>When set to '1' by core, an interrupt is signaled to MPUSS.<br>Write '1' to clear.<br>0x0 = Transmission ongoing<br>0x1 = Transmit data ready                    |
| 3   | RRDY  | R/W  | 0h    | Receive data ready IRQ enabled status.<br>Set to '1' by core when receiver mode, a new data is able to be read.<br>When set to '1' by core, an interrupt is signaled to MPUSS.<br>Write '1' to clear.<br>0x0 = No data available<br>0x1 = Receive data available                   |
| 2   | ARDY  | R/W  | 0h    | Register access ready IRQ enabled status.<br>When set to '1' it indicates that previous access has been performed and registers are ready to be accessed again.<br>An interrupt is signaled to MPUSS.<br>Write '1' to clear.<br>0x0 = Module busy<br>0x1 = Access ready            |
| 1   | NACK  | R/W  | 0h    | No acknowledgment IRQ enabled status.<br>Bit is set when No Acknowledge has been received, an interrupt is signaled to MPUSS.<br>Write '1' to clear this bit.<br>0x0 = Normal operation<br>0x1 = Not Acknowledge detected  |
| 0   | AL    | R/W  | 0h    | Arbitration lost IRQ enabled status.<br>This bit is automatically set by the hardware when it loses the Arbitration in master transmit mode, an interrupt is signaled to MPUSS.<br>During reads, it always returns 0.<br>0x0 = Normal operation<br>0x1 = Arbitration lost detected |

### 21.4.1.6 I2C\_IRQENABLE\_SET Register (offset = 2Ch) [reset = 0h]

I2C\_IRQENABLE\_SET is shown in [Figure 21-21](#) and described in [Table 21-14](#).

All 1-bit fields enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit will enable the field. Writing a 0 will have no effect, that is, the register value will not be modified. For all the internal fields of the I2C\_IRQENABLE\_SET register, the descriptions given in the I2C\_IRQSTATUS\_RAW subsection are valid.

**Figure 21-21. I2C\_IRQENABLE\_SET Register**

|          |        |        |          |         |         |         |        |
|----------|--------|--------|----------|---------|---------|---------|--------|
| 31       | 30     | 29     | 28       | 27      | 26      | 25      | 24     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 23       | 22     | 21     | 20       | 19      | 18      | 17      | 16     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 15       | 14     | 13     | 12       | 11      | 10      | 9       | 8      |
| Reserved | XDR_IE | RDR_IE | Reserved | ROVR    | XUDF    | AAS_IE  | BF_IE  |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |
| 7        | 6      | 5      | 4        | 3       | 2       | 1       | 0      |
| AERR_IE  | STC_IE | GC_IE  | XRDY_IE  | RRDY_IE | ARDY_IE | NACK_IE | AL_IE  |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h   | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-14. I2C\_IRQENABLE\_SET Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-15 | Reserved | R    | 0h    |   |
| 14    | XDR_IE   | R/W  | 0h    | Transmit draining interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[XDR].<br>0x0 = Transmit draining interrupt disabled<br>0x1 = Transmit draining interrupt enabled    |
| 13    | RDR_IE   | R/W  | 0h    | Receive draining interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[RDR].<br>0x0 = Receive draining interrupt disabled<br>0x1 = Receive draining interrupt enabled       |
| 12    | Reserved | R    | 0h    |   |
| 11    | ROVR     | R/W  | 0h    | Receive overrun enable set.<br>0x0 = Receive overrun interrupt disabled<br>0x1 = Receive draining interrupt enabled   |
| 10    | XUDF     | R/W  | 0h    | Transmit underflow enable set.<br>0x0 = Transmit underflow interrupt disabled<br>0x1 = Transmit underflow interrupt enabled   |
| 9     | AAS_IE   | R/W  | 0h    | Addressed as slave interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AAS].<br>0x0 = Addressed as slave interrupt disabled<br>0x1 = Addressed as slave interrupt enabled |
| 8     | BF_IE    | R/W  | 0h    | Bus free interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[BF].<br>0x0 = Bus free interrupt disabled<br>0x1 = Bus free interrupt enabled                                |
| 7     | AERR_IE  | R/W  | 0h    | Access error interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AERR].<br>0x0 = Access error interrupt disabled<br>0x1 = Access error interrupt enabled                  |

**Table 21-14. I2C\_IRQENABLE\_SET Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 6   | STC_IE  | R/W  | 0h    | Start condition interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[STC].<br>0x0 = Start condition interrupt disabled<br>0x1 = Start condition interrupt enabled                    |
| 5   | GC_IE   | R/W  | 0h    | General call interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[GC].<br>0x0 = General call interrupt disabled<br>0x1 = General call interrupt enabled                              |
| 4   | XRDY_IE | R/W  | 0h    | Transmit data ready interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[XRDY].<br>0x0 = Transmit data ready interrupt disabled<br>0x1 = Transmit data ready interrupt enabled       |
| 3   | RRDY_IE | R/W  | 0h    | Receive data ready interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[RRDY].<br>0x0 = Receive data ready interrupt disabled<br>0x1 = Receive data ready interrupt enabled          |
| 2   | ARDY_IE | R/W  | 0h    | Register access ready interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[ARDY].<br>0x0 = Register access ready interrupt disabled<br>0x1 = Register access ready interrupt enabled |
| 1   | NACK_IE | R/W  | 0h    | No acknowledgment interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[NACK].<br>0x0 = Not Acknowledge interrupt disabled<br>0x1 = Not Acknowledge interrupt enabled                 |
| 0   | AL_IE   | R/W  | 0h    | Arbitration lost interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AL].<br>0x0 = Arbitration lost interrupt disabled<br>0x1 = Arbitration lost interrupt enabled                  |

### 21.4.1.7 I2C\_IRQENABLE\_CLR Register (offset = 30h) [reset = 0h]

I2C\_IRQENABLE\_CLR is shown in [Figure 21-22](#) and described in [Table 21-15](#).

All 1-bit fields clear a specific interrupt event. Writing a 1 to a bit will disable the interrupt field. Writing a 0 will have no effect, that is, the register value will not be modified. For all the internal fields of the I2C\_IRQENABLE\_CLR register, the descriptions given in the I2C\_IRQSTATUS\_RAW subsection are valid.

**Figure 21-22. I2C\_IRQENABLE\_CLR Register**

|          |        |        |          |         |         |         |        |
|----------|--------|--------|----------|---------|---------|---------|--------|
| 31       | 30     | 29     | 28       | 27      | 26      | 25      | 24     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 23       | 22     | 21     | 20       | 19      | 18      | 17      | 16     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 15       | 14     | 13     | 12       | 11      | 10      | 9       | 8      |
| Reserved | XDR_IE | RDR_IE | Reserved | ROVR    | XUDF    | AAS_IE  | BF_IE  |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |
| 7        | 6      | 5      | 4        | 3       | 2       | 1       | 0      |
| AERR_IE  | STC_IE | GC_IE  | XRDY_IE  | RRDY_IE | ARDY_IE | NACK_IE | AL_IE  |
| R/W-0h   | R/W-0h | R/W-0h | R/W-0h   | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-15. I2C\_IRQENABLE\_CLR Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-15 | Reserved | R    | 0h    |   |
| 14    | XDR_IE   | R/W  | 0h    | Transmit draining interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[XDR].<br>0x0 = Transmit draining interrupt disabled<br>0x1 = Transmit draining interrupt enabled    |
| 13    | RDR_IE   | R/W  | 0h    | Receive draining interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[RDR].<br>0x0 = Receive draining interrupt disabled<br>0x1 = Receive draining interrupt enabled       |
| 12    | Reserved | R    | 0h    |   |
| 11    | ROVR     | R/W  | 0h    | Receive overrun enable clear.<br>0x0 = Receive overrun interrupt disabled<br>0x1 = Receive draining interrupt enabled   |
| 10    | XUDF     | R/W  | 0h    | Transmit underflow enable clear.<br>0x0 = Transmit underflow interrupt disabled<br>0x1 = Transmit underflow interrupt enabled   |
| 9     | AAS_IE   | R/W  | 0h    | Addressed as slave interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AAS].<br>0x0 = Addressed as slave interrupt disabled<br>0x1 = Addressed as slave interrupt enabled |
| 8     | BF_IE    | R/W  | 0h    | Bus Free interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[BF].<br>0x0 = Bus free interrupt disabled<br>0x1 = Bus free interrupt enabled                                |
| 7     | AERR_IE  | R/W  | 0h    | Access error interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AERR].<br>0x0 = Access error interrupt disabled<br>0x1 = Access error interrupt enabled                  |

**Table 21-15. I2C\_IRQENABLE\_CLR Register Field Descriptions (continued)**

| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 6   | STC_IE  | R/W  | 0h    | Start condition interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[STC].<br>0x0 = Start condition interrupt disabled<br>0x1 = Start condition interrupt enabled                    |
| 5   | GC_IE   | R/W  | 0h    | General call interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[GC].<br>0x0 = General call interrupt disabled<br>0x1 = General call interrupt enabled                              |
| 4   | XRDY_IE | R/W  | 0h    | Transmit data ready interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[XRDY].<br>0x0 = Transmit data ready interrupt disabled<br>0x1 = Transmit data ready interrupt enabled       |
| 3   | RRDY_IE | R/W  | 0h    | Receive data ready interrupt enable set.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[RRDY]<br>0x0 = Receive data ready interrupt disabled<br>0x1 = Receive data ready interrupt enabled             |
| 2   | ARDY_IE | R/W  | 0h    | Register access ready interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[ARDY].<br>0x0 = Register access ready interrupt disabled<br>0x1 = Register access ready interrupt enabled |
| 1   | NACK_IE | R/W  | 0h    | No acknowledgment interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[NACK].<br>0x0 = Not Acknowledge interrupt disabled<br>0x1 = Not Acknowledge interrupt enabled                 |
| 0   | AL_IE   | R/W  | 0h    | Arbitration lost interrupt enable clear.<br>Mask or unmask the interrupt signaled by bit in I2C_STAT[AL].<br>0x0 = Arbitration lost interrupt disabled<br>0x1 = Arbitration lost interrupt enabled                  |

### 21.4.1.8 I2C\_WE Register (offset = 34h) [reset = 0h]

I2C\_WE is shown in [Figure 21-23](#) and described in [Table 21-16](#).

Every 1-bit field in the I2C\_WE register enables a specific (synchronous) IRQ request source to generate an asynchronous wakeup (on the appropriate wakeup line). When a bit location is set to 1 by the local host, a wakeup is signaled to the local host if the corresponding event is captured by the core of the I2C controller. Value after reset is low (all bits). There is no need for an Access Error WakeUp event, since this event occurs only when the module is in Active Mode (for Interface/OCP accesses to FIFO) and is signaled by an interrupt. With the exception of Start Condition WakeUp, which is asynchronously detected when the Functional clock is turned-off, all the other WakeUp events require the Functional (System) clock to be enabled.

**Figure 21-23. I2C\_WE Register**

|          |        |        |          |         |         |         |        |
|----------|--------|--------|----------|---------|---------|---------|--------|
| 31       | 30     | 29     | 28       | 27      | 26      | 25      | 24     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 23       | 22     | 21     | 20       | 19      | 18      | 17      | 16     |
| Reserved |        |        |          |         |         |         |        |
| R-0h     |        |        |          |         |         |         |        |
| 15       | 14     | 13     | 12       | 11      | 10      | 9       | 8      |
| Reserved | XDR_WE | RDR_WE | Reserved | ROVR_WE | XUDF_WE | AAS_WE  | BF_WE  |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |
| 7        | 6      | 5      | 4        | 3       | 2       | 1       | 0      |
| Reserved | STC_WE | GC_WE  | Reserved | DRDY_WE | ARDY_WE | NACK_WE | AL_WE  |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h  | R/W-0h  | R/W-0h  | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-16. I2C\_WE Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-15 | Reserved | R    | 0h    |   |
| 14    | XDR_WE   | R/W  | 0h    | Transmit draining wakeup enable.<br>This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode, the TX FIFO level is below the threshold and the amount of data left to be transferred is less than TXTRSH value.<br>This allows for the module to inform the CPU that it can check the amount of data to be written to the FIFO.<br>0x0 = Transmit draining wakeup disabled<br>0x1 = Transmit draining wakeup enabled                           |
| 13    | RDR_WE   | R/W  | 0h    | Receive draining wakeup enable.<br>This read/write bit is used to enable or disable wakeup signal generation when I2C is in idle mode, configured as a receiver, and it has detected a stop condition on the bus but the RX FIFO threshold is not reached (but the FIFO is not empty).<br>This allows for the module to inform the CPU that it can check the amount of data to be transferred from the FIFO.<br>0x0 = Receive draining wakeup disabled<br>0x1 = Receive draining wakeup enabled |
| 12    | Reserved | R    | 0h    |   |
| 11    | ROVR_WE  | R/W  | 0h    | Receive overrun wakeup enable<br>0x0 = Receive overrun wakeup disabled<br>0x1 = Receive overrun wakeup enabled  |
| 10    | XUDF_WE  | R/W  | 0h    | Transmit underflow wakeup enable<br>0x0 = Transmit underflow wakeup disabled<br>0x1 = Transmit underflow wakeup enabled   |

**Table 21-16. I2C\_WE Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 9   | AAS_WE   | R/W  | 0h    | <p>Address as slave IRQ wakeup enable.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode, and external master addresses the I2C module as a slave.</p> <p>This allows for the module to inform the CPU that it can check which of the own addresses was used by the external master to access the I2C core.</p> <p>0x0 = Addressed as slave wakeup disabled<br/>0x1 = Addressed as slave wakeup enabled</p>   |
| 8   | BF_WE    | R/W  | 0h    | <p>Bus free IRQ wakeup enable.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode and the I2C bus became free.</p> <p>This allows for the module to inform the CPU that it can initiate its own transfer on the I2C line.</p> <p>0x0 = Bus Free wakeup disabled<br/>0x1 = Bus Free wakeup enabled</p>  |
| 7   | Reserved | R    | 0h    |  |
| 6   | STC_WE   | R/W  | 0h    | <p>Start condition IRQ wakeup set.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode (with the functional clock inactive) and a possible start condition is detected on the I2C line.</p> <p>The STC WakeUp is generated only if the I2C_SYSC.ClockActivity field indicates that the functional clock can be disabled.</p> <p>Note that if the functional clock is not active, the start condition is asynchronously detected (no filtering and synchronization is used). For this reason, it is possible that the signalized start condition to be a glitch.</p> <p>If the functional clock cannot be disabled (I2C_SYSC.ClockActivity = 10 or 11), the programmer should not enable this wakeup, since the module has other synchronously detected WakeUp event that might be used to exit from idle mode, only if the detected transfer is accessing the I2C module.</p> <p>0x0 = Start condition wakeup disabled<br/>0x1 = Start condition wakeup enabled</p> |
| 5   | GC_WE    | R/W  | 0h    | <p>General call IRQ wakeup enable.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is in idle mode and a general call is received on I2C line.</p> <p>0x0 = General call wakeup disabled<br/>0x1 = General call wakeup enabled</p>  |
| 4   | Reserved | R    | 0h    |  |
| 3   | DRDY_WE  | R/W  | 0h    | <p>Receive/Transmit data ready IRQ wakeup enable.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example module are set in F/S I2C master transmitter mode and FIFO is full).</p> <p>If in the middle of such a transaction, the FIFO buffer needs more data to be transferred, CPU must be informed to write (in case of transmitter mode) or read (if receiver mode) in/from the FIFO.</p> <p>0x0 = Transmit/receive data ready wakeup disabled<br/>0x1 = Transmit/receive data ready wakeup enabled</p>   |
| 2   | ARDY_WE  | R/W  | 0h    | <p>Register access ready IRQ wakeup enable.</p> <p>This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example the module is set in F/S I2C master transmitter mode and FIFO is full).</p> <p>If the current transaction is finished, the module needs to inform CPU about transmission completion.</p> <p>0x0 = Register access ready wakeup disabled<br/>0x1 = Register access ready wakeup enabled</p>  |

**Table 21-16. I2C\_WE Register Field Descriptions (continued)**

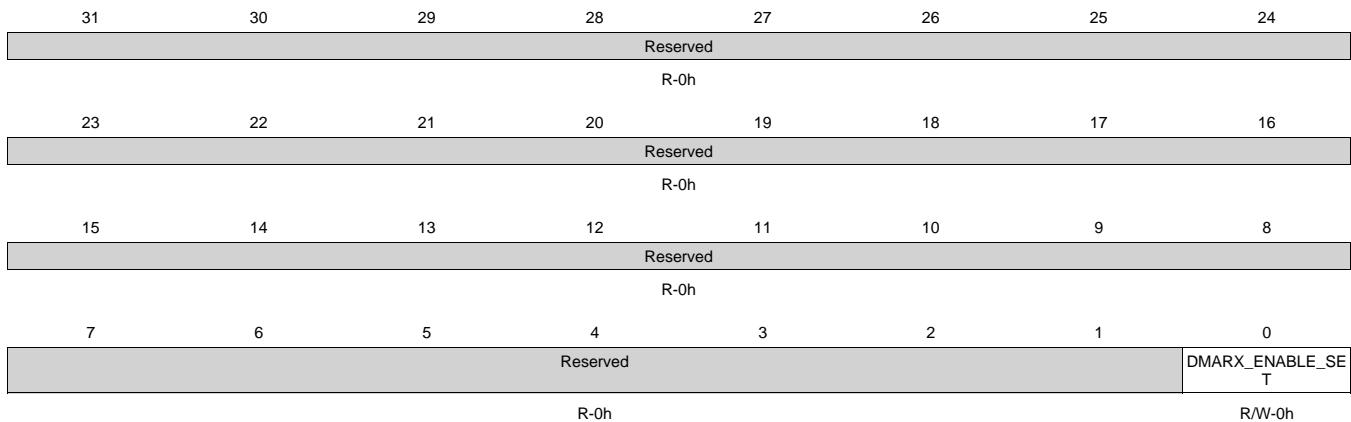
| Bit | Field   | Type | Reset | Description   |
|-----|---------|------|-------|---|
| 1   | NACK_WE | R/W  | 0h    | <p>No acknowledgment IRQ wakeup enable.<br/>           This read/write bit is used to enable or disable wakeup signal generation when I2C module is involved into a long transfer and no more registers accesses are performed on the interface (for example the module is set in F/S I2C master transmitter mode and FIFO is full).</p> <p>If in the middle of such of a transaction a Not Acknowledgment event is raised, the module needs to inform CPU about transmission error.</p> <p>0x0 = Not Acknowledge wakeup disabled<br/>           0x1 = Not Acknowledge wakeup enabled</p>   |
| 0   | AL_WE   | R/W  | 0h    | <p>Arbitration lost IRQ wakeup enable.<br/>           This read/write bit is used to enable or disable wakeup signal generation when I2C module is configured as a master and it loses the arbitration.</p> <p>This wake up is very useful when the module is configured as a master transmitter, all the necessary data is provided in the FIFO Tx, STT is enabled and the module enters in Idle Mode.</p> <p>If the module loses the arbitration, an Arbitration Lost event is raised and the module needs to inform CPU about transmission error.</p> <p>Note: The AL wakeup must be enabled only for multimaster communication.</p> <p>If the AL_WE is not enabled and the scenario described above occurs, the module will not be able to inform the CPU about the state of the transfer and it will be blocked in an undetermined state.</p> <p>0x0 = Arbitration lost wakeup disabled<br/>           0x1 = Arbitration lost wakeup enabled</p> |

### 21.4.1.9 I2C\_DMARXENABLE\_SET Register (offset = 38h) [reset = 0h]

I2C\_DMARXENABLE\_SET is shown in [Figure 21-24](#) and described in [Table 21-17](#).

The 1-bit field enables a receive DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, that is, the register value is not modified. Note that the I2C\_BUF.RDMA\_EN field is the global (slave) DMA enabler, and that it is disabled by default. The I2C\_BUF.RDMA\_EN field should also be set to 1 to enable a receive DMA request.

**Figure 21-24. I2C\_DMARXENABLE\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-17. I2C\_DMARXENABLE\_SET Register Field Descriptions**

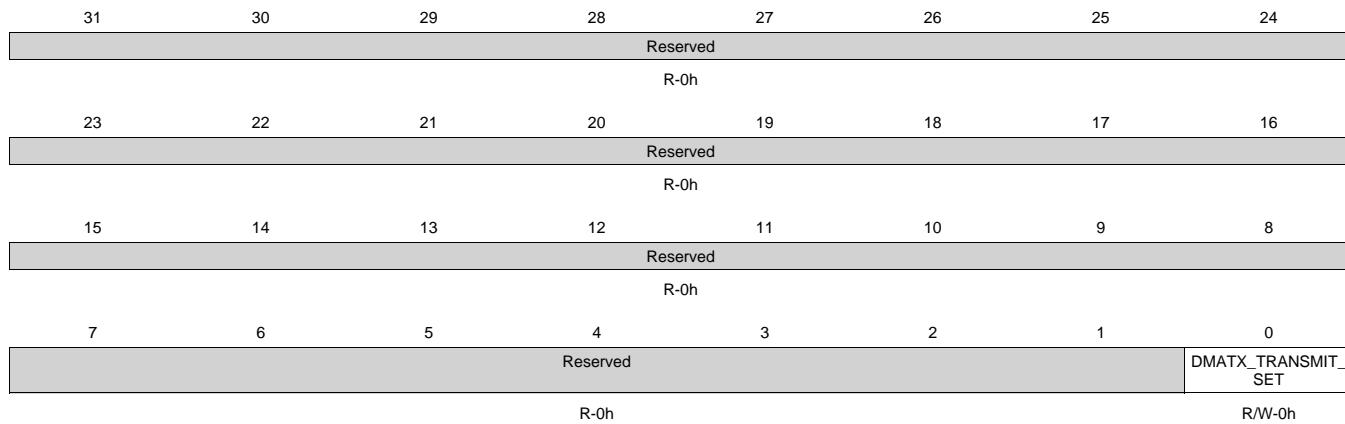
| Bit  | Field            | Type | Reset | Description                     |
|------|------------------|------|-------|---------------------------------|
| 31-1 | Reserved         | R    | 0h    |                                 |
| 0    | DMARX_ENABLE_SET | R/W  | 0h    | Receive DMA channel enable set. |

### 21.4.1.10 I2C\_DMATXENABLE\_SET Register (offset = 3Ch) [reset = 0h]

I2C\_DMATXENABLE\_SET is shown in [Figure 21-25](#) and described in [Table 21-18](#).

The 1-bit field enables a transmit DMA request. Writing a 1 to this field will set it to 1. Writing a 0 will have no effect, that is, the register value is not modified. Note that the I2C\_BUF.XDMA\_EN field is the global (slave) DMA enabler, and that it is disabled by default. The I2C\_BUF.XDMA\_EN field should also be set to 1 to enable a transmit DMA request.

**Figure 21-25. I2C\_DMATXENABLE\_SET Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-18. I2C\_DMATXENABLE\_SET Register Field Descriptions**

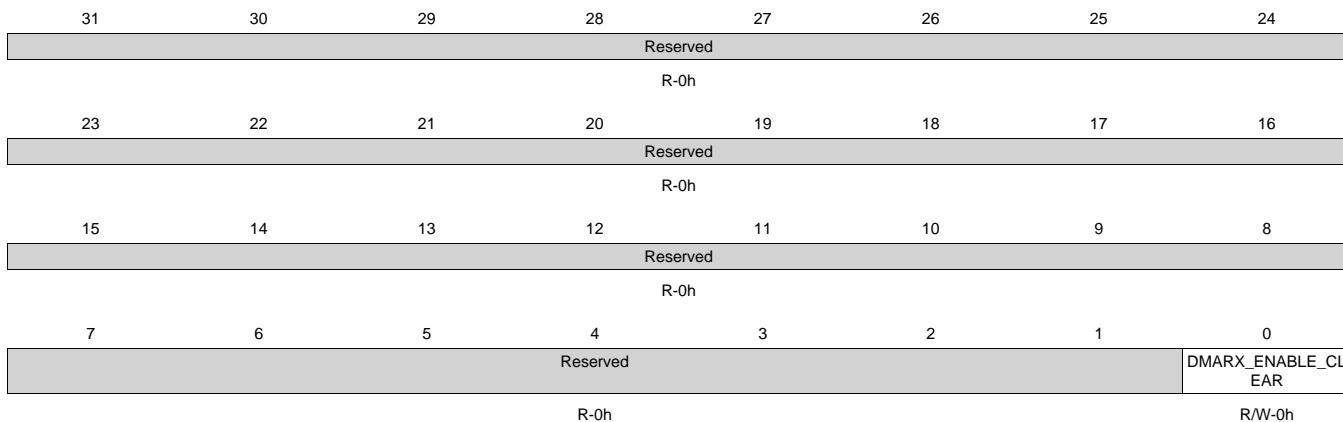
| Bit  | Field              | Type | Reset | Description                      |
|------|--------------------|------|-------|----------------------------------|
| 31-1 | Reserved           | R    | 0h    |                                  |
| 0    | DMATX_TRANSMIT_SET | R/W  | 0h    | Transmit DMA channel enable set. |

### 21.4.1.11 I2C\_DMARXENABLE\_CLR Register (offset = 40h) [reset = 0h]

I2C\_DMARXENABLE\_CLR is shown in [Figure 21-26](#) and described in [Table 21-19](#).

The 1-bit field disables a receive DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to 1 the DMARX\_ENABLE\_CLEAR field, is the reset of the DMA RX request and wakeup lines. Writing a 0 will have no effect, that is, the register value is not modified.

**Figure 21-26. I2C\_DMARXENABLE\_CLR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-19. I2C\_DMARXENABLE\_CLR Register Field Descriptions**

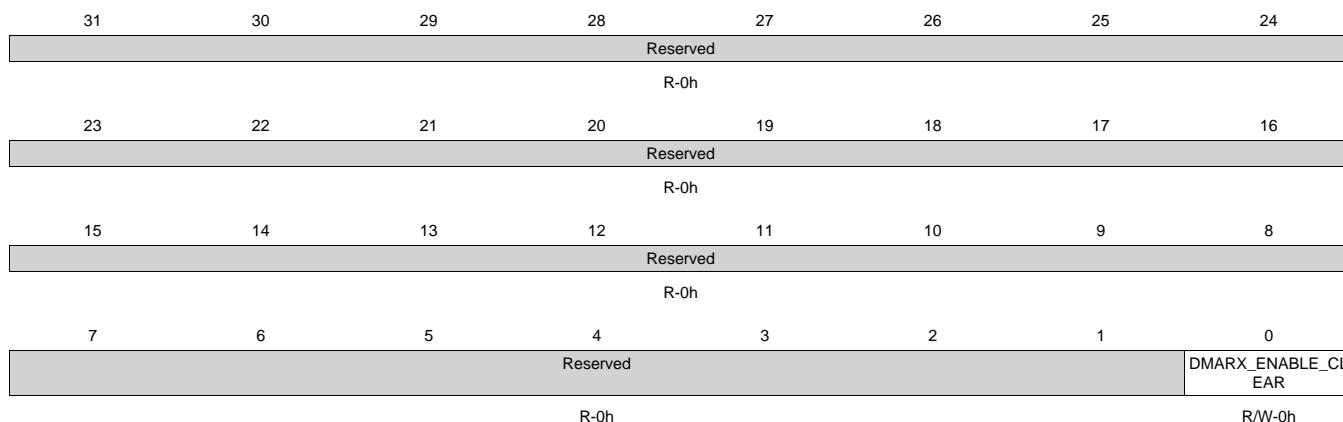
| Bit  | Field              | Type | Reset | Description                       |
|------|--------------------|------|-------|-----------------------------------|
| 31-1 | Reserved           | R    | 0h    |                                   |
| 0    | DMARX_ENABLE_CLEAR | R/W  | 0h    | Receive DMA channel enable clear. |

### 21.4.1.12 I2C\_DMATXENABLE\_CLR Register (offset = 44h) [reset = 0h]

I2C\_DMATXENABLE\_CLR is shown in [Figure 21-27](#) and described in [Table 21-20](#).

The 1-bit field disables a transmit DMA request. Writing a 1 to a bit will clear it to 0. Another result of setting to 1 the DMATX\_ENABLE\_CLEAR field, is the reset of the DMA TX request and wakeup lines. Writing a 0 will have no effect, that is, the register value is not modified.

**Figure 21-27. I2C\_DMATXENABLE\_CLR Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-20. I2C\_DMATXENABLE\_CLR Register Field Descriptions**

| Bit  | Field              | Type | Reset | Description                       |
|------|--------------------|------|-------|-----------------------------------|
| 31-1 | Reserved           | R    | 0h    |                                   |
| 0    | DMARX_ENABLE_CLEAR | R/W  | 0h    | Receive DMA channel enable clear. |

### 21.4.1.13 I2C\_DMARXWAKE\_EN Register (offset = 48h) [reset = 0h]

I2C\_DMARXWAKE\_EN is shown in [Figure 21-28](#) and described in [Table 21-21](#).

All 1-bit fields enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate wakeup line). Note that the I2C\_SYSC.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

**Figure 21-28. I2C\_DMARXWAKE\_EN Register**

|          |        |        |          |        |        |        |        |
|----------|--------|--------|----------|--------|--------|--------|--------|
| 31       | 30     | 29     | 28       | 27     | 26     | 25     | 24     |
| Reserved |        |        |          |        |        |        |        |
| R-0h     |        |        |          |        |        |        |        |
| 23       | 22     | 21     | 20       | 19     | 18     | 17     | 16     |
| Reserved |        |        |          |        |        |        |        |
| R-0h     |        |        |          |        |        |        |        |
| 15       | 14     | 13     | 12       | 11     | 10     | 9      | 8      |
| Reserved | XDR    | RDR    | Reserved | ROVR   | XUDF   | AAS    | BF     |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6      | 5      | 4        | 3      | 2      | 1      | 0      |
| Reserved | STC    | GC     | Reserved | DRDY   | ARDY   | NACK   | AL     |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-21. I2C\_DMARXWAKE\_EN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-15 | Reserved | R    | 0h    |   |
| 14    | XDR      | R/W  | 0h    | Transmit draining wakeup set.<br>0x0 = Transmit draining interrupt disabled<br>0x1 = Transmit draining interrupt enabled      |
| 13    | RDR      | R/W  | 0h    | Receive draining wakeup set.<br>0x0 = Receive draining interrupt disabled<br>0x1 = Receive draining interrupt enabled         |
| 12    | Reserved | R    | 0h    |   |
| 11    | ROVR     | R/W  | 0h    | Receive overrun wakeup set.<br>0x0 = Receive overrun interrupt disabled<br>0x1 = Receive draining interrupt enabled           |
| 10    | XUDF     | R/W  | 0h    | Transmit underflow wakeup set.<br>0x0 = Transmit underflow interrupt disabled<br>0x1 = Transmit underflow interrupt enabled   |
| 9     | AAS      | R/W  | 0h    | Address as slave IRQ wakeup set.<br>0x0 = Addressed as slave interrupt disabled<br>0x1 = Addressed as slave interrupt enabled |
| 8     | BF       | R/W  | 0h    | Bus free IRQ wakeup set.<br>0x0 = Bus free wakeup disabled<br>0x1 = Bus free wakeup enabled                                   |
| 7     | Reserved | R    | 0h    |   |
| 6     | STC      | R/W  | 0h    | Start condition IRQ wakeup set.<br>0x0 = Start condition wakeup disabled<br>0x1 = Start condition wakeup enabled              |
| 5     | GC       | R/W  | 0h    | General call IRQ wakeup set.<br>0x0 = General call wakeup disabled<br>0x1 = General call wakeup enabled                       |

**Table 21-21. I2C\_DMARXWAKE\_EN Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 4   | Reserved | R    | 0h    |  |
| 3   | DRDY     | R/W  | 0h    | Receive/transmit data ready IRQ wakeup set.<br>0x0 = Transmit/receive data ready wakeup disabled<br>0x1 = Transmit/receive data ready wakeup enabled |
| 2   | ARDY     | R/W  | 0h    | Register access ready IRQ wakeup set.<br>0x0 = Register access ready wakeup disabled<br>0x1 = Register access ready wakeup enabled                   |
| 1   | NACK     | R/W  | 0h    | No acknowledgment IRQ wakeup set.<br>0x0 = Not Acknowledge wakeup disabled<br>0x1 = Not Acknowledge wakeup enabled                                   |
| 0   | AL       | R/W  | 0h    | Arbitration lost IRQ wakeup set.<br>0x0 = Arbitration lost wakeup disabled<br>0x1 = Arbitration lost wakeup enabled                                  |

### 21.4.1.14 I2C\_DMATXWAKE\_EN Register (offset = 4Ch) [reset = 0h]

I2C\_DMATXWAKE\_EN is shown in [Figure 21-29](#) and described in [Table 21-22](#).

All 1-bit fields enable a specific (synchronous) DMA request source to generate an asynchronous wakeup (on the appropriate wakeup line). Note that the I2C\_SYSC.ENAWAKEUP field is the global (slave) wakeup enabler, and that it is disabled by default.

**Figure 21-29. I2C\_DMATXWAKE\_EN Register**

|          |        |        |          |        |        |        |        |
|----------|--------|--------|----------|--------|--------|--------|--------|
| 31       | 30     | 29     | 28       | 27     | 26     | 25     | 24     |
| Reserved |        |        |          |        |        |        |        |
| R-0h     |        |        |          |        |        |        |        |
| 23       | 22     | 21     | 20       | 19     | 18     | 17     | 16     |
| Reserved |        |        |          |        |        |        |        |
| R-0h     |        |        |          |        |        |        |        |
| 15       | 14     | 13     | 12       | 11     | 10     | 9      | 8      |
| Reserved | XDR    | RDR    | Reserved | ROVR   | XUDF   | AAS    | BF     |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6      | 5      | 4        | 3      | 2      | 1      | 0      |
| Reserved | STC    | GC     | Reserved | DRDY   | ARDY   | NACK   | AL     |
| R-0h     | R/W-0h | R/W-0h | R-0h     | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-22. I2C\_DMATXWAKE\_EN Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-15 | Reserved | R    | 0h    |   |
| 14    | XDR      | R/W  | 0h    | Transmit draining wakeup set.<br>0x0 = Transmit draining interrupt disabled<br>0x1 = Transmit draining interrupt enabled      |
| 13    | RDR      | R/W  | 0h    | Receive draining wakeup set.<br>0x0 = Receive draining interrupt disabled<br>0x1 = Receive draining interrupt enabled         |
| 12    | Reserved | R    | 0h    |   |
| 11    | ROVR     | R/W  | 0h    | Receive overrun wakeup set.<br>0x0 = Receive overrun interrupt disabled<br>0x1 = Receive draining interrupt enabled           |
| 10    | XUDF     | R/W  | 0h    | Transmit underflow wakeup set.<br>0x0 = Transmit underflow interrupt disabled<br>0x1 = Transmit underflow interrupt enabled   |
| 9     | AAS      | R/W  | 0h    | Address as slave IRQ wakeup set.<br>0x0 = Addressed as slave interrupt disabled<br>0x1 = Addressed as slave interrupt enabled |
| 8     | BF       | R/W  | 0h    | Bus free IRQ wakeup set.<br>0x0 = Bus free wakeup disabled<br>0x1 = Bus free wakeup enabled                                   |
| 7     | Reserved | R    | 0h    |   |
| 6     | STC      | R/W  | 0h    | Start condition IRQ wakeup set.<br>0x0 = Start condition wakeup disabled<br>0x1 = Start condition wakeup enabled              |
| 5     | GC       | R/W  | 0h    | General call IRQ wakeup set.<br>0x0 = General call wakeup disabled<br>0x1 = General call wakeup enabled                       |

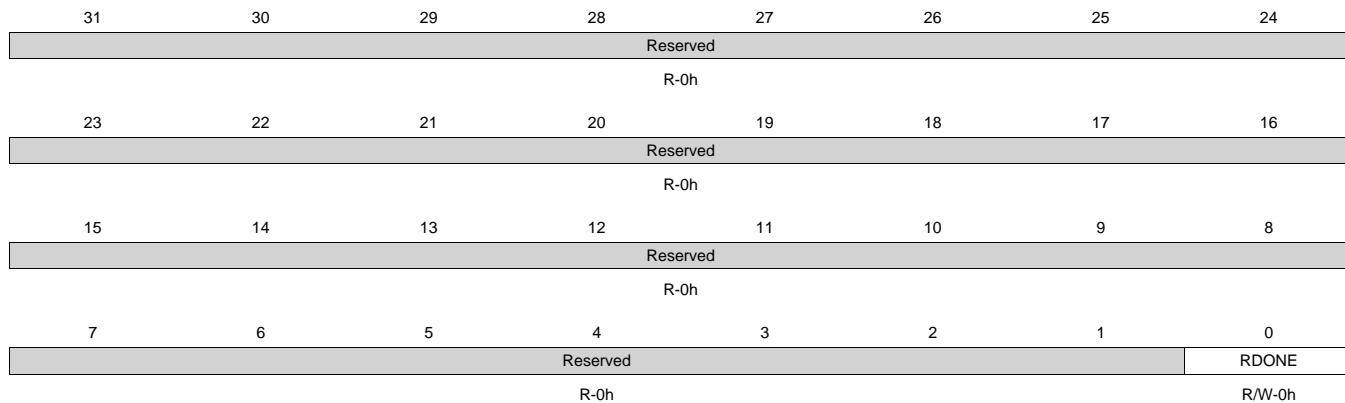
**Table 21-22. I2C\_DMATXWAKE\_EN Register Field Descriptions (continued)**

| Bit | Field    | Type | Reset | Description  |
|-----|----------|------|-------|--|
| 4   | Reserved | R    | 0h    |  |
| 3   | DRDY     | R/W  | 0h    | Receive/transmit data ready IRQ wakeup set.<br>0x0 = Transmit/receive data ready wakeup disabled<br>0x1 = Transmit/receive data ready wakeup enabled |
| 2   | ARDY     | R/W  | 0h    | Register access ready IRQ wakeup set.<br>0x0 = Register access ready wakeup disabled<br>0x1 = Register access ready wakeup enabled                   |
| 1   | NACK     | R/W  | 0h    | No acknowledgment IRQ wakeup set.<br>0x0 = Not Acknowledge wakeup disabled<br>0x1 = Not Acknowledge wakeup enabled                                   |
| 0   | AL       | R/W  | 0h    | Arbitration lost IRQ wakeup set.<br>0x0 = Arbitration lost wakeup disabled<br>0x1 = Arbitration lost wakeup enabled                                  |

### 21.4.1.15 I2C\_SYSS Register (offset = 90h) [reset = 0h]

I2C\_SYSS is shown in [Figure 21-30](#) and described in [Table 21-23](#).

**Figure 21-30. I2C\_SYSS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-23. I2C\_SYSS Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-1 | Reserved | R    | 0h    |  |
| 0    | RDONE    | R/W  | 0h    | <p>Reset done bit.</p> <p>This read-only bit indicates the state of the reset in case of hardware reset, global software reset (I2C_SYSC.SRST) or partial software reset (I2C_CON.I2C_EN).</p> <p>The module must receive all its clocks before it can grant a reset-completed status.</p> <p>Value after reset is low.</p> <p>0x0 = Internal module reset in ongoing</p> <p>0x1 = Reset completed</p> |

### 21.4.1.16 I2C\_BUF Register (offset = 94h) [reset = 0h]

I2C\_BUF is shown in [Figure 21-31](#) and described in [Table 21-24](#).

This read/write register enables DMA transfers and allows the configuration of FIFO thresholds for the FIFO management (see the FIFO Management subsection).

**Figure 21-31. I2C\_BUF Register**

|          |            |    |    |        |    |    |    |
|----------|------------|----|----|--------|----|----|----|
| 31       | 30         | 29 | 28 | 27     | 26 | 25 | 24 |
| Reserved |            |    |    |        |    |    |    |
| R-0h     |            |    |    |        |    |    |    |
| 23       | 22         | 21 | 20 | 19     | 18 | 17 | 16 |
| Reserved |            |    |    |        |    |    |    |
| R-0h     |            |    |    |        |    |    |    |
| 15       | 14         | 13 | 12 | 11     | 10 | 9  | 8  |
| RDMA_EN  | RXFIFO_CLR |    |    | RXTRSH |    |    |    |
| R/W-0h   | R/W-0h     |    |    | R/W-0h |    |    |    |
| 7        | 6          | 5  | 4  | 3      | 2  | 1  | 0  |
| XDMA_EN  | TXFIFO_CLR |    |    | TXTRSH |    |    |    |
| R/W-0h   | R/W-0h     |    |    | R/W-0h |    |    |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-24. I2C\_BUF Register Field Descriptions**

| Bit   | Field      | Type | Reset | Description  |
|-------|------------|------|-------|--|
| 31-16 | Reserved   | R    | 0h    |  |
| 15    | RDMA_EN    | R/W  | 0h    | <p>Receive DMA channel enable.<br/> When this bit is set to 1, the receive DMA channel is enabled and the receive data ready status bit (I2C_IRQSTATUS_RAW: RRDY) is forced to 0 by the core.<br/> Value after reset is low.<br/> 0x0 = Receive DMA channel disabled<br/> 0x1 = Receive DMA channel enabled</p>  |
| 14    | RXFIFO_CLR | R/W  | 0h    | <p>Receive FIFO clear.<br/> When set, receive FIFO is cleared (hardware reset for RX FIFO generated).<br/> This bit is automatically reset by the hardware.<br/> During reads, it always returns 0.<br/> Value after reset is low.<br/> 0x0 = Normal mode<br/> 0x1 = Rx FIFO is reset</p>  |
| 13-8  | RXTRSH     | R/W  | 0h    | <p>Threshold value for FIFO buffer in RX mode.<br/> The receive threshold value is used to specify the trigger level for data receive transfers.<br/> The value is specified from the Interface/OCP point of view.<br/> Value after reset is 00h.<br/> For the FIFO management description, see the FIFO Management subsection.<br/> Note<br/> 1: programmed threshold cannot exceed the actual depth of the FIFO.<br/> Note<br/> 2: the threshold must not be changed while a transfer is in progress (after STT was configured or after the module was addressed as a slave).<br/> 0x0 = Receive Threshold value = 1<br/> 0x1 = Receive Threshold value = 2<br/> 0x3F = Receive Threshold value = 64</p> |

**Table 21-24. I2C\_BUF Register Field Descriptions (continued)**

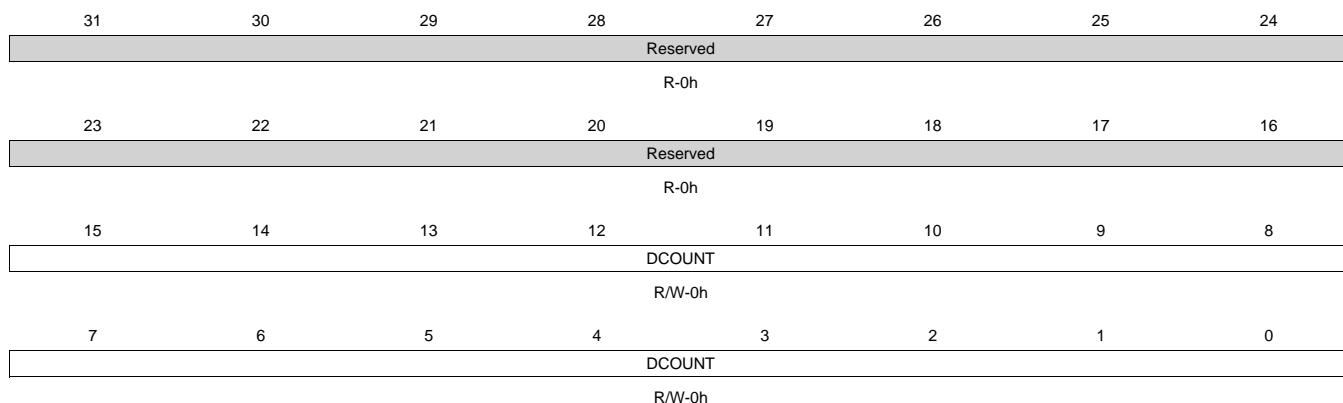
| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 7   | XDMA_EN    | R/W  | 0h    | <p>Transmit DMA channel enable.<br/> When this bit is set to 1, the transmit DMA channel is enabled and the transmit data ready status (I2C_IRQSTATUS_RAW: XRDY) bit is forced to 0 by the core.<br/> Value after reset is low.<br/> 0x0 = Transmit DMA channel disabled<br/> 0x1 = Transmit DMA channel enabled</p>   |
| 6   | TXFIFO_CLR | R/W  | 0h    | <p>Transmit FIFO clear.<br/> When set, transmit FIFO is cleared (hardware reset for TX FIFO). This bit is automatically reset by the hardware.<br/> During reads, it always returns 0.<br/> Value after reset is low.<br/> 0x0 = Normal mode<br/> 0x1 = Tx FIFO is reset</p>   |
| 5-0 | TXTRSH     | R/W  | 0h    | <p>Threshold value for FIFO buffer in TX mode.<br/> The Transmit Threshold value is used to specify the trigger level for data transfers.<br/> The value is specified from the OCP point of view.<br/> Value after reset is 00h Note<br/> 1: programmed threshold cannot exceed the actual depth of the FIFO.<br/> Note<br/> 2: the threshold must not be changed while a transfer is in progress (after STT was configured or after the module was addressed as a slave).<br/> 0x0 = Transmit Threshold value = 1<br/> 0x1 = Transmit Threshold value = 2<br/> 0x3F = Transmit Threshold value = 64</p> |

### 21.4.1.17 I2C\_CNT Register (offset = 98h) [reset = 0h]

I2C\_CNT is shown in [Figure 21-32](#) and described in [Table 21-25](#).

**CAUTION:** During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This read/write register is used to control the numbers of bytes in the I2C data payload.

**Figure 21-32. I2C\_CNT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-25. I2C\_CNT Register Field Descriptions**

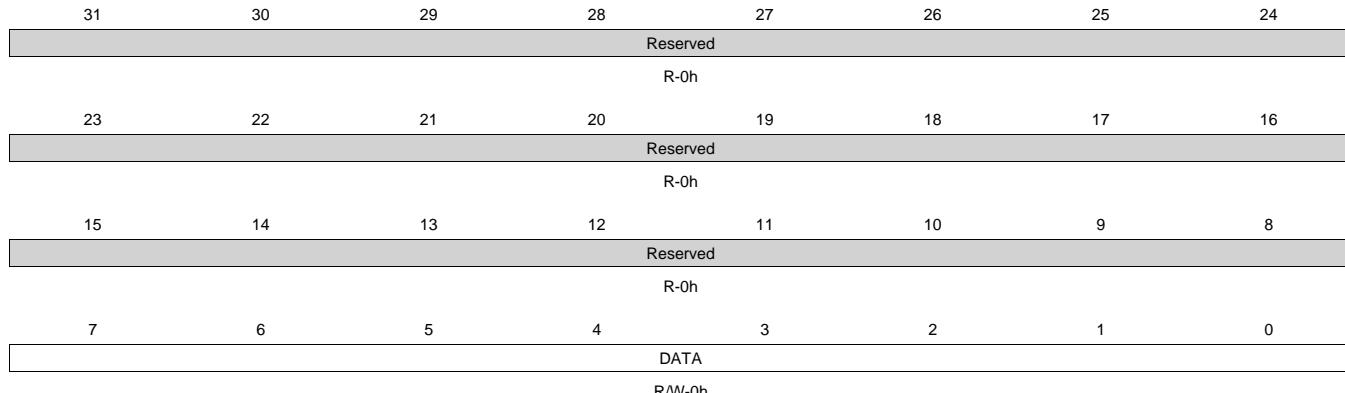
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-16 | Reserved | R    | 0h    |   |
| 15-0  | DCOUNT   | R/W  | 0h    | <p>Data count.<br/>I2C Master Mode only (receive or transmit F/S).<br/>This<br/>16-bit countdown counter decrements by 1 for every byte received or sent through the I2C interface.<br/>A write initializes DCOUNT to a saved initial value.<br/>A read returns the number of bytes that are yet to be received or sent.<br/>A read into DCOUNT returns the initial value only before a start condition and after a stop condition.<br/>When DCOUNT reaches 0, the core generates a stop condition if a stop condition was specified (I2C_CON.STP = 1) and the ARDY status flag is set to 1 in the I2C_IRQSTATUS_RAW register.<br/>Note that DCOUNT must not be reconfigured after I2C_CON.STT was enabled and before ARDY is received.<br/>Note<br/>1: In case of I2C mode of operation, if I2C_CON.STP = 0, then the I2C asserts SCL = 0 when DCOUNT reaches 0.<br/>The CPU can then reprogram DCOUNT to a new value and resume sending or receiving data with a new start condition (restart).<br/>This process repeats until the CPU sets to 1 the I2C_CON.STP bit.<br/>The ARDY flag is set each time DCOUNT reaches 0 and DCOUNT is reloaded to its initial value.<br/>Values after reset are low (all 16 bits).<br/>Note<br/>2: Since for DCOUNT = 0, the transfer length is 65536, the module does not allow the possibility to initiate zero data bytes transfers.<br/>0x0 = Data counter = 65536 bytes (216)<br/>0x1 = Data counter = 1 bytes<br/>0xFFFF = Data counter = 65535 bytes (216 - 1)</p> |

### 21.4.1.18 I2C\_DATA Register (offset = 9Ch) [reset = 0h]

I2C\_DATA is shown in [Figure 21-33](#) and described in [Table 21-26](#).

This register is the entry point for the local host to read data from or write data to the FIFO buffer.

**Figure 21-33. I2C\_DATA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-26. I2C\_DATA Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | DATA     | R/W  | 0h    | <p>Transmit/Receive data FIFO endpoint.<br/>           When read, this register contains the received I2C data.<br/>           When written, this register contains the byte value to transmit over the I2C data.<br/>           In SYSTEST loop back mode (I2C_SYSTEST: TMODE = 11), this register is also the entry/receive point for the data.<br/>           Values after reset are unknown (all 8-bits).<br/>           Note: A read access, when the buffer is empty, returns the previous read data value.<br/>           A write access, when the buffer is full, is ignored.<br/>           In both events, the FIFO pointers are not updated and an Access Error (AERR) Interrupt is generated.</p> |

### 21.4.1.19 I2C\_CON Register (offset = A4h) [reset = 0h]

I2C\_CON is shown in [Figure 21-34](#) and described in [Table 21-27](#).

During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register (except STP enable). Changing it may result in an unpredictable behavior.

**Figure 21-34. I2C\_CON Register**

|          |          |        |        |          |        |        |        |
|----------|----------|--------|--------|----------|--------|--------|--------|
| 31       | 30       | 29     | 28     | 27       | 26     | 25     | 24     |
| Reserved |          |        |        |          |        |        |        |
| R-0h     |          |        |        |          |        |        |        |
| 23       | 22       | 21     | 20     | 19       | 18     | 17     | 16     |
| Reserved |          |        |        |          |        |        |        |
| R-0h     |          |        |        |          |        |        |        |
| 15       | 14       | 13     | 12     | 11       | 10     | 9      | 8      |
| I2C_EN   | Reserved | OPMODE |        | STB      | MST    | TRX    | XSA    |
| R/W-0h   | R-0h     | R/W-0h |        | R/W-0h   | R/W-0h | R/W-0h | R/W-0h |
| 7        | 6        | 5      | 4      | 3        | 2      | 1      | 0      |
| XOA0     | XOA1     | XOA2   | XOA3   | Reserved |        | STP    | STT    |
| R/W-0h   | R/W-0h   | R/W-0h | R/W-0h | R-0h     |        | R/W-0h | R/W-0h |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-27. I2C\_CON Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15    | I2C_EN   | R/W  | 0h    | I2C module enable.<br>When this bit is cleared to 0, the I2C controller is not enabled and reset.<br>When 0, receive and transmit FIFOs are cleared and all status bits are set to their default values.<br>All configuration registers (I2C_IRQENABLE_SET, I2C_IRQWAKE_SET, I2C_BUF, I2C_CNT, I2C_CON, I2C_OA, I2C_SA, I2C_PSC, I2C_SCLL and I2C_SCLH) are not reset, they keep their initial values and can be accessed.<br>The CPU must set this bit to 1 for normal operation.<br>Value after reset is low.<br>0x0 = Controller in reset. FIFO are cleared and status bits are set to their default value.<br>0x1 = Module enabled |
| 14    | Reserved | R    | 0h    |  |
| 13-12 | OPMODE   | R/W  | 0h    | Operation mode selection.<br>These two bits select module operation mode.<br>Value after reset is 00.<br>0x0 = I2C Fast/Standard mode<br>0x1 = Reserved<br>0x2 = Reserved<br>0x3 = Reserved  |
| 11    | STB      | R/W  | 0h    | Start byte mode (I2C master mode only).<br>The start byte mode bit is set to 1 by the CPU to configure the I2C in start byte mode (I2C_SA = 0000 0001).<br>See the Philips I2C spec for more details [1].<br>Value after reset is low.<br>0x0 = Normal mode<br>0x1 = Start byte mode   |

Table 21-27. I2C\_CON Register Field Descriptions (continued)

| Bit | Field | Type | Reset | Description  |
|-----|-------|------|-------|--|
| 10  | MST   | R/W  | 0h    | <p>Master/slave mode (I2C mode only).<br/> When this bit is cleared, the I2C controller is in the slave mode and the serial clock (SCL) is received from the master device.<br/> When this bit is set, the I2C controller is in the master mode and generates the serial clock.<br/> Note: This bit is automatically cleared at the end of the transfer on a detected stop condition, in case of arbitration lost or when the module is configured as a master but addressed as a slave by an external master.<br/> Value after reset is low.<br/> 0x0 = Slave mode<br/> 0x1 = Master mode</p>   |
| 9   | TRX   | R/W  | 0h    | <p>Transmitter/receiver mode (I2C master mode only).<br/> When this bit is cleared, the I2C controller is in the receiver mode and data on data line SDA is shifted into the receiver FIFO and can be read from I2C_DATA register.<br/> When this bit is set, the I2C controller is in the transmitter mode and the data written in the transmitter FIFO via I2C_DATA is shifted out on data line SDA.<br/> Value after reset is low.<br/> The operating modes are defined as follows: MST = 0, TRX = x, Operating Mode = Slave receiver.<br/> MST = 0, TRX = x, Operating Mode = Slave transmitter.<br/> MST = 1, TRX = 0, Operating Modes = Master receiver.<br/> MST = 1, TRX = 0, Operating Modes = Master transmitter.<br/> 0x0 = Receiver mode<br/> 0x1 = Transmitter mode</p> |
| 8   | XSA   | R/W  | 0h    | <p>Expand slave address.<br/> (I2C mode only).<br/> When set, this bit expands the slave address to 10-bit.<br/> Value after reset is low.<br/> 0x0 = 7-bit address mode<br/> 0x1 = 10-bit address mode</p>  |
| 7   | XOA0  | R/W  | 0h    | <p>Expand own address 0.<br/> (I2C mode only).<br/> When set, this bit expands the base own address (OA0) to 10-bit.<br/> Value after reset is low.<br/> 0x0 = 7-bit address mode<br/> 0x1 = 10-bit address mode</p>   |
| 6   | XOA1  | R/W  | 0h    | <p>Expand own address 1.<br/> (I2C mode only).<br/> When set, this bit expands the first alternative own address (OA1) to 10-bit.<br/> Value after reset is low.<br/> 0x0 = 7-bit address mode<br/> 0x1 = 10-bit address mode</p>  |
| 5   | XOA2  | R/W  | 0h    | <p>Expand own address 2.<br/> (I2C mode only).<br/> When set, this bit expands the second alternative own address (OA2) to 10-bit.<br/> Value after reset is low.<br/> 0x0 = 7-bit address mode. (I2C mode only).<br/> 0x1 = 10-bit address mode</p>   |
| 4   | XOA3  | R/W  | 0h    | <p>Expand own address 3.<br/> When set, this bit expands the third alternative own address (OA3) to 10-bit.<br/> Value after reset is low.<br/> 0x0 = 7-bit address mode<br/> 0x1 = 10-bit address mode</p>  |

**Table 21-27. I2C\_CON Register Field Descriptions (continued)**

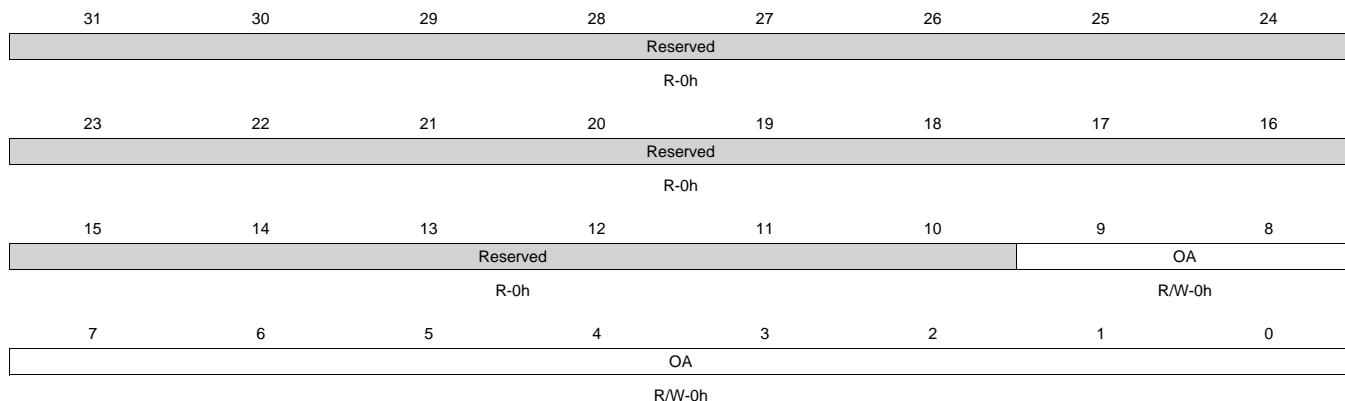
| Bit | Field    | Type | Reset | Description   |
|-----|----------|------|-------|---|
| 3-2 | Reserved | R    | 0h    |   |
| 1   | STP      | R/W  | 0h    | <p>Stop condition (I2C master mode only).<br/>           This bit can be set to a 1 by the CPU to generate a stop condition.<br/>           It is reset to 0 by the hardware after the stop condition has been generated.</p> <p>The stop condition is generated when DCOUNT passes 0.<br/>           When this bit is not set to 1 before the end of the transfer (DCOUNT = 0), the stop condition is not generated and the SCL line is held to 0 by the master, which can re-start a new transfer by setting the STT bit to 1.</p> <p>Value after reset is low<br/>           0x0 = No action or stop condition detected<br/>           0x1 = Stop condition queried</p>  |
| 0   | STT      | R/W  | 0h    | <p>Start condition (I2C master mode only).<br/>           This bit can be set to a 1 by the CPU to generate a start condition.<br/>           It is reset to 0 by the hardware after the start condition has been generated.</p> <p>The start/stop bits can be configured to generate different transfer formats.</p> <p>Value after reset is low.<br/>           Note: DCOUNT is data count value in I2C_CNT register.<br/>           STT = 1, STP = 0, Conditions = Start, Bus Activities = S-A-D.<br/>           STT = 0, STP = 1, Conditions = Stop, Bus Activities = P.<br/>           STT = 1, STP = 1, Conditions = Start-Stop (DCOUNT=n), Bus Activities = S-A-D..(n)..D-P.<br/>           STT = 1, STP = 0, Conditions = Start (DCOUNT=n), Bus Activities = S-A-D..(n)..D.<br/>           0x0 = No action or start condition detected<br/>           0x1 = Start condition queried</p> |

### 21.4.1.20 I2C\_OA Register (offset = A8h) [reset = 0h]

I2C\_OA is shown in [Figure 21-35](#) and described in [Table 21-28](#).

**CAUTION:** During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the module's base I2C 7-bit or 10-bit address (base own address).

**Figure 21-35. I2C\_OA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-28. I2C\_OA Register Field Descriptions**

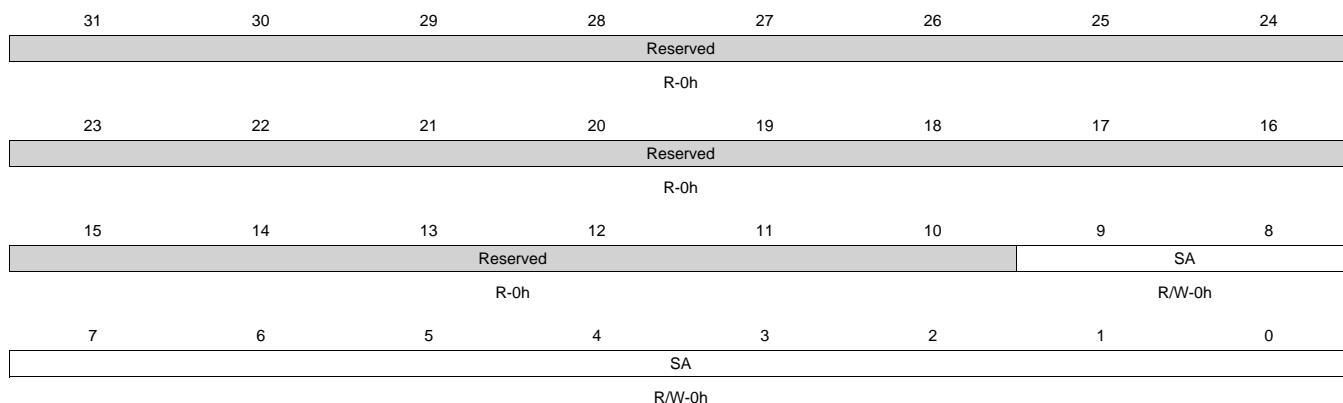
| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-10 | Reserved | R    | 0h    |  |
| 9-0   | OA       | R/W  | 0h    | Own address.<br>This field specifies either: A 10-bit address coded on OA [9:0] when XOA (Expand Own Address, I2C_CON[7]) is set to 1. or A 7-bit address coded on OA [6:0] when XOA (Expand Own Address, I2C_CON[7]) is cleared to 0. In this case, OA [9:7] bits must be cleared to 000 by application software. Value after reset is low (all 10 bits). |

### 21.4.1.21 I2C\_SA Register (offset = ACh) [reset = 0h]

I2C\_SA is shown in [Figure 21-36](#) and described in [Table 21-29](#).

**CAUTION:** During an active transfer phase (between STT having been set to 1 and reception of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the addressed I2C module 7-bit or 10-bit address (slave address).

**Figure 21-36. I2C\_SA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-29. I2C\_SA Register Field Descriptions**

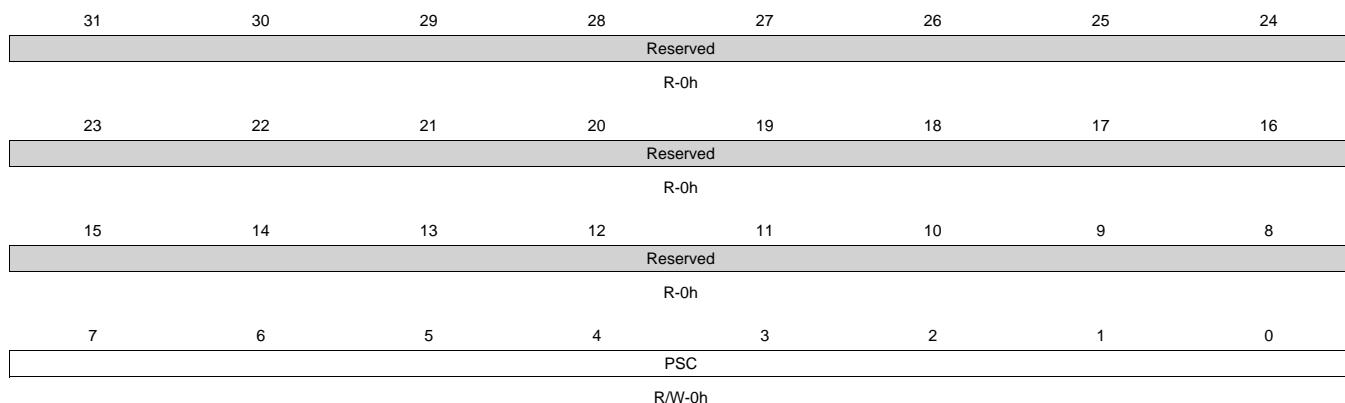
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-10 | Reserved | R    | 0h    |   |
| 9-0   | SA       | R/W  | 0h    | <p>Slave address.</p> <p>This field specifies either: A 10-bit address coded on SA [9:0] when XSA (Expand Slave Address, I2C_CON[8]) is set to 1, or A 7-bit address coded on SA [6:0] when XSA (Expand Slave Address, I2C_CON[8]) is cleared to 0.</p> <p>In this case, SA [9:7] bits must be cleared to 000 by application software.</p> <p>Value after reset is low (all 10 bits).</p> |

### 21.4.1.22 I2C\_PSC Register (offset = B0h) [reset = 0h]

I2C\_PSC is shown in [Figure 21-37](#) and described in [Table 21-30](#).

**CAUTION:** During an active mode (I2C\_EN bit in I2C\_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the internal clocking of the I2C peripheral core.

**Figure 21-37. I2C\_PSC Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-30. I2C\_PSC Register Field Descriptions**

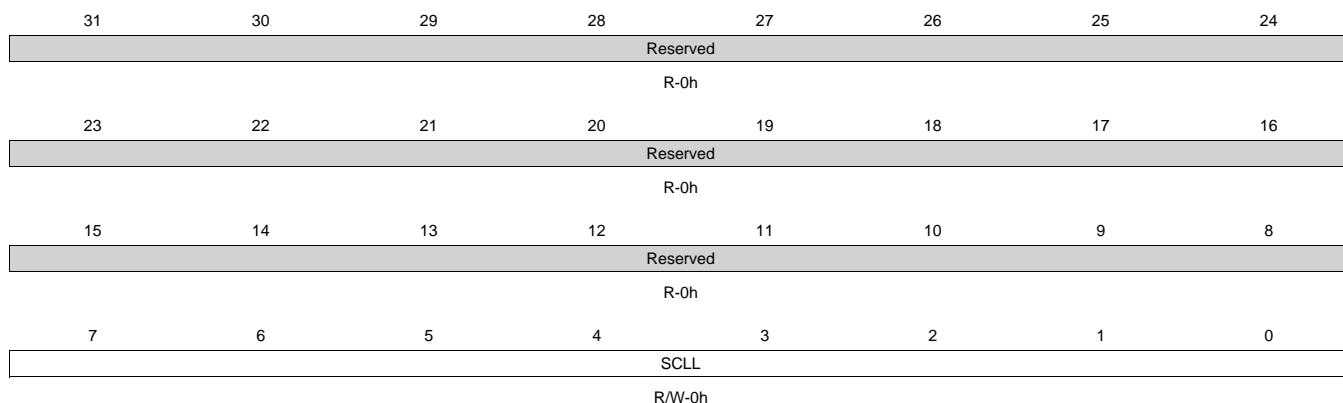
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | PSC      | R/W  | 0h    | Fast/Standard mode prescale sampling clock divider value.<br>The core uses this 8-bit value to divide the system clock (SCLK) and generates its own internal sampling clock (ICLK) for Fast and Standard operation modes.<br>The core logic is sampled at the clock rate of the system clock for the module divided by (PSC + 1).<br>Value after reset is low (all 8 bits).<br>0x0 = Divide by 1<br>0x1 = Divide by 2<br>0xFF = Divide by 256 |

### 21.4.1.23 I2C\_SCLL Register (offset = B4h) [reset = 0h]

I2C\_SCLL is shown in [Figure 21-38](#) and described in [Table 21-31](#).

**CAUTION:** During an active mode (I2C\_EN bit in I2C\_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to determine the SCL low time value when master.

**Figure 21-38. I2C\_SCLL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-31. I2C\_SCLL Register Field Descriptions**

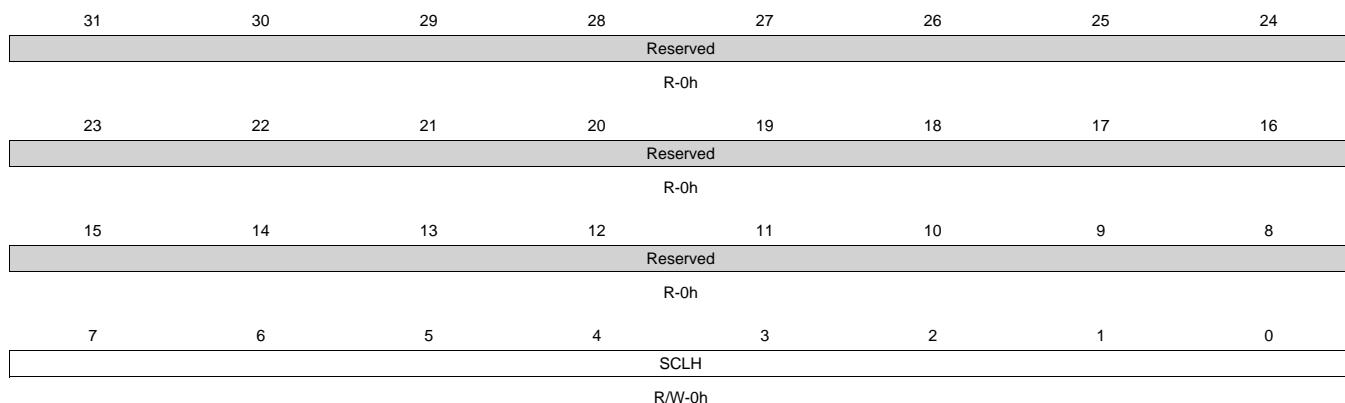
| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-8 | Reserved | R    | 0h    |   |
| 7-0  | SCLL     | R/W  | 0h    | Fast/Standard mode SCL low time.<br>I2C master mode only, (FS).<br>This<br>8-bit value is used to generate the SCL low time value (tLOW) when<br>the peripheral is operated in master mode.<br>$tLOW = (SCLL + 7) * \text{ICLK}$ time period, Value after reset is low (all 8<br>bits). |

#### 21.4.1.24 I2C\_SCLH Register (offset = B8h) [reset = 0h]

I2C\_SCLH is shown in [Figure 21-39](#) and described in [Table 21-32](#).

**CAUTION:** During an active mode (I2C\_EN bit in I2C\_CON register is set to 1), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to determine the SCL high time value when master.

**Figure 21-39. I2C\_SCLH Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-32. I2C\_SCLH Register Field Descriptions**

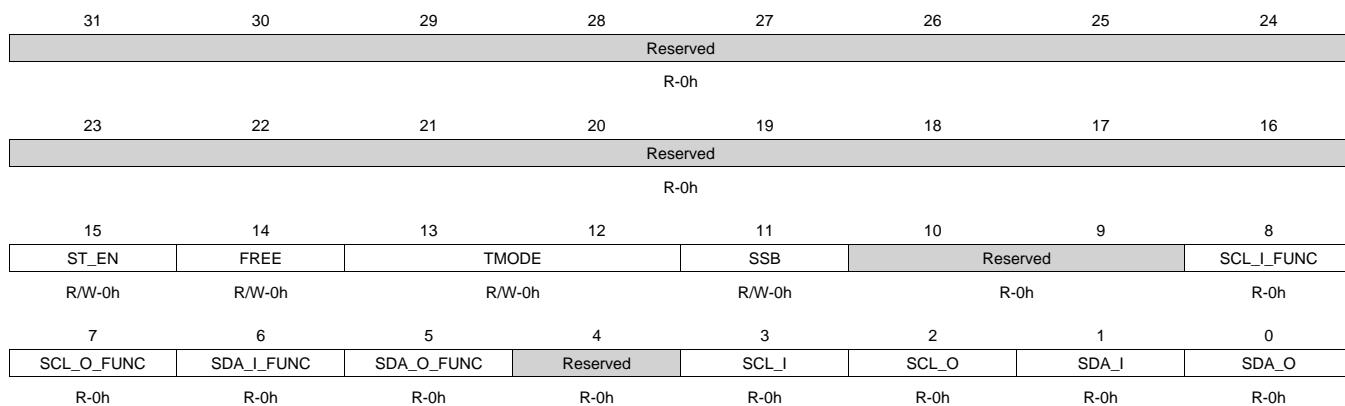
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-8 | Reserved | R    | 0h    |  |
| 7-0  | SCLH     | R/W  | 0h    | <p>Fast/Standard mode SCL low time.<br/>I2C master mode only, (FS).</p> <p>This 8-bit value is used to generate the SCL high time value (<math>t_{HIGH}</math>) when the peripheral is operated in master mode.</p> <p>- <math>t_{HIGH} = (SCLH + 5) * \text{ICLK}</math> time period.</p> <p>Value after reset is low (all 8 bits).</p> |

### 21.4.1.25 I2C\_SYSTEST Register (offset = BCh) [reset = 0h]

I2C\_SYSTEST is shown in [Figure 21-40](#) and described in [Table 21-33](#).

**CAUTION:** Never enable this register for normal I2C operation. This register is used to facilitate system-level tests by overriding some of the standard functional features of the peripheral. It allows testing of SCL counters, controlling the signals that connect to I/O pins, or creating digital loop-back for self-test when the module is configured in system test (SYSTEST) mode. It also provides stop/non-stop functionality in the debug mode.

**Figure 21-40. I2C\_SYSTEST Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-33. I2C\_SYSTEST Register Field Descriptions**

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-16 | Reserved | R    | 0h    |  |
| 15    | ST_EN    | R/W  | 0h    | System test enable.<br>This bit must be set to 1 to permit other system test registers bits to be set.<br>Value after reset is low.<br>0x0 = Normal mode. All other bits in register are read only.<br>0x1 = System test enabled. Permit other system test registers bits to be set. |

Table 21-33. I2C\_SYSTEST Register Field Descriptions (continued)

| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 14    | FREE     | R/W  | 0h    | <p>Free running mode (on breakpoint).<br/> This bit is used to determine the state of the I2C controller when a breakpoint is encountered in the HLL debugger.<br/> Note: This bit can be set independently of ST_EN value.</p> <p>FREE =<br/> 0: the I2C controller stops immediately after completion of the on-going bit transfer.<br/> Stopping the transfer is achieved by forcing the SCL line low.<br/> Note that in this case there will be no status register updates.</p> <p>FREE =<br/> 1: the I2C interface runs free.<br/> When Suspend indication will be asserted, there will be no accesses on the OCP Interface (the CPU is in debug mode) and consequently the FIFOs will reach full/empty state (according to RX or TX modes) and the I2C SDA line will be kept low.<br/> Note that the status registers will be updated, but no DMA, IRQ or WakeUp will be generated.</p> <p>The status registers likely to be updated in this mode are:<br/> I2C_IRQSTATUS_RAW.XRDY, I2C_IRQSTATUS_RAW.RRDY,<br/> I2C_IRQSTATUS_RAW.XUDF, I2C_IRQSTATUS_RAW.ROVR,<br/> I2C_IRQSTATUS_RAW.ARDY and I2C_IRQSTATUS_RAW.NACK.<br/> Value after reset is low.</p> <p>0x0 = Stop mode (on breakpoint condition). If Master mode, it stops after completion of the on-going bit transfer. In slave mode, it stops during the phase transfer when 1 byte is completely transmitted/received.</p> <p>0x1 = Free running mode</p> |
| 13-12 | TMODE    | R/W  | 0h    | <p>Test mode select.<br/> In normal functional mode (ST_EN = 0), these bits are don't care.<br/> They are always read as 00 and a write is ignored.<br/> In system test mode (ST_EN = 1), these bits can be set according to the following table to permit various system tests.<br/> Values after reset are low (2 bits).</p> <p>SCL counter test mode: in this mode, the SCL pin is driven with a permanent clock as if mastered with the parameters set in the I2C_PSC, I2C_SCLL, and I2C_SCLH registers.</p> <p>Loop back mode: in the master transmit mode only, data transmitted out of the I2C_DATA register (write action) is received in the same I2C_DATA register via an internal path through the FIFO buffer.<br/> The DMA and interrupt requests are normally generated if enabled.</p> <p>SDA/SCL IO mode: in this mode, the SCL IO and SDA IO are controlled via the I2C_SYSTEST [5:0] register bits.</p> <p>0x0 = Functional mode (default)<br/> 0x1 = Reserved</p> <p>0x2 = Test of SCL counters (SCLL, SCLH, PSC). SCL provides a permanent clock with master mode.</p> <p>0x3 = Loop back mode select + SDA/SCL IO mode select</p>   |
| 11    | SSB      | R/W  | 0h    | <p>Set status bits.<br/> Writing 1 into this bit also sets the 6 read/clear-only status bits contained in I2C_IRQSTATUS_RAW register (bits 5:0) to 1.<br/> Writing 0 into this bit doesn't clear status bits that are already set only writing 1 into a set status bit can clear it (see I2C_IRQSTATUS_RAW operation).<br/> This bit must be cleared prior attempting to clear a status bit.<br/> Value after reset is low.</p> <p>0x0 = No action.<br/> 0x1 = Set all interrupt status bits to 1.</p>   |
| 10-9  | Reserved | R    | 0h    |  |

**Table 21-33. I2C\_SYSTEST Register Field Descriptions (continued)**

| Bit | Field      | Type | Reset | Description  |
|-----|------------|------|-------|--|
| 8   | SCL_I_FUNC | R    | 0h    | SCL line input value (functional mode).<br>This read-only bit returns the logical state taken by the SCL line (either 1 or 0).<br>It is active both in functional and test mode.<br>Value after reset is low.<br>0x0 = Read 0 from SCL line<br>0x1 = Read 1 from SCL line  |
| 7   | SCL_O_FUNC | R    | 0h    | SCL line output value (functional mode).<br>This read-only bit returns the value driven by the module on the SCL line (either 1 or 0).<br>It is active both in functional and test mode.<br>Value after reset is low.<br>0x0 = Driven 0 on SCL line<br>0x1 = Driven 1 on SCL line  |
| 6   | SDA_I_FUNC | R    | 0h    | SDA line input value (functional mode).<br>This read-only bit returns the logical state taken by the SDA line (either 1 or 0).<br>It is active both in functional and test mode.<br>Value after reset is low.<br>0x0 = Read 0 from SDA line<br>0x1 = Read 1 from SDA line  |
| 5   | SDA_O_FUNC | R    | 0h    | SDA line output value (functional mode).<br>This read-only bit returns the value driven by the module on the SDA line (either 1 or 0).<br>It is active both in functional and test mode.<br>Value after reset is low.<br>0x0 = Driven 0 to SDA line<br>0x1 = Driven 1 to SDA line  |
| 4   | Reserved   | R    | 0h    |  |
| 3   | SCL_I      | R    | 0h    | SCL line sense input value.<br>In normal functional mode (ST_EN = 0), this read-only bit always reads 0.<br>In system test mode (ST_EN = 1 and TMODE = 11), this read-only bit returns the logical state taken by the SCL line (either 1 or 0).<br>Value after reset is low.<br>0x0 = Read 0 from SCL line<br>0x1 = Read 1 from SCL line   |
| 2   | SCL_O      | R    | 0h    | SCL line drive output value.<br>In normal functional mode (ST_EN = 0), this bit is don't care.<br>It always reads 0 and a write is ignored.<br>In system test mode (ST_EN = 1 and TMODE = 11), a 0 forces a low level on the SCL line and a 1 puts the I2C output driver to a high-impedance state.<br>Value after reset is low.<br>0x0 = Forces 0 on the SCL data line<br>0x1 = SCL output driver in high-impedance state |
| 1   | SDA_I      | R    | 0h    | SDA line sense input value.<br>In normal functional mode (ST_EN = 0), this read-only bit always reads 0.<br>In system test mode (ST_EN = 1 and TMODE = 11), this read-only bit returns the logical state taken by the SDA line (either 1 or 0).<br>Value after reset is low.<br>0x0 = Read 0 from SDA line<br>0x1 = Read 1 from SDA line   |

Table 21-33. I2C\_SYSTEST Register Field Descriptions (continued)

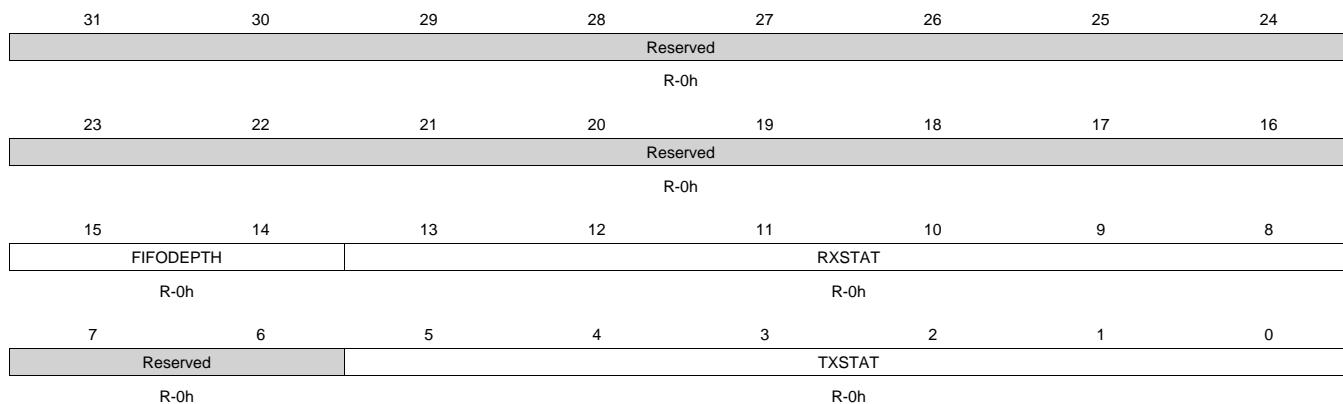
| Bit | Field | Type | Reset | Description   |
|-----|-------|------|-------|---|
| 0   | SDA_O | R    | 0h    | <p>SDA line drive output value.</p> <p>In normal functional mode (ST_EN = 0), this bit is don't care.</p> <p>It reads as 0 and a write is ignored.</p> <p>In system test mode (ST_EN = 1 and TMODE = 11), a 0 forces a low level on the SDA line and a 1 puts the I2C output driver to a high-impedance state.</p> <p>Value after reset is low.</p> <p>0x0 = Write 0 to SDA line</p> <p>0x1 = Write 1 to SDA line</p> |

### 21.4.1.26 I2C\_BUFSTAT Register (offset = C0h) [reset = 0h]

I2C\_BUFSTAT is shown in [Figure 21-41](#) and described in [Table 21-34](#).

This read-only register reflects the status of the internal buffers for the FIFO management (see the FIFO Management subsection).

**Figure 21-41. I2C\_BUFSTAT Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-34. I2C\_BUFSTAT Register Field Descriptions**

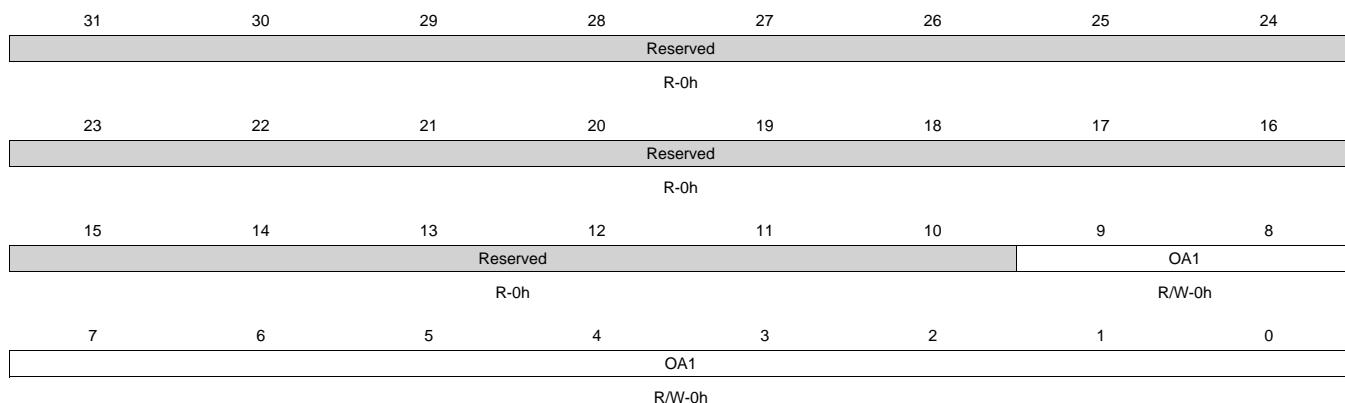
| Bit   | Field     | Type | Reset | Description  |
|-------|-----------|------|-------|--|
| 31-16 | Reserved  | R    | 0h    |  |
| 15-14 | FIFODEPTH | R    | 0h    | Internal FIFO buffers depth.<br>This read-only bit indicates the internal FIFO buffer depth.<br>Value after reset is given by the boundary module generic parameter.<br>0x0 = 8-bytes FIFO<br>0x1 = 16-bytes FIFO<br>0x2 = 32-bytes FIFO<br>0x3 = 64-bytes FIFO  |
| 13-8  | RXSTAT    | R    | 0h    | RX buffer status.<br>This read-only field indicates the number of bytes to be transferred from the FIFO at the end of the I2C transfer (when RDR is asserted). It corresponds to the level indication of the RX FIFO (number of written locations). Value after reset is 0.                            |
| 7-6   | Reserved  | R    | 0h    |  |
| 5-0   | TXSTAT    | R    | 0h    | TX buffer status.<br>This read-only field indicates the number of data bytes still left to be written in the TX FIFO (it's equal with the initial value of I2C_CNT.DCOUNT minus the number of data bytes already written in the TX FIFO through the OCP Interface). Value after reset is equal with 0. |

### 21.4.1.27 I2C\_OA1 Register (offset = C4h) [reset = 0h]

I2C\_OA1 is shown in [Figure 21-42](#) and described in [Table 21-35](#).

**CAUTION:** During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 1 - OA1).

**Figure 21-42. I2C\_OA1 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-35. I2C\_OA1 Register Field Descriptions**

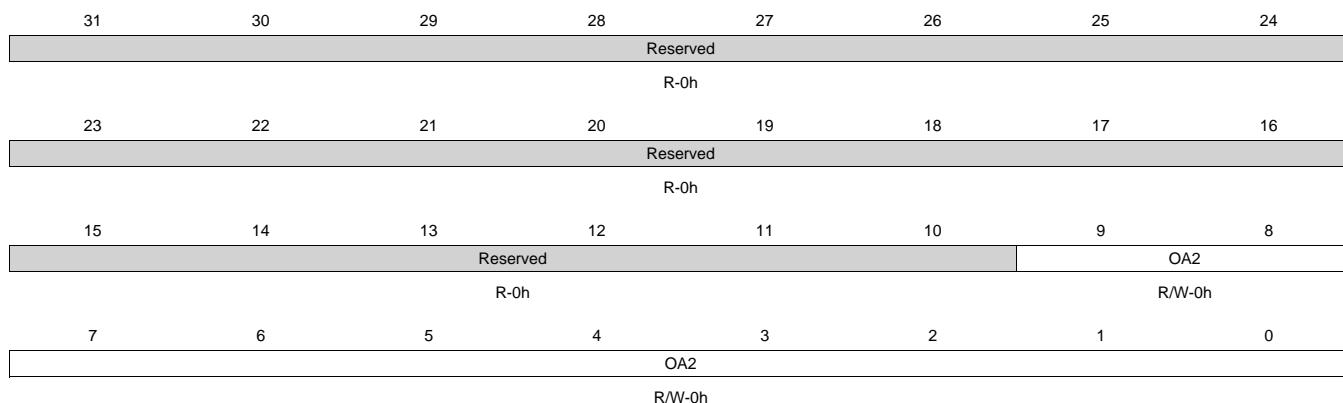
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-10 | Reserved | R    | 0h    |   |
| 9-0   | OA1      | R/W  | 0h    | Own address 1.<br>This field specifies either: A 10-bit address coded on OA1 [9:0] when XOA1 (Expand Own Address 1 - XOA1, I2C_CON[6]) is set to 1.<br>A 7-bit address coded on OA1 [6:0] when XOA1 (Expand Own Address 1 XOA1, I2C_CON[6]) is cleared to 0.<br>In this case, OA1 [9:7] bits must be cleared to 000 by application software.<br>Value after reset is low (all 10 bits). |

### 21.4.1.28 I2C\_OA2 Register (offset = C8h) [reset = 0h]

I2C\_OA2 is shown in [Figure 21-43](#) and described in [Table 21-36](#).

**CAUTION:** During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 2 - OA2).

**Figure 21-43. I2C\_OA2 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-36. I2C\_OA2 Register Field Descriptions**

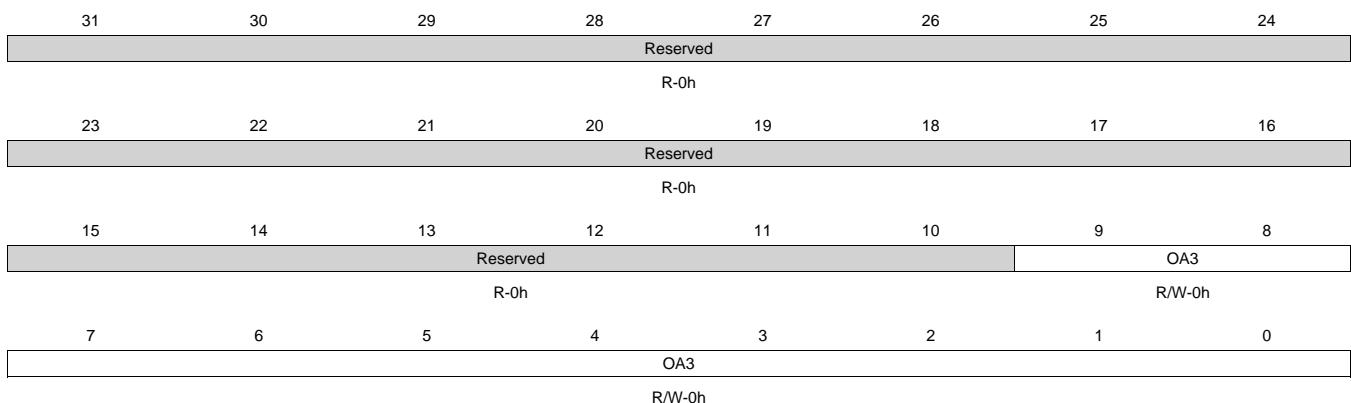
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-10 | Reserved | R    | 0h    |   |
| 9-0   | OA2      | R/W  | 0h    | Own address 2.<br>This field specifies either: A 10-bit address coded on OA2 [9:0] when XOA1 (Expand Own Address 2 - XOA2, I2C_CON[5]) is set to 1.<br>A 7-bit address coded on OA2 [6:0] when XOA2 (Expand Own Address 2 XOA2, I2C_CON[5]) is cleared to 0.<br>In this case, OA2 [9:7] bits must be cleared to 000 by application software.<br>Value after reset is low (all 10 bits). |

### 21.4.1.29 I2C\_OA3 Register (offset = CCh) [reset = 0h]

I2C\_OA3 is shown in [Figure 21-44](#) and described in [Table 21-37](#).

**CAUTION:** During an active transfer phase (between STT has been set to 1 and receiving of ARDY), no modification must be done in this register. Changing it may result in an unpredictable behavior. This register is used to specify the first alternative I2C 7-bit or 10-bit address (own address 3 - OA3).

**Figure 21-44. I2C\_OA3 Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-37. I2C\_OA3 Register Field Descriptions**

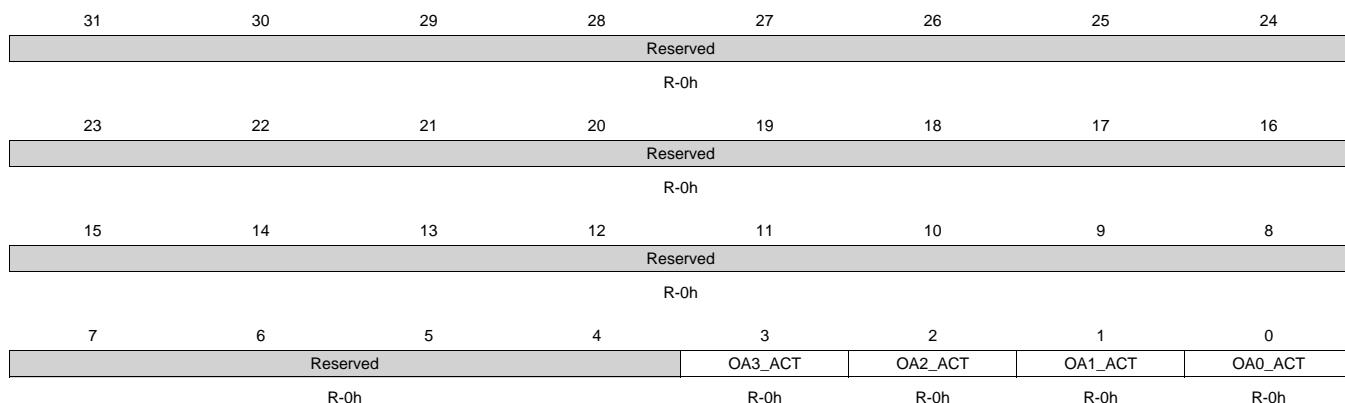
| Bit   | Field    | Type | Reset | Description   |
|-------|----------|------|-------|---|
| 31-10 | Reserved | R    | 0h    |   |
| 9-0   | OA3      | R/W  | 0h    | Own address 2.<br>This field specifies either: A 10-bit address coded on OA3 [9:0] when XOA3 (Expand Own Address 3 - XOA3, I2C_CON[4]) is set to 1.<br>A 7-bit address coded on OA3 [6:0] when XOA1 (Expand Own Address 3 XOA3, I2C_CON[4]) is cleared to 0.<br>In this case, OA3 [9:7] bits must be cleared to 000 by application software.<br>Value after reset is low (all 10 bits). |

### 21.4.1.30 I2C\_ACTOA Register (offset = D0h) [reset = 0h]

I2C\_ACTOA is shown in [Figure 21-45](#) and described in [Table 21-38](#).

This read-only register is used to indicate which one of the module's four own addresses the external master used when addressing the module. The CPU can read this register when the AAS indication was activated. The indication is cleared at the end of the transfer.

**Figure 21-45. I2C\_ACTOA Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-38. I2C\_ACTOA Register Field Descriptions**

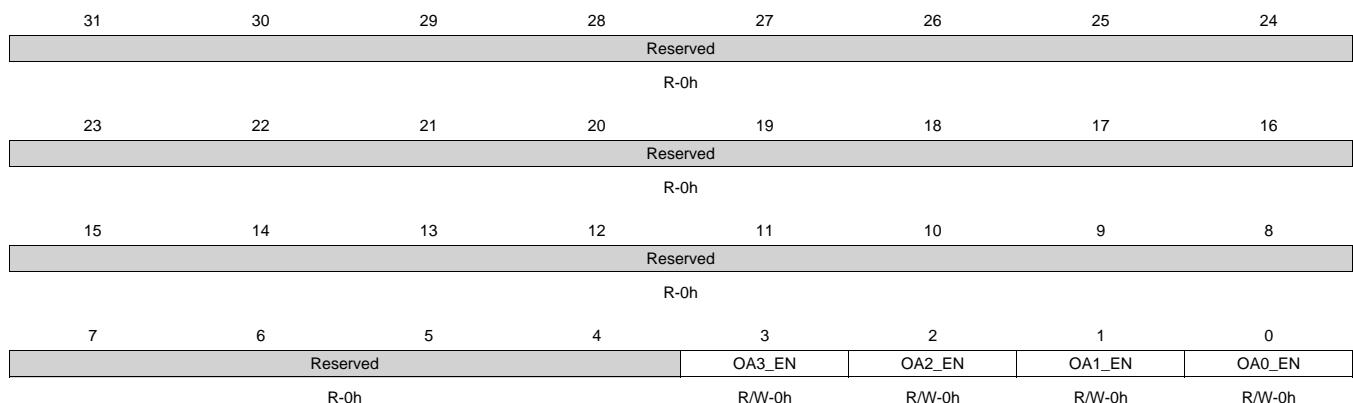
| Bit  | Field    | Type | Reset | Description  |
|------|----------|------|-------|--|
| 31-4 | Reserved | R    | 0h    |  |
| 3    | OA3_ACT  | R    | 0h    | Own address 3 active.<br>When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.<br>Value after reset is low.<br>0x0 = Own address inactive<br>0x1 = Own address active |
| 2    | OA2_ACT  | R    | 0h    | Own address 2 active.<br>When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.<br>Value after reset is low.<br>0x0 = Own address inactive<br>0x1 = Own address active |
| 1    | OA1_ACT  | R    | 0h    | Own address 1 active.<br>When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.<br>Value after reset is low.<br>0x0 = Own address inactive<br>0x1 = Own address active |
| 0    | OA0_ACT  | R    | 0h    | Own address 0 active.<br>When a bit location is set to 1 by the core, it signalizes to the Local Host that an external master using the corresponding own address addressed the module.<br>Value after reset is low.<br>0x0 = Own address inactive<br>0x1 = Own address active |

### 21.4.1.31 I2C\_SBLOCK Register (offset = D4h) [reset = 0h]

I2C\_SBLOCK is shown in [Figure 21-46](#) and described in [Table 21-39](#).

This read/write register controls the automatic blocking of I2C clock feature in slave mode. It is used for the Local Host to configure for which of the 4 own addresses, the core must block the I2C clock (keep SCL line low) right after the Address Phase, when it is addressed as a slave.

**Figure 21-46. I2C\_SBLOCK Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 21-39. I2C\_SBLOCK Register Field Descriptions**

| Bit  | Field    | Type | Reset | Description   |
|------|----------|------|-------|---|
| 31-4 | Reserved | R    | 0h    |   |
| 3    | OA3_EN   | R/W  | 0h    | Enable I2C clock blocking for own address 3.<br>When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase.<br>For releasing the I2C clock the CPU must write 0 in the corresponding field.<br>Value after reset is low.<br>0x0 = I2C clock released<br>0x1 = I2C clock blocked |
| 2    | OA2_EN   | R/W  | 0h    | Enable I2C clock blocking for own address 2.<br>When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase.<br>For releasing the I2C clock the CPU must write 0 in the corresponding field.<br>Value after reset is low.<br>0x0 = I2C clock released<br>0x1 = I2C clock blocked |
| 1    | OA1_EN   | R/W  | 0h    | Enable I2C clock blocking for own address 1.<br>When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase.<br>For releasing the I2C clock the CPU must write 0 in the corresponding field.<br>Value after reset is low.<br>0x0 = I2C clock released<br>0x1 = I2C clock blocked |

**Table 21-39. I2C\_SBLOCK Register Field Descriptions (continued)**

| Bit | Field  | Type | Reset | Description   |
|-----|--------|------|-------|---|
| 0   | OA0_EN | R/W  | 0h    | <p>Enable I2C clock blocking for own address 0.</p> <p>When the CPU sets a bit location to 1, if an external master using the corresponding own address addresses the core, the core will block the I2C clock right after the address phase.</p> <p>For releasing the I2C clock the CPU must write 0 in the corresponding field.</p> <p>Value after reset is low.</p> <p>0x0 = I2C clock released<br/>0x1 = I2C clock blocked</p> |

## **Multichannel Audio Serial Port (McASP)**

This chapter describes the McASP of the device.

| Topic                                    | Page        |
|--|-------------|
| <b>22.1 Introduction .....</b>           | <b>3751</b> |
| <b>22.2 Integration .....</b>            | <b>3753</b> |
| <b>22.3 Functional Description .....</b> | <b>3755</b> |
| <b>22.4 McASP Registers .....</b>        | <b>3808</b> |

## 22.1 Introduction

### 22.1.1 Purpose of the Peripheral

The multichannel audio serial port (McASP) functions as a general-purpose audio serial port optimized for the needs of multichannel audio applications. The McASP is useful for time-division multiplexed (TDM) stream, Inter-Integrated Sound (I2S) protocols, and intercomponent digital audio interface transmission (DIT). The McASP consists of transmit and receive sections that may operate synchronized, or completely independently with separate master clocks, bit clocks, and frame syncs, and using different transmit modes with different bit-stream formats. The McASP module also includes serializers that can be individually enabled to either transmit or receive.

### 22.1.2 Features

Features of the McASP include:

- Two independent clock generator modules for transmit and receive.
  - Clocking flexibility allows the McASP to receive and transmit at different rates. For example, the McASP can receive data at 48 kHz but output up-sampled data at 96 kHz or 192 kHz.
- Independent transmit and receive modules, each includes:
  - Programmable clock and frame sync generator.
  - TDM streams from 2 to 32, and 384 time slots.
  - Support for time slot sizes of 8, 12, 16, 20, 24, 28, and 32 bits.
  - Data formatter for bit manipulation.
- Individually assignable serial data pins (up to 6 pins).
- Glueless connection to audio analog-to-digital converters (ADC), digital-to-analog converters (DAC), codec, digital audio interface receiver (DIR), and S/PDIF transmit physical layer components.
- Wide variety of I2S and similar bit-stream format.
- Integrated digital audio interface transmitter (DIT) supports (up to 10 transmit pins):
  - S/PDIF, IEC60958-1, AES-3 formats.
  - Enhanced channel status/user data RAM.
- 384-slot TDM with external digital audio interface receiver (DIR) device.
  - For DIR reception, an external DIR receiver integrated circuit should be used with I2S output format and connected to the McASP receive section.
- Extensive error checking and recovery.
  - Transmit underruns and receiver overruns due to the system not meeting real-time requirements.
  - Early or late frame sync in TDM mode.
  - Out-of-range high-frequency master clock for both transmit and receive.
  - External error signal coming into the AMUTEIN input.
  - DMA error due to incorrect programming.

### 22.1.3 Protocols Supported

The McASP supports a wide variety of protocols.

- Transmit section supports:
  - Wide variety of I2S and similar bit-stream formats.
  - TDM streams from 2 to 32 time slots.
  - S/PDIF, IEC60958-1, AES-3 formats.
- Receive section supports:
  - Wide variety of I2S and similar bit-stream formats.
  - TDM streams from 2 to 32 time slots.
  - TDM stream of 384 time slots specifically designed for easy interface to external digital interface

receiver (DIR) device transmitting DIR frames to McASP using the I<sub>2</sub>S protocol (one time slot for each DIR subframe).

The transmit and receive sections may each be individually programmed to support the following options on the basic serial protocol:

- Programmable clock and frame sync polarity (rising or falling edge): ACLKR/X, AHCLKR/X, and AFSR/X.
- Slot length (number of bits per time slot): 8, 12, 16, 20, 24, 28, 32 bits supported.
- Word length (bits per word): 8, 12, 16, 20, 24, 28, 32 bits; always less than or equal to the time slot length.
- First-bit data delay: 0, 1, 2 bit clocks.
- Left/right alignment of word inside slot.
- Bit order: MSB first or LSB first.
- Bit mask/pad/rotate function.
  - Automatically aligns data internally in either Q31 or integer formats.
  - Automatically masks nonsignificant bits (sets to 0, 1, or extends value of another bit).

In DIT mode for McASP, additional features of the transmitter are:

- Transmit-only mode 384 time slots (subframe) per frame.
- Bi-phase encoded 3.3 V output.
- Support for consumer and professional applications.
- Channel status RAM (384 bits).
- User data RAM (384 bits).
- Separate valid bit (V) for subframe A, B.

In I<sub>2</sub>S mode, the transmit and receive sections can support simultaneous transfers on up to all serial data pins operating as 192 kHz stereo channels.

In DIT mode, the transmitter can support a 192 kHz frame rate (stereo) on up to all serial data pins simultaneously (note that the internal bit clock for DIT runs two times faster than the equivalent bit clock for I<sub>2</sub>S mode, due to the need to generate Biphase Mark Encoded Data).

#### 22.1.4 Unsupported McASP Features

The unsupported McASP features in this device include the following:

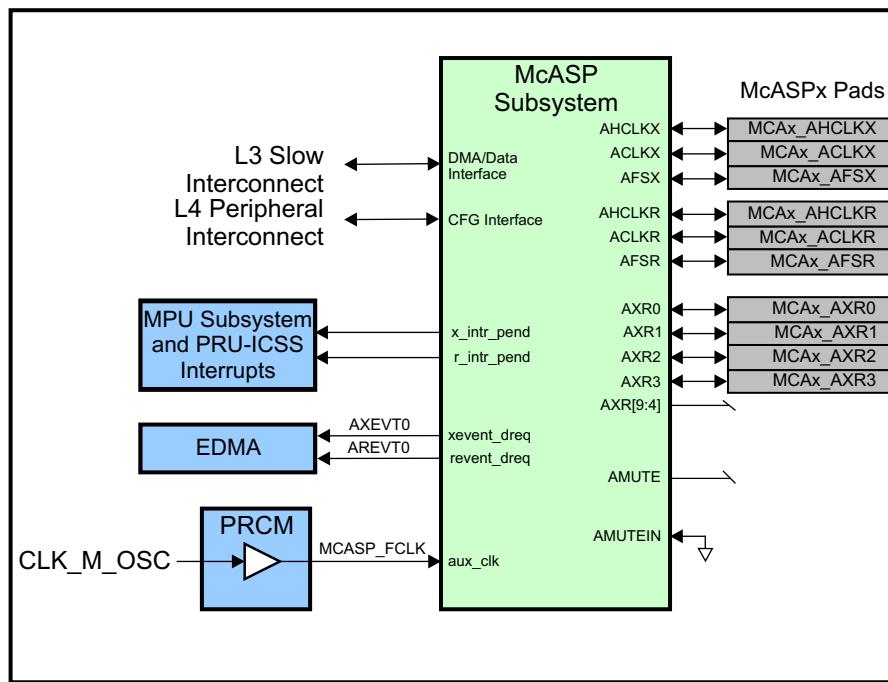
- AXR[9:4] — Signals are not pinned out
- AMUTE — Not connected
- AMUTEIN — Not connected

## 22.2 Integration

The device contains two instantiations of the McASP subsystem: McASP0 and McASP1. The McASP subsystem includes a McASP peripheral, and transmit/receive buffers.

Each McASP is configured with four serializers.

**Figure 22-1. McASP0–1 Integration**



### 22.2.1 McASP Connectivity Attributes

The general connectivity attributes for the McASP modules are summarized in [Table 22-1](#)

**Table 22-1. McASP Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L3S_GCLK (OCP Clock)<br>PD_PER_MCASP_FCLK (Aux Clock)   |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 1 Transmit Interrupt per instance<br>x_intr_pend - to MPU Subsystem (MCATXINTx) and PRU-ICSS (mcasp_x_intr_pend)<br>1 Receive Interrupt<br>r_intr_pend - to MPU Subsystem (MCARXINTx) and PRU-ICSS (mcasp_r_intr_pend) |
| DMA Requests        | 2 DMA requests per instance to EDMA (Transmit: AXEVTx, Receive: AREVTx)  |
| Physical Address    | L3 Slow slave port (data)<br>L4 Peripheral slave port (CFG)  |

## 22.2.2 McASP Clock and Reset Management

The McASP module uses functional clocks either generated internally (master mode) or supplied from its serial interface (slave mode). The internal interconnect interface clock is used for the module internal OCP interface. Internal registers select the source of the functional clocks and the applied divider ratio.

**Table 22-2. McASP Clock Signals**

| Clock Signal               | Max Freq | Reference / Source | Comments                       |
|----------------------------|----------|--------------------|--------------------------------|
| ocp_clk<br>Interface clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l3s_gclk<br>From PRCM   |
| auxclk<br>Functional clock | 26 MHz   | CLK_M_OSC          | pd_per_mcasp_fclk<br>From PRCM |

## 22.2.3 McASP Pin List

The McASP external interface signals are shown in [Table 22-3](#).

**Table 22-3. McASP Pin List**

| Pin             | Type | Description                |
|-----------------|------|----------------------------|
| McASPx_AXR[3:0] | I/O  | Audio transmit/receive pin |
| McASPx_ACLKX    | I/O  | Transmit clock             |
| McASPx_FSX      | I/O  | Frame synch for transmit   |
| McASPx_AHCLKX   | I/O  | High speed transmit clock  |
| McASPx_ACLKR    | I/O  | Receive clock              |
| McASPx_FSR      | I/O  | Frame synch for receive    |
| McASPx_AHCLKR   | I/O  | High speed receive clock   |

## 22.3 Functional Description

### 22.3.1 Overview

Figure 22-2 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators, error-checking logic, and up to four serial data pins. See the device-specific data manual for the number of data pins available on your device.

All the McASP pins on the device may be individually programmed as general-purpose I/O (GPIO) if they are not used for serial port functions.

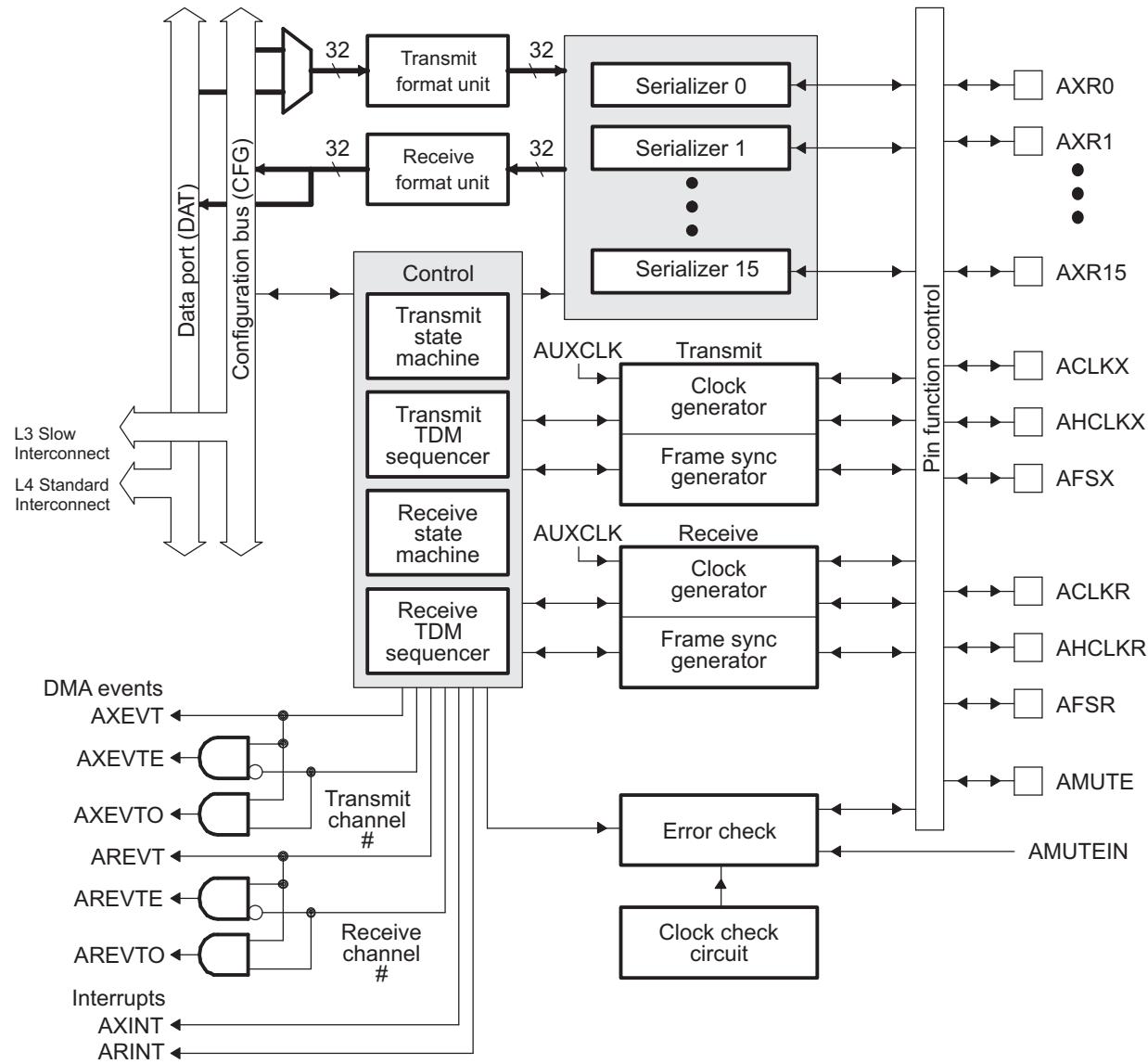
The McASP includes the following pins:

- Serializers;
  - Data pins AXRn: Up to four pins.
- Transmit clock generator:
  - AHCLKX: McASP transmit high-frequency master clock.
  - ACLKX: McASP transmit bit clock.
- Transmit Frame Sync Generator;
  - AFSX: McASP transmit frame sync or left/right clock (LRCLK).
- Receive clock generator;
  - AHCLKR: McASP receive high-frequency master clock.
  - ACLKR: McASP receive bit clock.
- Receive Frame Sync Generator;
  - AFSR: McASP receive frame sync or left/right clock (LRCLK).
- Mute in/out;
  - AMUTEIN: McASP mute input (from external device).
  - AMUTE: McASP mute output.
  - Data pins AXRn.

### 22.3.2 Functional Block Diagram

Figure 22-2 shows the major blocks of the McASP. The McASP has independent receive/transmit clock generators and frame sync generators.

**Figure 22-2. McASP Block Diagram**

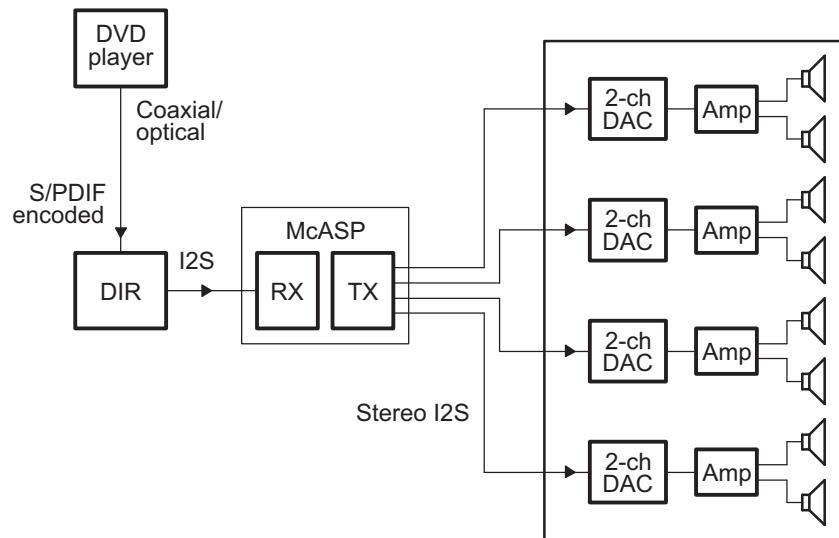


- A McASP has 6 serial data pins.
- B AMUTEIN is not a dedicated McASP pin, but typically comes from one of the external interrupt pins.

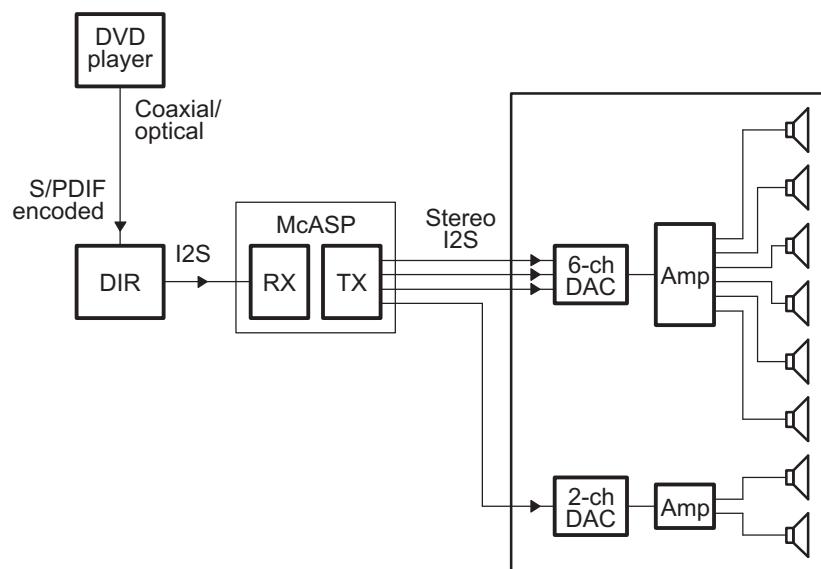
### 22.3.2.1 System Level Connections

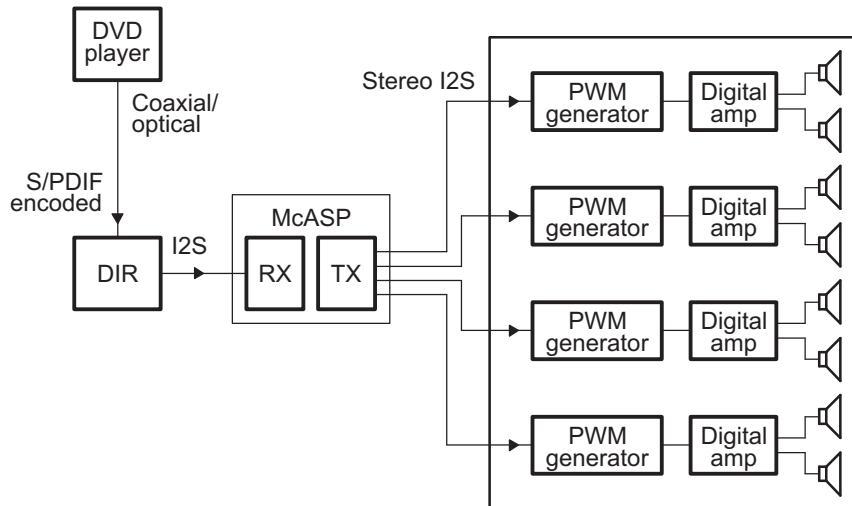
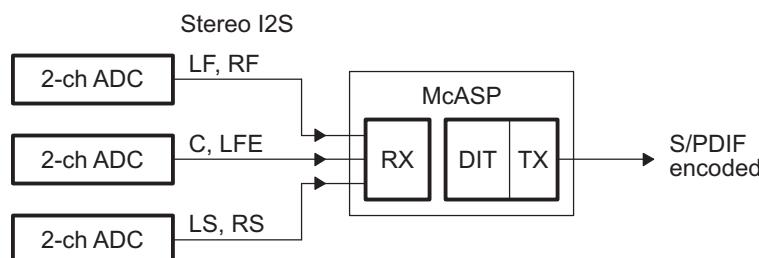
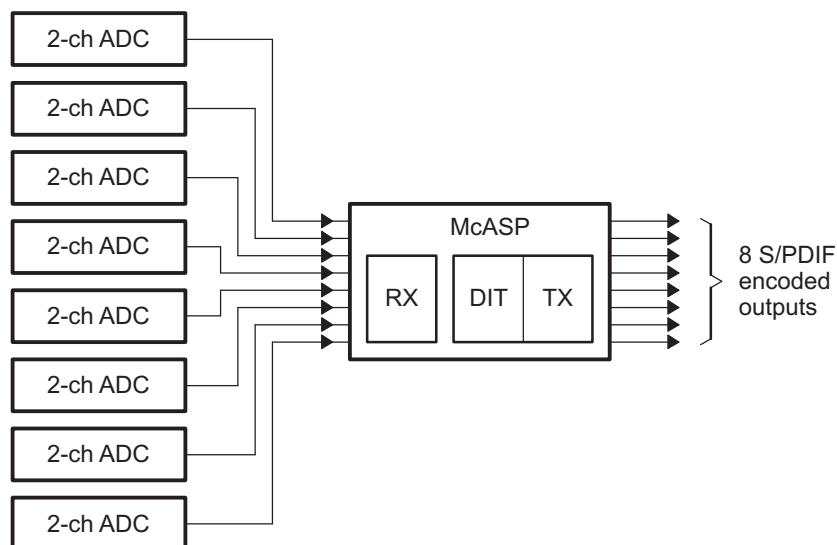
Figure 22-3 through Figure 22-7 show examples of McASP usage in digital audio encoder/decoder systems.

**Figure 22-3. McASP to Parallel 2-Channel DACs**



**Figure 22-4. McASP to 6-Channel DAC and 2-Channel DAC**



**Figure 22-5. McASP to Digital Amplifier**

**Figure 22-6. McASP as Digital Audio Encoder**

**Figure 22-7. McASP as 16 Channel Digital Processor**


### 22.3.3 Industry Standard Compliance Statement

The McASP supports the following industry standard interfaces.

#### 22.3.3.1 TDM Format

The McASP transmitter and receiver support the multichannel, synchronous time-division-multiplexed (TDM) format via the TDM transfer mode. Within this transfer mode, a wide variety of serial data formats are supported, including formats compatible with devices using the Inter-Integrated Sound (I2S) protocol. This section briefly discusses the TDM format and the I2S protocol.

##### 22.3.3.1.1 TDM Format

The TDM format is typically used when communicating between integrated circuit devices on the same printed circuit board or on another printed circuit board within the same piece of equipment. For example, the TDM format is used to transfer data between the processor and one or more analog-to-digital converter (ADC), digital-to-analog converter (DAC), or S/PDIF receiver (DIR) devices.

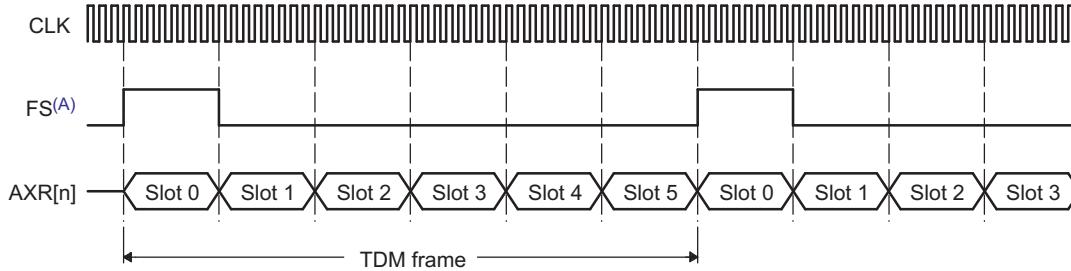
The TDM format consists of three components in a basic synchronous serial transfer: the clock, the data, and the frame sync. In a TDM transfer, all data bits ( $AXR_n$ ) are synchronous to the serial clock (ACLKX or ACLKR). The data bits are grouped into words and slots (as defined in [Section 22.3.4](#)). The "slots" are also commonly referred to as "time slots" or "channels" in TDM terminology. A frame consists of multiple slots (or channels). Each TDM frame is defined by the frame sync signal (AFSX or AFSR). Data transfer is continuous and periodic, since the TDM format is most commonly used to communicate with data converters that operate at a fixed sample rate.

There are no delays between slots. The last bit of slot N is followed immediately on the next serial clock cycle with the first bit of slot N + 1, and the last bit of the last slot is followed immediately on the next serial clock with the first bit of the first slot. However, the frame sync may be offset from the first bit of the first slot with a 0, 1, or 2-cycle delay.

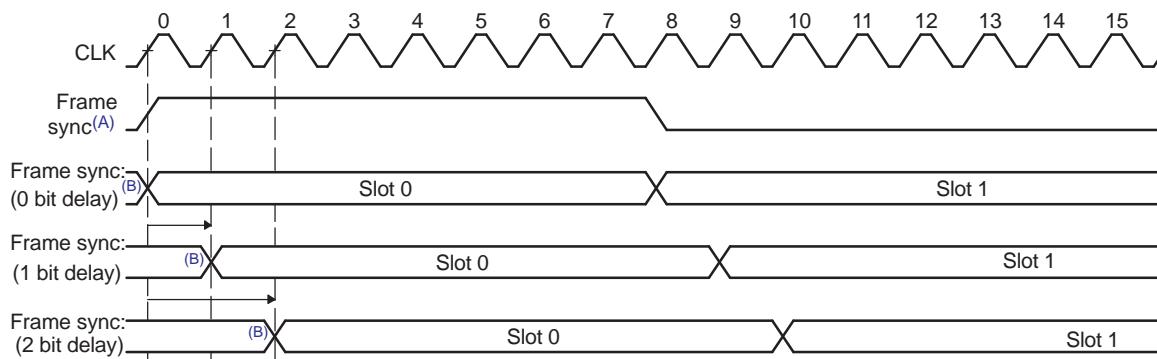
It is required that the transmitter and receiver in the system agree on the number of bits per slot, since the determination of a slot boundary is not made by the frame sync signal (although the frame sync marks the beginning of slot 0 and the beginning of a new frame).

[Figure 22-8](#) shows the TDM format. [Figure 22-9](#) shows the different bit delays from the frame sync.

**Figure 22-8. TDM Format—6 Channel TDM Example**



A FS duration of slot is shown. FS duration of single bit is also supported.

**Figure 22-9. TDM Format Bit Delays from Frame Sync**


- A FS duration of slot is shown. FS duration of single bit is also supported.  
 B Last bit of last slot of previous frame. No gap between this bit and the first bit of slot 0 is allowed.

In a typical audio system, one frame of data is transferred during each data converter sample period  $f_s$ . To support multiple channels, the choices are to either include more time slots per frame (thus operating with a higher bit clock rate), or to use additional data pins to transfer the same number of channels (thus operating with a slower bit clock rate).

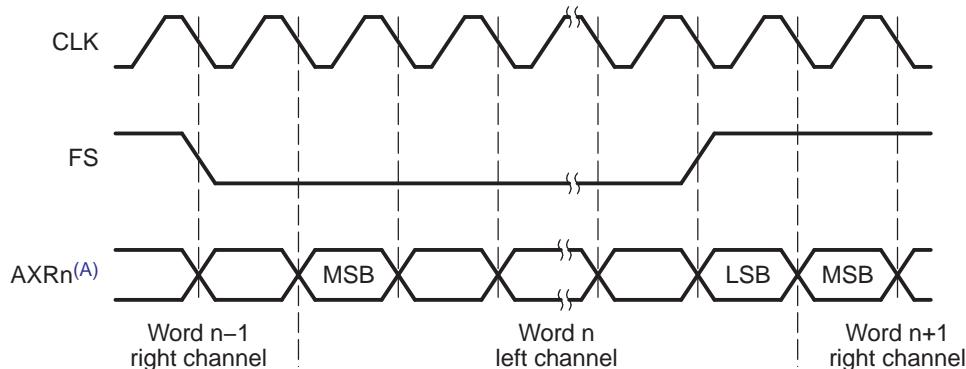
For example, a particular six channel DAC may be designed to transfer over a single serial data pin AXRn as shown in [Figure 22-8](#). In this case the serial clock must run fast enough to transfer a total of 6 channels within each frame period. Alternatively, a similar six channel DAC may be designed to use three serial data pins AXR[0,1,2], transferring two channels of data on each pin during each sample period. In the latter case, if the sample period remains the same, the serial clock can run three times slower than the former case. The McASP is flexible enough to support either type of DAC.

#### 22.3.3.1.2 Inter-Integrated Sound (I2S) Format

The inter-integrated sound (I2S) format is used extensively in audio interfaces. The TDM transfer mode of the McASP supports the I2S format when configured to 2 slots per frame.

I2S format is specifically designed to transfer a stereo channel (left and right) over a single data pin AXRn. "Slots" are also commonly referred to as "channels". The frame width duration in the I2S format is the same as the slot size. The frame signal is also referred to as "word select" in the I2S format. [Figure 22-10](#) shows the I2S protocol.

The McASP supports transfer of multiple stereo channels over multiple AXRn pins.

**Figure 22-10. Inter-Integrated Sound (I2S) Format**


- A 1 to 6 data pins may be supported.

### 22.3.3.2 S/PDIF Coding Format

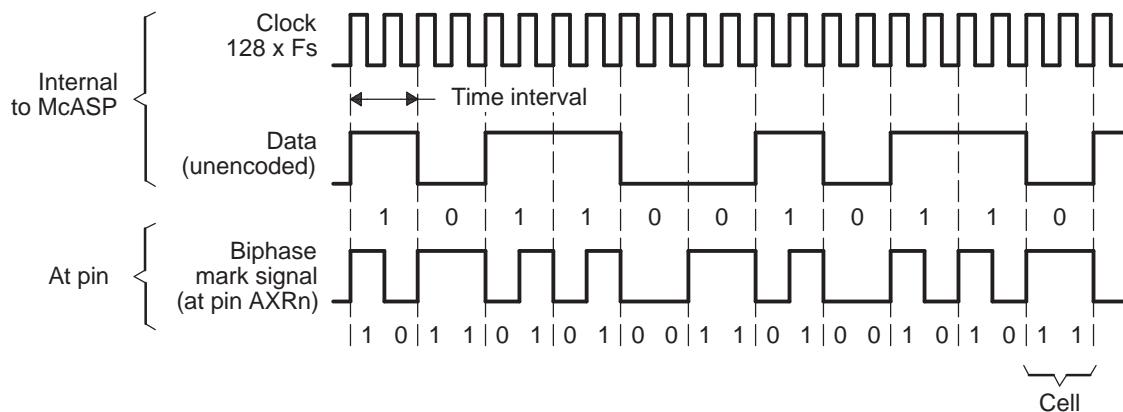
The McASP transmitter supports the S/PDIF format with 3.3V biphase-mark encoded output. The S/PDIF format is supported by the digital audio interface transmit (DIT) transfer mode of the McASP. This section briefly discusses the S/PDIF coding format.

#### 22.3.3.2.1 Biphase-Mark Code (BMC)

In S/PDIF format, the digital signal is coded using the biphase-mark code (BMC). The clock, frame, and data are embedded in only one signal—the data pin AXRn. In the BMC system, each data bit is encoded into two logical states (00, 01, 10, or 11) at the pin. These two logical states form a cell. The duration of the cell, which equals to the duration of the data bit, is called a time interval. A logical 1 is represented by two transitions of the signal within a time interval, which corresponds to a cell with logical states 01 or 10. A logical 0 is represented by one transition within a time interval, which corresponds to a cell with logical states 00 or 11. In addition, the logical level at the start of a cell is inverted from the level at the end of the previous cell. [Figure 22-11](#) and [Table 22-4](#) show how data is encoded to the BMC format.

As shown in [Figure 22-11](#), the frequency of the clock is twice the unencoded data bit rate. In addition, the clock is always programmed to  $128 \times f_s$ , where  $f_s$  is the sample rate (see [Section 22.3.3.2.3](#) for details on how this clock rate is derived based on the S/PDIF format). The device receiving in S/PDIF format can recover the clock and frame information from the BMC signal.

**Figure 22-11. Biphase-Mark Code (BMC)**



**Table 22-4. Biphase-Mark Encoder**

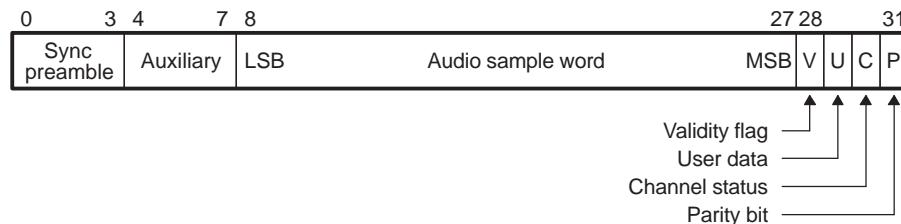
| Data (Unencoded) | Previous State at Pin AXRn |   | BMC-Encoded Cell Output at AXRn |
|------------------|----------------------------|---|---------------------------------|
|                  | 0                          | 1 |                                 |
| 0                | 0                          |   | 11                              |
| 0                | 1                          |   | 00                              |
| 1                | 0                          |   | 10                              |
| 1                | 1                          |   | 01                              |

### 22.3.3.2.2 Subframe Format

Every audio sample transmitted in a subframe consists of 32 S/PDIF time intervals (or cells), numbered from 0 to 31. [Figure 22-12](#) shows a subframe.

- **Time intervals 0-3** carry one of the three permitted preambles to signify the type of audio sample in the current subframe. The preamble is *not* encoded in BMC format, and therefore the preamble code can contain more than two consecutive 0 or 1 logical states in a row. See [Table 22-5](#).
- **Time intervals 4-27** carry the audio sample word in linear 2s-complement representation. The most-significant bit (MSB) is carried by time interval 27. When a 24-bit coding range is used, the least-significant bit (LSB) is in time interval 4. When a 20-bit coding range is used, time intervals 8-27 carry the audio sample word with the LSB in time interval 8. Time intervals 4-7 may be used for other applications and are designated auxiliary sample bits.
- If the source provides fewer bits than the interface allows (either 20 or 24), the unused LSBs are set to logical 0. For a nonlinear PCM audio application or a data application, the main data field may carry any other information.
- **Time interval 28** carries the validity bit (V) associated with the main data field in the subframe.
- **Time interval 29** carries the user data channel (U) associated with the main data field in the subframe.
- **Time interval 30** carries the channel status information (C) associated with the main data field in the subframe. The channel status indicates if the data in the subframe is digital audio or some other type of data.
- **Time interval 31** carries a parity bit (P) such that time intervals 4-31 carry an even number of 1s and an even number of 0s (even parity). As shown in [Table 22-5](#), the preambles (time intervals 0-3) are also defined with even parity.

**Figure 22-12. S/PDIF Subframe Format**



**Table 22-5. Preamble Codes**

| Preamble Code <sup>(1)</sup> | Previous Logical State | Logical States on pin AXRn <sup>(2)</sup> | Description                     |
|------------------------------|------------------------|---|---------------------------------|
| B (or Z)                     | 0                      | 1110 1000                                 | Start of a block and subframe 1 |
| M (or X)                     | 0                      | 1110 0010                                 | Subframe 1                      |
| W (or Y)                     | 0                      | 1110 0100                                 | Subframe 2                      |

<sup>(1)</sup> Historically, preamble codes are referred to as B, M, W. For use in professional applications, preambles are referred to as Z, X, Y, respectively.

<sup>(2)</sup> The preamble is not BMC encoded. Each logical state is synchronized to the serial clock. These 8 logical states make up time slots (cells) 0 to 3 in the S/PDIF stream.

As shown in [Table 22-5](#), the McASP DIT only generates one polarity of preambles and it assumes the previous logical state to be 0. This is because the McASP assures an even-polarity encoding scheme when transmitting in DIT mode. If an underrun condition occurs, the DIT resynchronizes to the correct logic level on the AXRn pin before continuing with the next transmission.

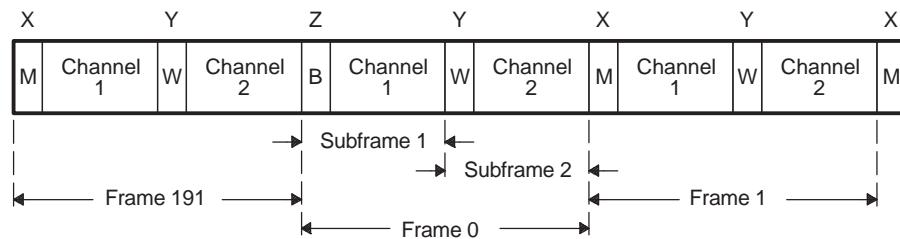
### 22.3.3.2.3 Frame Format

An S/PDIF frame is composed of two subframes (Figure 22-13). For linear coded audio applications, the rate of frame transmission normally corresponds exactly to the source sampling frequency  $f_s$ . The S/PDIF format clock rate is therefore  $128 \times f_s$  ( $128 = 32$  cells/subframe  $\times 2$  clocks/cell  $\times 2$  subframes/sample). For example, for an S/PDIF stream at a 192 kHz sampling frequency, the serial clock is  $128 \times 192$  kHz = 24.58 MHz.

In 2-channel operation mode, the samples taken from both channels are transmitted by time multiplexing in consecutive subframes. Both subframes contain valid data. The first subframe (**left** or **A** channel in stereophonic operation and **primary** channel in monophonic operation) normally starts with preamble M. However, the preamble of the first subframe changes to preamble B once every 192 frames to identify the start of the block structure used to organize the channel status information. The second subframe (**right** or **B** channel in stereophonic operation and **secondary** channel in monophonic operation) always starts with preamble W.

In single-channel operation mode in a professional application, the frame format is the same as in the 2-channel mode. Data is carried in the first subframe and may be duplicated in the second subframe. If the second subframe is not carrying duplicate data, cell 28 (validity bit) is set to logical 1.

**Figure 22-13. S/PDIF Frame Format**



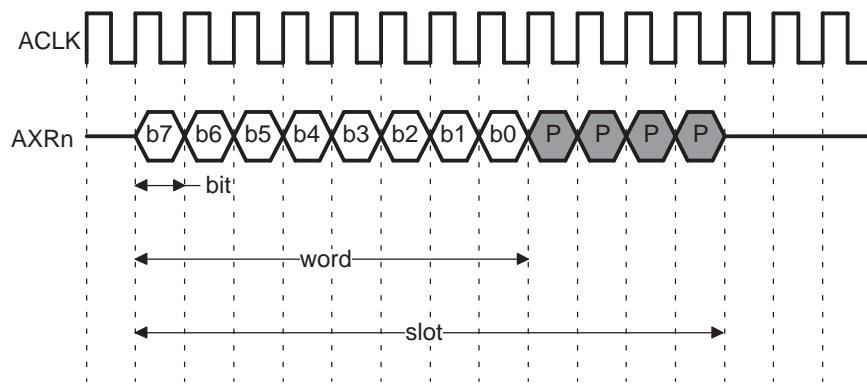
### 22.3.4 Definition of Terms

The serial bit stream transmitted or received by the McASP is a long sequence of 1s and 0s, either output or input on one of the audio transmit/receive pins (AXRn). However, the sequence has a hierarchical organization that can be described in terms of frames of data, slots, words, and bits.

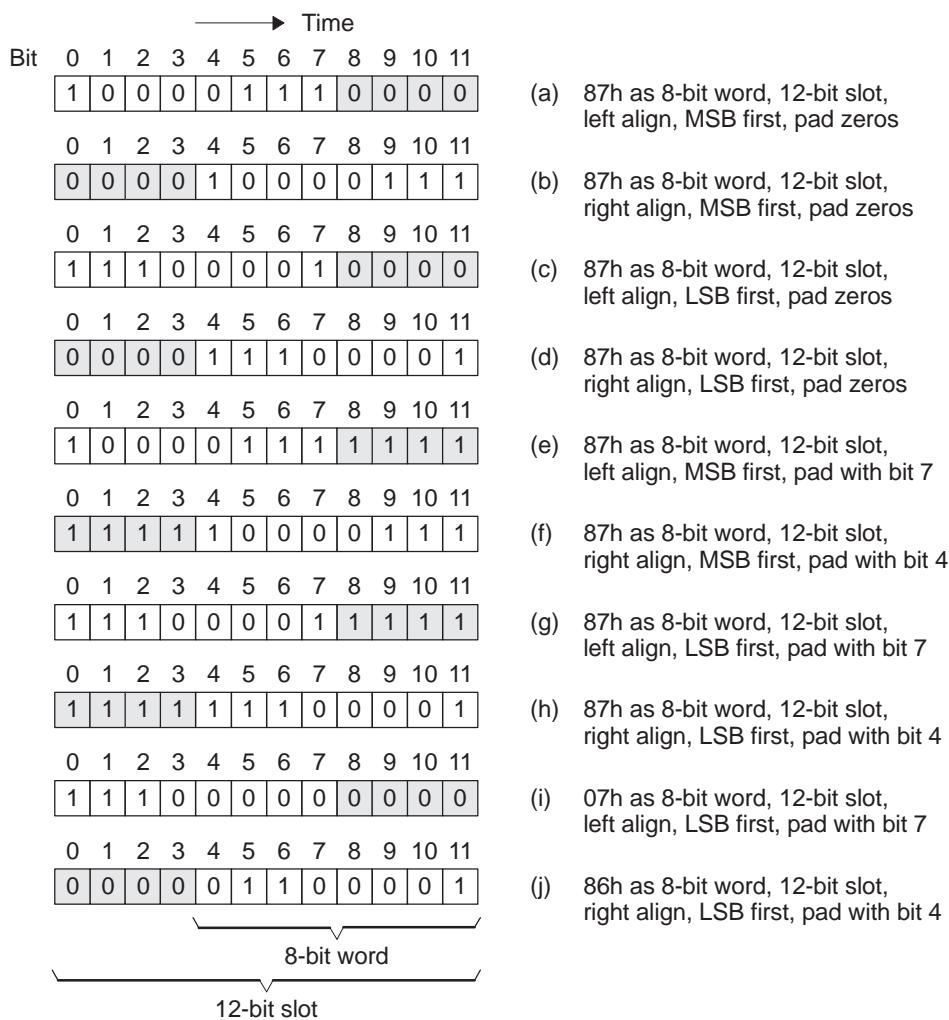
A basic synchronous serial interface consists of three important components: clock, frame sync, and data. Figure 22-14 shows two of the three basic components—the clock (ACLK) and the data (AXRn).

Figure 22-14 does not specify whether the clock is for transmit (ACLKX) or receive (ACLKR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses ACLKX as the serial clock, and the receiver uses ACLKR as the serial clock. Optionally, the receiver can use ACLKX as the serial clock when the transmitter and receiver of the McASP are configured to operate synchronously.

|             |   |
|-------------|---|
| <b>Bit</b>  | A bit is the smallest entity in the serial data stream. The beginning and end of each bit is marked by an edge of the serial clock. The duration of a bit is a serial clock period. A 1 is represented by a logic high on the AXRn pin for the entire duration of the bit. A 0 is represented by a logic low on the AXRn pin for the entire duration of the bit.  |
| <b>Word</b> | A word is a group of bits that make up the data being transferred between the processor and the external device. Figure 22-14 shows an 8-bit word.  |
| <b>Slot</b> | A slot consists of the bits that make up the word, and may consist of additional bits used to pad the word to a convenient number of bits for the interface between the processor and the external device. In Figure 22-14, the audio data consists of only 8 bits of useful data (8-bit word), but it is padded with 4 zeros (12-bit slot) to satisfy the desired protocol in interfacing to an external device. Within a slot, the bits may be shifted in/out of the McASP on the AXRn pin either MSB or LSB first. When the word size is smaller than the slot size, the word may be aligned to the left (beginning) of the slot or to the right (end) of the slot. The additional bits in the slot not belonging to the word may be padded with 0, 1, or with one of the bits (the MSB or the LSB typically) from the data word. These options are shown in Figure 22-15. |

**Figure 22-14. Definition of Bit, Word, and Slot**


- (1) b7:b0 - bits. Bits b7 to b0 form a word.
- (2) P - pad bits. Bits b7 to b0, together with the four pad bits, form a slot.
- (3) In this example, the data is transmitted MSB first, left aligned.

**Figure 22-15. Bit Order and Word Alignment Within a Slot Examples**


1 Unshaded: bit belongs to word

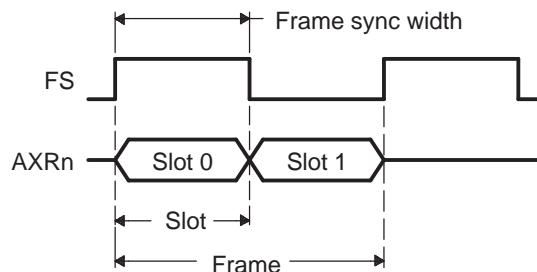
1 Shaded: bit is a pad bit

The third basic element of a synchronous serial interface is the frame synchronization signal, also referred to as frame sync in this document.

**Frame** A frame contains one or multiple slots, as determined by the desired protocol. [Figure 22-16](#) shows an example frame of data and the frame definitions. [Figure 22-16](#) does not specify whether the frame sync (FS) is for transmit (AFSX) or receive (AFSR) because the definitions of terms apply to both receive and transmit interfaces. In operation, the transmitter uses AFSX and the receiver uses AFSR. Optionally, the receiver can use AFSX as the frame sync when the transmitter and receiver of the McASP are configured to operate synchronously.

This section only shows the generic definition of the frame sync. See [Section 22.3.3](#) and [Section 22.3.8.1](#) for details on the frame sync formats required for the different transfer modes and protocols (burst mode, TDM mode and I2S format, DIT mode and S/PDIF format).

**Figure 22-16. Definition of Frame and Frame Sync Width**



(1) In this example, there are two slots in a frame, and FS duration of slot length is shown.

Other terms used throughout the document:

|                          |  |
|--------------------------|--|
| <b>TDM</b>               | Time-division multiplexed. See <a href="#">Section 22.3.3.1</a> for details on the TDM protocol.   |
| <b>DIR</b>               | Digital audio interface receive. The McASP does not natively support receiving in the S/PDIF format. The McASP supports I2S format output by an external DIR device. |
| <b>DIT</b>               | Digital audio interface transmit. The McASP supports transmitting in S/PDIF format on up to all data pins configured as outputs.                                     |
| <b>I2S</b>               | Inter-Integrated Sound protocol, commonly used on audio interfaces. The McASP supports the I2S protocol as part of the TDM mode (when configured as a 2-slot frame). |
| <b>Slot or Time Slot</b> | For TDM format, the term time slot is interchangeable with the term slot defined in this section. For DIT format, a McASP time slot corresponds to a DIT subframe.   |

### 22.3.5 Clock and Frame Sync Generators

The McASP clock generators are able to produce two independent clock zones: transmit and receive clock zones. The serial clock generators may be programmed independently for the transmit section and the receive section, and may be completely asynchronous to each other. The serial clock (clock at the bit rate) may be sourced:

- Internally - by passing through two clock dividers off the internal clock source (AUXCLK).
- Externally - directly from ACLKR/X pin.
- Mixed - an external high-frequency clock is input to the McASP on either the AHCLKX or AHCLKR pins, and divided down to produce the bit rate clock.

In the internal/mixed cases, the bit rate clock is generated internally and should be driven out on the ACLKX (for transmit) or ACLKR (for receive) pins. In the internal case, an internally-generated high-frequency clock may be driven out onto the AHCLKX or AHCLKR pins to serve as a reference clock for other components in the system.

The McASP requires a minimum of a bit clock and a frame sync to operate, and provides the capability to reference these clocks from an external high-frequency master clock. In DIT mode, it is possible to use only internally-generated clocks and frame syncs.

### 22.3.5.1 Transmit Clock

The transmit bit clock, ACLKX, (Figure 22-17) may be either externally sourced from the ACLKX pin or internally generated, as selected by the CLKXM bit. If internally generated (CLKXM = 1), the clock is divided down by a programmable bit clock divider (CLKXDIV) from the transmit high-frequency master clock (AHCLKX).

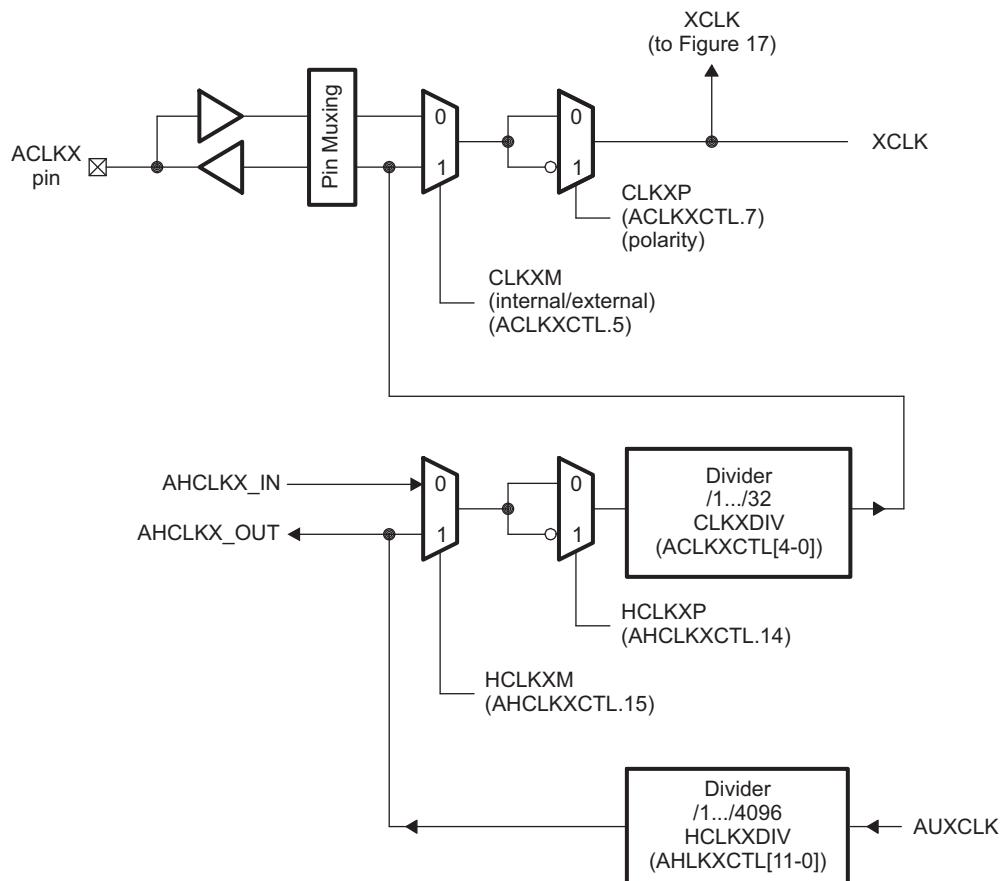
Internally, the McASP always shifts transmit data at the rising edge of the internal transmit clock, XCLK, (Figure 22-17). The CLKXP mux determines if ACLKX needs to be inverted to become XCLK. If CLKXP = 0, the CLKXP mux directly passes ACLKX to XCLK. As a result, the McASP shifts transmit data at the rising edge of ACLKX. If CLKXP = 1, the CLKX mux passes the inverted version of ACLKX to XCLK. As a result, the McASP shifts transmit data at the falling edge of ACLKX.

The transmit high-frequency master clock, AHCLKX, may be either externally sourced from the AHCLKX pin or internally generated, as selected by the HCLKXM bit. If internally generated ( $HCLKXM = 1$ ), the clock is divided down by a programmable high clock divider (HCLKXDIV) from McASP internal clock source AUXCLK. The transmit high-frequency master clock may be (but is not required to be) output on the AHCLKX pin where it is available to other devices in the system.

The transmit clock configuration is controlled by the following registers:

- ACLKXCTL.
  - AHCLKXCTL.

**Figure 22-17. Transmit Clock Generator Block Diagram**



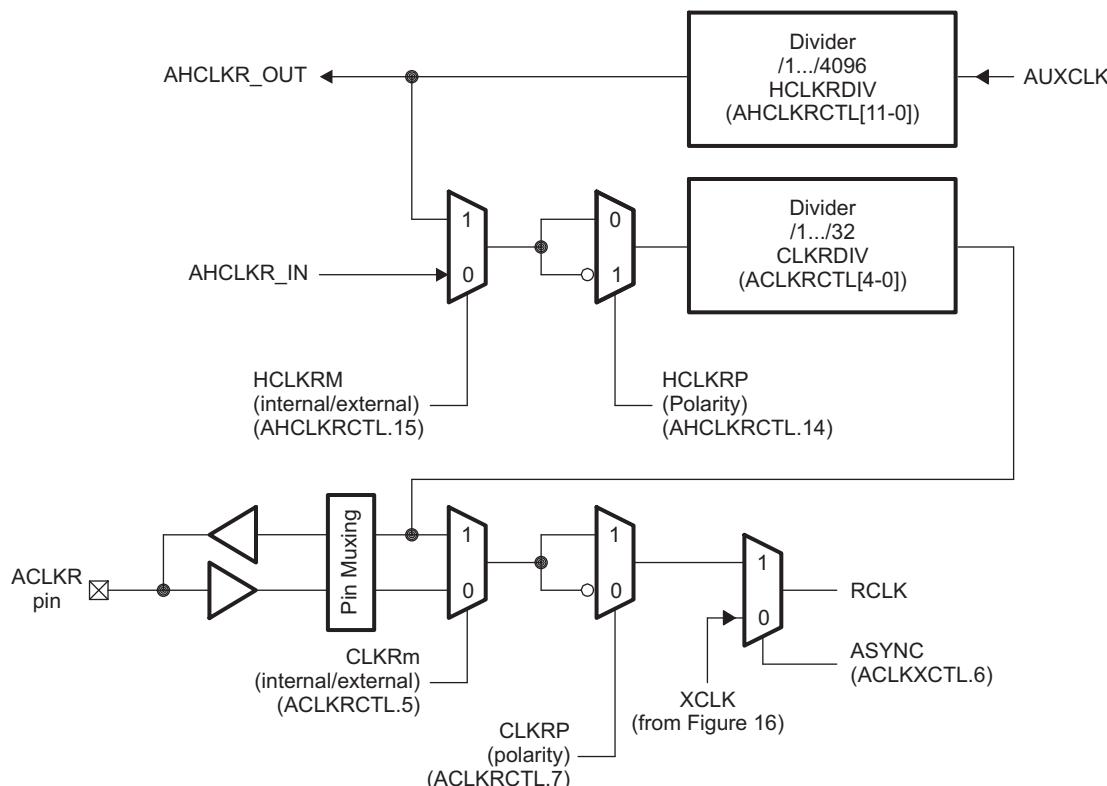
### 22.3.5.2 Receive Clock

The receiver also has the option to operate synchronously from the ACLKX and AFSX signals. This is achieved when the ASYNC bit in the transmit clock control register (ACLKXCTL) is cleared to 0 (see Figure 22-18). The receiver may be configured with different polarity (CLKRP) and frame sync data delay options from those options of the transmitter.

The receive clock configuration is controlled by the following registers:

- ACLKRCTL.
- AHCLKRCTL.

**Figure 22-18. Receive Clock Generator Block Diagram**



### 22.3.5.3 Frame Sync Generator

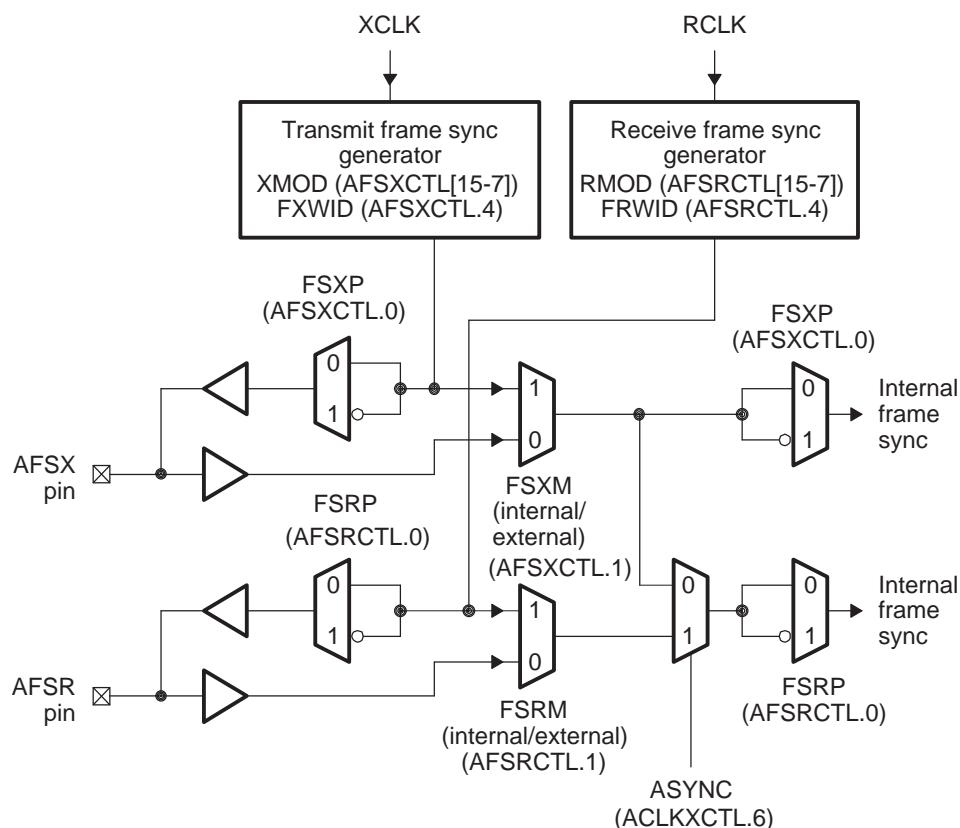
There are two different modes for frame sync: burst and TDM. A block diagram of the frame sync generator is shown in [Figure 22-19](#). The frame sync options are programmed by the receive and transmit frame sync control registers (AFSRCTL and AFSXCTL). The options are:

- Internally-generated or externally-generated.
- Frame sync polarity: rising edge or falling edge.
- Frame sync width: single bit or single word.
- Bit delay: 0, 1, or 2 cycles before the first data bit.

The transmit frame sync pin is AFSX and the receive frame sync pin is AFSR. A typical usage for these pins is to carry the left/right clock (LRCLK) signal when transmitting and receiving stereo data.

Regardless if the AFSX/AFSR is internally generated or externally sourced, the polarity of AFSX/AFSR is determined by FSXP/FSRP, respectively, to be either rising or falling edge. If FSXP/FSRP = 0, the frame sync polarity is rising edge. If FSRP/FSXP = 1, the frame sync polarity is falling edge.

**Figure 22-19. Frame Sync Generator Block Diagram**



#### 22.3.5.4 Clocking Examples

Some examples of processes using the McASP clocking and frame flexibility are:

- Receive data from a DVD at 48 kHz, but output up-sampled or decoded audio at 96 kHz or 192 kHz. This could be accomplished by inputting a high-frequency master clock (for example, 512 × receive FS), receiving with an internally-generated bit clock ratio of divide-by-8, and transmitting with an internally-generated bit clock ratio of divide-by-4 or divide-by-2.
- Transmit/receive data based on one sample rate (for example, 44.1 kHz), and transmit/receive data at a different sample rate (for example, 48 kHz).

#### 22.3.6 Signal Descriptions

The signals used on the McASP audio interface are listed in [Table 22-6](#).

**Table 22-6. McASP Interface Signals**

| Pin                        | I/O/Z | Device Reset<br>(RESET = 0) | Description                                     |
|----------------------------|-------|-----------------------------|---|
| <b>Transmitter Control</b> |       |                             |   |
| AHCLKX                     | I/O/Z | Input                       | Transmit high-frequency master clock            |
| AFSX                       | I/O/Z | Input                       | Transmit frame sync or left/right clock (LRCLK) |
| ACLKX                      | I/O/Z | Input                       | Transmit bit clock                              |
| <b>Receiver Control</b>    |       |                             |   |
| AHCLKR                     | I/O/Z | Input                       | Receive high-frequency master clock             |
| AFSR                       | I/O/Z | Input                       | Receive frame sync or left/right clock (LRCLK)  |
| ACLKR                      | I/O/Z | Input                       | Receive bit clock                               |
| <b>Mute</b>                |       |                             |   |
| AMUTE                      | I/O/Z | Input                       | Mute output                                     |
| AMUTEIN                    | I/O/Z | Input                       | Mute input                                      |
| <b>Data</b>                |       |                             |   |
| AXRn                       | I/O/Z | Input                       | TX/RX data pins                                 |

#### 22.3.7 Pin Multiplexing

The McASP signals share pins with other processor functions. For detailed information on the McASP pin multiplexing and configuration, see the pin multiplexing information in the device-specific data manual.

## 22.3.8 Transfer Modes

### 22.3.8.1 Burst Transfer Mode

The McASP supports a burst transfer mode, which is useful for nonaudio data such as passing control information between two processors. Burst transfer mode uses a synchronous serial format similar to the TDM mode. The frame sync generation is not periodic or time-driven as in TDM mode, but data driven, and the frame sync is generated for each data word transferred.

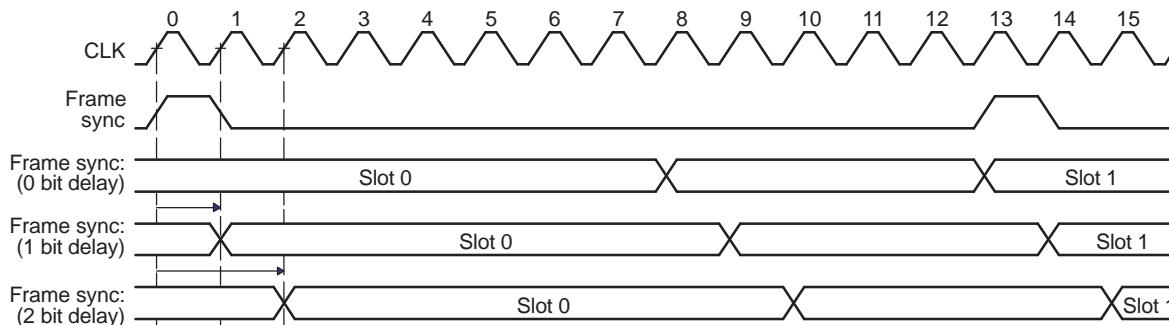
When operating in burst frame sync mode (Figure 22-20), as specified for transmit (XMOD = 0 in AFSXCTL) and receive (RMOD = 0 in AFSRCTL), one slot is shifted for each active edge of the frame sync signal that is recognized. Additional clocks after the slot and before the next frame sync edge are ignored.

In burst frame sync mode, the frame sync delay may be specified as 0, 1, or 2 serial clock cycles. This is the delay between the frame sync active edge and the start of the slot. The frame sync signal lasts for a single bit clock duration (FRWID = 0 in AFSRCTL, FXWID = 0 in AFSXCTL).

For transmit, when generating the transmit frame sync internally, the frame sync begins when the previous transmission has completed and when all the XBUF[n] (for every serializer set to operate as a transmitter) has been updated with new data.

For receive, when generating the receive frame sync internally, frame sync begins when the previous transmission has completed and when all the RBUF[n] (for every serializer set to operate as a receiver) has been read.

**Figure 22-20. Burst Frame Sync Mode**



The control registers must be configured as follows for the burst transfer mode. The burst mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 22.3.12.2](#) to configure this register.
- AMUTE: Not applicable. Leave at default.
- DLBCTL: If loopback mode is desired, configure this register according to [Section 22.3.10.5](#), otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select non-DIT mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to [Section 22.3.9.2](#) and [Section 22.3.10.3](#).
- RFMT/XFMT: Program all fields according to data format desired. See [Section 22.3.10.3](#).
- AFSRCTL/AFSXCTL: Clear **RMOD/XMOD** bits to 0 to indicate burst mode. Clear **FRWID/FXWID** bits to 0 for single bit frame sync duration. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See [Section 22.3.5](#).
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See [Section 22.3.5](#).

- RTDM/XTDM: Program RTDMS0/XTDMS0 to 1 to indicate one active slot only. Leave other fields at default.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Not applicable. Leave at default.
- SRCTLn: Program SRMOD to inactive/transmitter/receiver as desired. DISMOD is not applicable and should be left at default.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

### 22.3.8.2 Time-Division Multiplexed (TDM) Transfer Mode

The McASP time-division multiplexed (TDM) transfer mode supports the TDM format discussed in [Section 22.3.3.1](#).

Transmitting data in the TDM transfer mode requires a minimum set of pins:

- ACLKX - transmit bit clock.
- AFSX - transmit frame sync (or commonly called left/right clock).
- One or more serial data pins, AXR<sub>n</sub>, whose serializers have been configured to transmit.

The transmitter has the option to receive the ACLKX bit clock as an input, or to generate the ACLKX bit clock by dividing down the AHCLKX high-frequency master clock. The transmitter can either generate AHCLKX internally or receive AHCLKX as an input. See [Section 22.3.5.1](#).

Similarly, to receive data in the TDM transfer mode requires a minimum set of pins:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock).
- One or more serial data pins, AXR<sub>n</sub>, whose serializers have been configured to receive.

The receiver has the option to receive the ACLKR bit clock as an input or to generate the ACLKR bit clock by dividing down the AHCLKR high-frequency master clock. The receiver can either generate AHCLKR internally or receive AHCLKR as an input. See [Section 22.3.5.2](#) and [Section 22.3.5.3](#).

The control registers must be configured as follows for the TDM mode. The TDM mode specific bit fields are in bold face:

- PFUNC: The clock, frame, data pins must be configured for McASP function.
- PDIR: The clock, frame, data pins must be configured to the direction desired.
- PDOUT, PDIN, PDSET, PDCLR: Not applicable. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 22.3.12.2](#) to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: If loopback mode is desired, configure this register according to [Section 22.3.10.5](#), otherwise leave this register at default.
- DITCTL: DITEN must be left at default 0 to select TDM mode. Leave the register at default.
- RMASK/XMASK: Mask desired bits according to [Section 22.3.9.2](#) and [Section 22.3.10.3](#).
- RFMT/XFMT: Program all fields according to data format desired. See [Section 22.3.10.3](#).
- AFSRCTL/AFSXCTL: Set **RMOD/XMOD** bits to 2-32 for TDM mode. Configure other fields as desired.
- ACLKRCTL/ACLKXCTL: Program all fields according to bit clock desired. See [Section 22.3.5](#).
- AHCLKRCTL/AHCLKXCTL: Program all fields according to high-frequency clock desired. See [Section 22.3.5](#).
- RTDM/XTDM: Program all fields according to the time slot characteristics desired.
- RINTCTL/XINTCTL: Program all fields according to interrupts desired.
- RCLKCHK/XCLKCHK: Program all fields according to clock checking desired.
- SRCTLn: Program all fields according to serializer operation desired.
- DITCSRA[n], DITCSRB[n], DITUDRA[n], DITUDRB[n]: Not applicable. Leave at default.

### 22.3.8.2.1 TDM Time Slots

TDM mode on the McASP can extend to support multiprocessor applications, with up to 32 time slots per frame. For each of the time slots, the McASP may be configured to participate or to be inactive by configuring XTDM and/or RTDM (this allows multiple processors to communicate on the same TDM serial bus).

The TDM sequencer (separate ones for transmit and receive) functions in this mode. The TDM sequencer counts the slots beginning with the frame sync. For each slot, the TDM sequencer checks the respective bit in either XTDM or RTDM to determine if the McASP should transmit/receive in that time slot.

If the transmit/receive bit is active, the McASP functions normally during that time slot; otherwise, the McASP is inactive during that time slot; no update to the buffer occurs, and no event is generated.

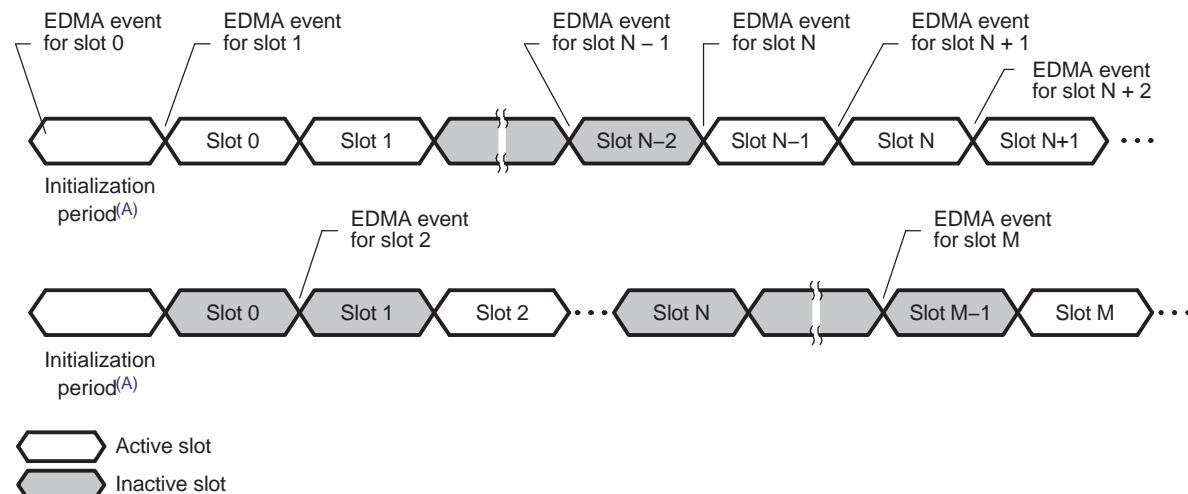
Transmit pins are automatically set to a high-impedance state, 0, or 1 during that slot, as determined by bit DISMOD in SRCTL[n].

**Figure 22-21** shows when the transmit DMA event AXEVT is generated. See [Section 22.3.10.1.1](#) for details on data ready and the initialization period indication. The transmit DMA event for an active time slot (slot N) is generated during the previous time slot (slot N - 1), regardless if the previous time slot (slot N - 1) is active or inactive.

During an active transmit time slot (slot N), if the next time slot (slot N + 1) is configured to be active, the copy from XRBUF[n] to XRSR[n] generates the DMA event for time slot N + 1. If the next time slot (slot N + 1) is configured to be inactive, then the DMA event will be delayed to time slot M - 1. In this case, slot M is the next active time slot. The DMA event for time slot M is generated during the first bit time of slot M - 1.

The receive DMA request generation does not need this capability, since the receive DMA event is generated after data is received in the buffer (looks back in time). If a time slot is disabled, then no data is copied to the buffer for that time slot and no DMA event is generated.

**Figure 22-21. Transmit DMA Event (AXEVT) Generation in TDM Time Slots**



A See [Section 22.3.12.2](#), step 7a.

### 22.3.8.2.2 Special 384 Slot TDM Mode for Connection to External DIR

The McASP receiver also supports a 384 time slot TDM mode (DIR mode), to support S/PDIF, AES-3, IEC-60958 receiver ICs whose natural block (block corresponds to McASP frame) size is 384 samples. The advantage to using the 384 time slot TDM mode is that interrupts may be generated synchronous to the S/PDIF, AES-3, IEC-60958, such as the last slot interrupt.

The receive TDM time slot register (RTDM) should be programmed to all 1s during reception of a DIR block. Other TDM functionalities (for example, inactive slots) are not supported (only the slot counter counts the 384 subframes in a block).

To receive data in the DIR mode, the following pins are typically needed:

- ACLKR - receive bit clock.
- AFSR - receive frame sync (or commonly called left/right clock). In this mode, AFSR should be connected to a DIR which outputs a start of block signal, instead of LRCLK.
- One or more serial data pins, AXRn, whose serializers have been configured to receive.

For this special DIR mode, the control registers can be configured just as for TDM mode, except set RMOD in AFSRCTL to 384 to receive 384 time slots.

### 22.3.8.3 Digital Audio Interface Transmit (DIT) Transfer Mode

In addition to the TDM and burst transfer modes, which are suitable for transmitting audio data between ICs inside the same system, the digital audio interface transmit (DIT) transfer mode of the McASP also supports transmission of audio data in the S/PDIF, AES-3, or IEC-60958 format. These formats are designed to carry audio data between different systems through an optical or coaxial cable. The DIT mode only applies to serializers configured as transmitters, not receivers. See [Section 22.3.3.2](#) for a description of the S/PDIF format.

#### 22.3.8.3.1 Transmit DIT Encoding

The McASP operation in DIT mode is basically identical to the 2 time slot TDM mode, but the data transmitted is output as a biphase mark encoded bit stream, with preamble, channel status, user data, validity, and parity automatically stuffed into the bit stream by the McASP. The McASP includes separate validity bits for even/odd subframes and two 384-bit RAM modules to hold channel status and user data bits.

The transmit TDM time slot register (XTDM) should be programmed to all 1s during DIT mode. TDM functionality is not supported in DIT mode, except that the TDM slot counter counts the DIT subframes.

To transmit data in the DIT mode, the following pins are typically needed:

- AHCLKX - transmit high-frequency master clock.
- One or more serial data pins, AXRn, whose serializers have been configured to transmit.

AHCLKX is optional (the internal clock source may be used instead), but if used as a reference, the processor provides a clock check circuit that continually monitors the AHCLKX input for stability.

If the McASP is configured to transmit in the DIT mode on more than one serial data pin, the bit streams on all pins will be synchronized. In addition, although they will carry unique audio data, they will carry the same channel status, user data, and validity information.

The actual 24-bit audio data must always be in bit positions 23-0 after passing through the first three stages of the transmit format unit.

For left-aligned Q31 data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 010 (rotate right by 8 bits).
- XRVRS = 0 (no bit reversal, LSB first).
- XMASK = FFFF FF00h-FFFF 0000h (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present).
- XPAD = 00 (pad extra bits with 0).

For right-aligned data, the following transmit format unit settings process the data into right aligned 24-bit audio data ready for transmission:

- XROT = 000 (rotate right by 0 bits).
- XRVRS = 0 (no bit reversal, LSB first).
- XMASK = 00FF FFFFh to 0000 FFFFh (depending upon whether 24, 23, 22, 21, 20, 19, 18, 17, or 16 valid audio data bits are present).
- XPAD = 00 (pad extra bits with 0).

#### **22.3.8.3.2 Transmit DIT Clock and Frame Sync Generation**

The DIT transmitter only works in the following configuration:

- In transmit frame control register (AFSXCTL):
  - Internally-generated transmit frame sync, FSXM = 1.
  - Rising-edge frame sync, FSXP = 0.
  - Bit-width frame sync, FXWID = 0.
  - 384-slot TDM, XMOD = 1 1000 0000b.
- In transmit clock control register (ACLKXCTL), ASYNC = 1.
- In transmit bitstream format register (XFMT), XSSZ = 1111 (32-bit slot size).

All combinations of AHCLKX and ACLKX are supported.

This is a summary of the register configurations required for DIT mode. The DIT mode specific bit fields are in bold face:

- PFUNC: The data pins must be configured for McASP function. If AHCLKX is used, it must also be configured for McASP function.
- PDIR: The data pins must be configured as outputs. If AHCLKX is used as an input reference, it should be configured as input. If internal clock source AUXCLK is used as the reference clock, it may be output on the AHCLKX pin by configuring AHCLKX as an output.
- PDOOUT, PDIN, PDSET, PDCLR: Not applicable for DIT operation. Leave at default.
- GBLCTL: Follow the initialization sequence in [Section 22.3.12.2](#) to configure this register.
- AMUTE: Program all fields according to mute control desired.
- DLBCTL: Not applicable. Loopback is not supported for DIT mode. Leave at default.
- DITCTL: **DITEN** bit must be set to 1 to enable DIT mode. Configure other bits as desired.
- RMASK: Not applicable. Leave at default.
- RFMT: Not applicable. Leave at default.
- AFSRCTL: Not applicable. Leave at default.
- ACLKRCTL: Not applicable. Leave at default.
- AHCLKRCTL: Not applicable. Leave at default.
- RTDM: Not applicable. Leave at default.
- RINTCTL: Not applicable. Leave at default.
- RCLKCHK: Not applicable. Leave at default.
- **XMASK**: Mask desired bits according to the discussion in this section, depending upon left-aligned or right-aligned internal data.

- **XFMT:** XDATDLY = 0. XRVRS = 0. XPAD = 0. XPBIT = default (not applicable). XSSZ = Fh (32-bit slot). XBUSEL = configured as desired. **XROT** bit is configured according to the discussion in this section, either 0 or 8-bit rotate.
- **AFSXCTL:** Configure the bits according to the discussion in this section.
- **ACLKXCTL:** ASYNC = 1. Program CLKXDIV bits to obtain the bit clock rate desired. Configure CLKXP and CLKXM bits as desired, because CLKX is not actually used in the DIT protocol.
- **AHCLKXCTL:** Program all fields according to high-frequency clock desired.
- **XTDM:** Set to FFFF FFFFh for all active slots for DIT transfers.
- **XINTCTL:** Program all fields according to interrupts desired.
- **XCLKCHK:** Program all fields according to clock checking desired.
- **SRCTLn:** Set **SRMOD** = 1 (transmitter) for the DIT pins. DISMOD field is don't care for DIT mode.
- **DITCSRA[n], DITCSRB[n]:** Program the channel status bits as desired.
- **DITUDRA[n], DITUDRB[n]:** Program the user data bits as desired.

#### **22.3.8.3.3 DIT Channel Status and User Data Register Files**

The channel status registers (DITCSRA $n$  and DITCSRB $n$ ) and user data registers (DITUDRA $n$  and DITUDRB $n$ ) are not double buffered. Typically the programmer uses one of the synchronizing interrupts, such as last slot, to create an event at a safe time so the register may be updated. In addition, the CPU reads the transmit TDM slot counter to determine which word of the register is being used.

It is a requirement that the software avoid writing to the word of user data and channel status that are being used to encode the current time slot; otherwise, it will be indeterminate whether the old or new data is used to encode the bitstream.

The DIT subframe format is defined in [Section 22.3.3.2.2](#). The channel status information (C) and User Data (U) are defined in these DIT control registers:

- DITCSRA0 to DITCSRA5: The 192 bits in these six registers contain the channel status information for the LEFT channel within each frame.
- DITCSRB0 to DITCSRB5: The 192 bits in these six registers contain the channel status information for the RIGHT channel within each frame.
- DITUDRA0 to DITUDRA5: The 192 bits in these six registers contain the user data information for the LEFT channel within each frame.
- DITUDRB0 to DITUDRB5: The 192 bits in these six registers contain the user data information for the RIGHT channel within each frame.

The S/PDIF block format is shown in [Figure 22-13](#). There are 192 frames within a block (frame 0 to frame 191). Within each frame there are two subframes (subframe 1 and 2 for left and right channels, respectively). The channel status and user data information sent on each subframe is summarized in [Table 22-7](#).

**Table 22-7. Channel Status and User Data for Each DIT Block**

| Frame  | Subframe | Preamble | Channel Status defined in: | User Data defined in: |
|--|----------|----------|----------------------------|-----------------------|
| <b>Defined by DITCSRA0, DITCSRB0, DITUDRA0, DITUDRB0</b> |          |          |                            |                       |
| 0  | 1 (L)    | B        | DITCSRA0[0]                | DITUDRA0[0]           |
| 0  | 2 (R)    | W        | DITCSRB0[0]                | DITUDRB0[0]           |
| 1  | 1 (L)    | M        | DITCSRA0[1]                | DITUDRA0[1]           |
| 1  | 2 (R)    | W        | DITCSRB0[1]                | DITUDRB0[1]           |
| 2  | 1 (L)    | M        | DITCSRA0[2]                | DITUDRA0[2]           |
| 2  | 2 (R)    | W        | DITCSRB0[2]                | DITUDRB0[2]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 31   | 1 (L)    | M        | DITCSRA0[31]               | DITUDRA0[31]          |
| 31   | 2 (R)    | W        | DITCSRB0[31]               | DITUDRB0[31]          |
| <b>Defined by DITCSRA1, DITCSRB1, DITUDRA1, DITUDRB1</b> |          |          |                            |                       |
| 32   | 1 (L)    | M        | DITCSRA1[0]                | DITUDRA1[0]           |
| 32   | 2 (R)    | W        | DITCSRB1[0]                | DITUDRB1[0]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 63   | 1 (L)    | M        | DITCSRA1[31]               | DITUDRA1[31]          |
| 63   | 2 (R)    | W        | DITCSRB1[31]               | DITUDRB1[31]          |
| <b>Defined by DITCSRA2, DITCSRB2, DITUDRA2, DITUDRB2</b> |          |          |                            |                       |
| 64   | 1 (L)    | M        | DITCSRA2[0]                | DITUDRA2[0]           |
| 64   | 2 (R)    | W        | DITCSRB2[0]                | DITUDRB2[0]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 95   | 1 (L)    | M        | DITCSRA2[31]               | DITUDRA2[31]          |
| 95   | 2 (R)    | W        | DITCSRB2[31]               | DITUDRB2[31]          |
| <b>Defined by DITCSRA3, DITCSRB3, DITUDRA3, DITUDRB3</b> |          |          |                            |                       |
| 96   | 1 (L)    | M        | DITCSRA3[0]                | DITUDRA3[0]           |
| 96   | 2 (R)    | W        | DITCSRB3[0]                | DITUDRB3[0]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 127  | 1 (L)    | M        | DITCSRA3[31]               | DITUDRA3[31]          |
| 127  | 2 (R)    | W        | DITCSRB3[31]               | DITUDRB3[31]          |
| <b>Defined by DITCSRA4, DITCSRB4, DITUDRA4, DITUDRB4</b> |          |          |                            |                       |
| 128  | 1 (L)    | M        | DITCSRA4[0]                | DITUDRA4[0]           |
| 128  | 2 (R)    | W        | DITCSRB4[0]                | DITUDRB4[0]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 159  | 1 (L)    | M        | DITCSRA4[31]               | DITUDRA4[31]          |
| 159  | 2 (R)    | W        | DITCSRB4[31]               | DITUDRB4[31]          |
| <b>Defined by DITCSRA5, DITCSRB5, DITUDRA5, DITUDRB5</b> |          |          |                            |                       |
| 160  | 1 (L)    | M        | DITCSRA5[0]                | DITUDRA5[0]           |
| 160  | 2 (R)    | W        | DITCSRB5[0]                | DITUDRB5[0]           |
| ...  | ...      | ...      | ...                        | ...                   |
| 191  | 1 (L)    | M        | DITCSRA5[31]               | DITUDRA5[31]          |
| 191  | 2 (R)    | W        | DITCSRB5[31]               | DITUDRB5[31]          |

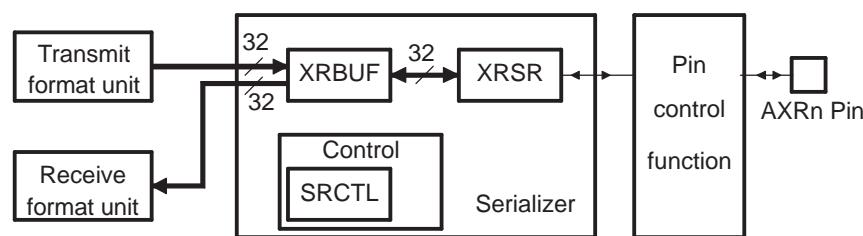
## 22.3.9 General Architecture

### 22.3.9.1 Serializers

Figure 22-22 shows the block diagram of the serializer and its interface to other units within the McASP. The serializers take care of shifting serial data in and out of the McASP. Each serializer consists of a shift register (XRSR), a data buffer (XRBUF), a control register (SRCTL), and logic to support the data alignment options of the McASP. For each serializer, there is a dedicated serial data pin (AXRn) and a dedicated control register (SRCTL[n]). The control register allows the serializer to be configured as a transmitter, receiver, or as inactive. When configured as a transmitter the serializer shifts out data to the serial data pin AXRn. When configured as a receiver, the serializer shifts in data from the AXRn pin. The serializer is clocked from the transmit/receive section clock (ACLKX/ACLKR) if configured to transmit/receive respectively.

All serializers that are configured to transmit operate in lock-step. Similarly, all serializers that are configured to receive also operate in lock-step. This means that at most there are two zones per McASP, one for transmit and one for receive.

**Figure 22-22. Individual Serializer and Connections Within McASP**



For receive, data is shifted in through the AXRn pin to the shift register XRSR. Once the entire slot of data is collected in the XRSR, the data is copied to the data buffer XRBUF. The data is now ready to be read by the processor through the RBUF register, which is an alias of the XRBUF for receive. When the processor reads from the RBUF, the McASP passes the data from RBUF through the receive format unit and returns the formatted data to the processor.

For transmit, the processor services the McASP by writing data into the XBUF register, which is an alias of the XRBUF for transmit. The data automatically passes through the transmit format unit before actually reaching the XRBUF register in the serializer. The data is then copied from XRBUF to XRSR, and shifted out from the AXRn synchronously to the serial clock.

In DIT mode, in addition to the data, the serializer shifts out other DIT-specific information accordingly (preamble, user data, etc.).

The serializer configuration is controlled by SRCTL[n].

### 22.3.9.2 Format Unit

The McASP has two data formatting units, one for transmit and one for receive. These units automatically remap the data bits within the transmitted and received words between a natural format for the processor (such as a Q31 representation) and the required format for the external serial device (such as "I2S format"). During the remapping process, the format unit also can mask off certain bits or perform sign extension.

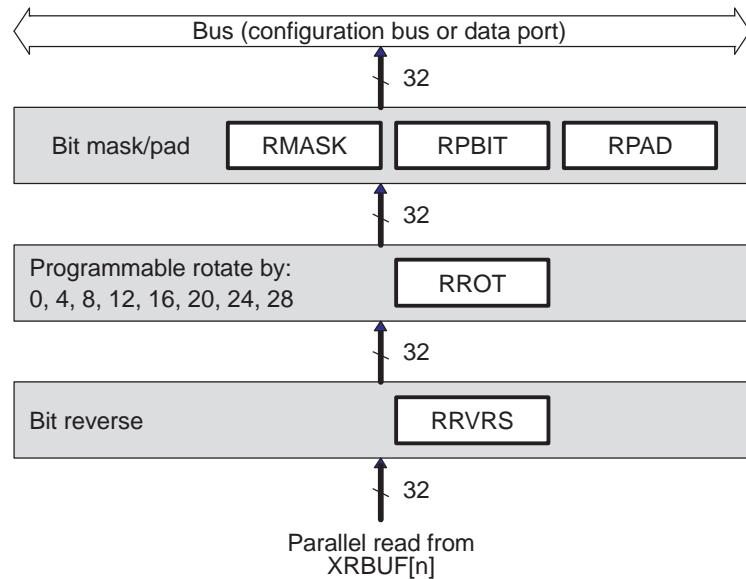
Since all transmitters share the same data formatting unit, the McASP only supports one transmit format at a time. For example, the McASP will not transmit in "I2S format" on serializer 0, while transmitting "Left Justified" on serializer 1. Likewise, the receiver section of the McASP only supports one data format at a time, and this format applies to all receiving serializers. However, the McASP can transmit in one format while receiving in a completely different format.

This formatting unit consists of three stages:

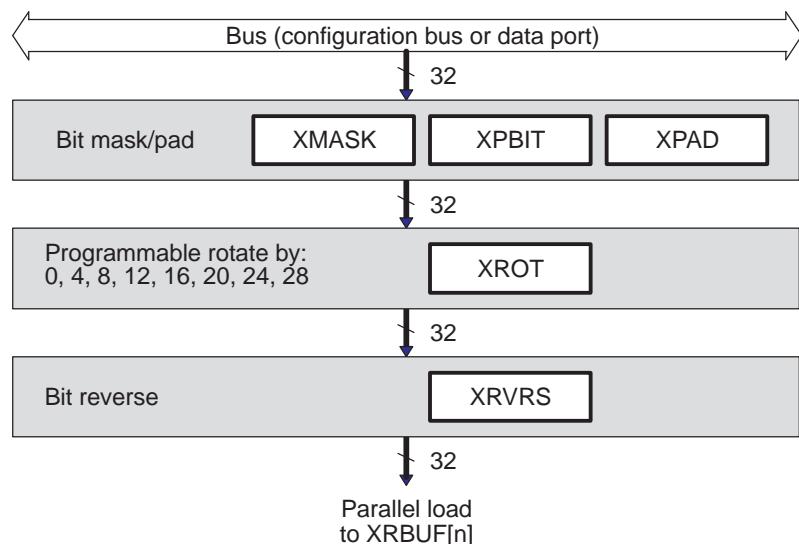
- Bit mask and pad (masks off bits, performs sign extension)
- Rotate right (aligns data within word)
- Bit reversal (selects between MSB first or LSB first)

[Figure 22-23](#) shows a block diagram of the receive formatting unit, and [Figure 22-24](#) shows the transmit formatting unit. Note that the order in which data flows through the three stages is different between the transmit and receive formatting units.

**Figure 22-23. Receive Format Unit**



**Figure 22-24. Transmit Format Unit**



The bit mask and pad stage includes a full 32-bit mask register, allowing selected individual bits to either pass through the stage unchanged, or be masked off. The bit mask and pad then pad the value of the masked off bits by inserting either a 0, a 1, or one of the original 32 bits as the pad value. The last option allows for sign-extension when the sign bit is selected to pad the remaining bits.

The rotate right stage performs bitwise rotation by a multiple of 4 bits (between 0 and 28 bits), programmable by the (R/X)FMT register. Note that this is a rotation process, not a shifting process, so bit 0 gets shifted back into bit 31 during the rotation.

The bit reversal stage either passes all 32 bits directly through, or swaps them. This allows for either MSB or LSB first data formats. If bit reversal is not enabled, then the McASP will naturally transmit and receive in an LSB first order.

Finally, note that the (R/X)DATDLY bits in (R/X)FMT also determine the data format. For example, the difference between I2S format and left-justified is determined by the delay between the frame sync edge and the first data bit of a given time slot. For I2S format, (R/X)DATDLY should be set to a 1-bit delay, whereas for left-justified format, it should be set to a 0-bit delay.

The combination of all the options in (R/X)FMT means that the McASP supports a wide variety of data formats, both on the serial data lines, and in the internal processor representation.

[Section 22.3.10.3](#) provides more detail and specific examples. The examples use internal representation in integer and Q31 notation, but other fractional notations are also possible.

### 22.3.9.3 State Machine

The receive and transmit sections have independent state machines. Each state machine controls the interactions between the various units in the respective section. In addition, the state machine keeps track of error conditions and serial port status.

No serial transfers can occur until the respective state machine is released from reset. See initialization sequence for details ([Section 22.3.12](#)).

The receive state machine is controlled by the RFMT register, and it reports the McASP status and error conditions in the RSTAT register. Similarly, the transmit state machine is controlled by the XFMT register, and it reports the McASP status and error conditions in the XSTAT register.

### 22.3.9.4 TDM Sequencer

There are separate TDM sequencers for the transmit section and the receive section. Each TDM sequencer keeps track of the slot count. In addition, the TDM sequencer checks the bits of (R/X)TDM and determines if the McASP should receive/transmit in that time slot.

If the McASP should participate (transmit/receive bit is active) in the time slot, the McASP functions normally. If the McASP should not participate (transmit/receive bit is inactive) in the time slot, no transfers between the XRBUF and XRSR registers in the serializer would occur during that time slot. In addition, the serializers programmed as transmitters place their data output pins in a predetermined state (logic low, high, or high impedance) as programmed by each serializer control register (SRCTL). Refer also to [Section 22.3.8.2](#) for details on how DMA event or interrupt generations are handled during inactive time slots in TDM mode.

The receive TDM sequencer is controlled by register RTDM and reports current receive slot to RSLOT. The transmit TDM sequencer is controlled by register XTDM and reports current transmit slot to XSLOT.

### 22.3.9.5 Clock Check Circuit

A common source of error in audio systems is a serial clock failure due to instabilities in the off-chip DIR circuit. To detect a clock error quickly, a clock-check circuit is included in the McASP for both transmit and receive clocks, since both may be sourced from off chip.

The clock check circuit can detect and recover from transmit and receive clock failures. See [Section 22.3.10.4.6](#) for implementation and programming details.

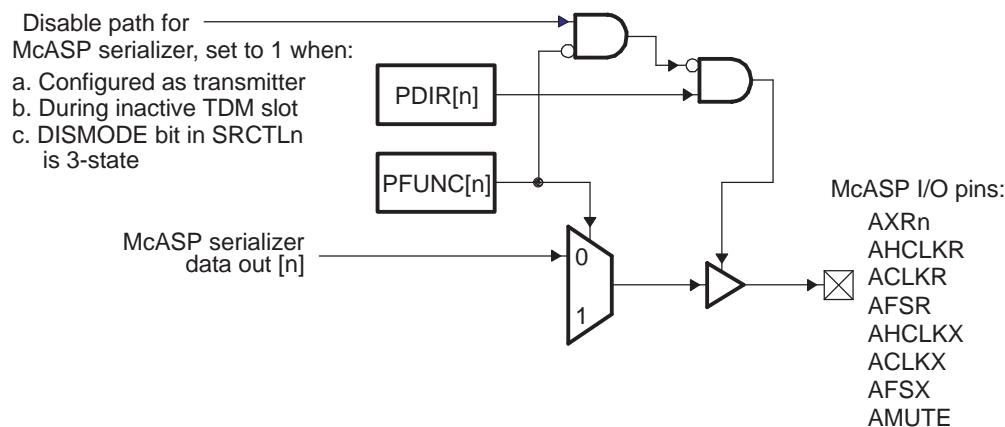
### 22.3.9.6 Pin Function Control

All McASP pins except AMUTEIN are bidirectional input/output pins. In addition, these bidirectional pins function either as McASP or general-purpose I/O (GPIO) pins. The following registers control the pin functions:

- Pin function register (PFUNC): selects pin to function as McASP or GPIO.
- Pin direction register (PDIR): selects pin to be input or output.
- Pin data input register (PDIN): shows data input at the pin.
- Pin data output register (PDOUT): data to be output at the pin if the pin is configured as GPIO output ( $\text{PFUNC}[n] = 1$  and  $\text{PDIR}[n] = 1$ ). Not applicable when the pin is configured as McASP pin ( $\text{PFUNC}[n] = 0$ ).
- Pin data set register (PDSET): alias of PDOUT. Writing a 1 to  $\text{PDSET}[n]$  sets the respective  $\text{PDOUT}[n]$  to 1. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output ( $\text{PFUNC}[n] = 1$  and  $\text{PDIR}[n] = 1$ ).
- Pin data clear register (PDCLR): alias of PDOUT. Writing a 1 to  $\text{PDCLR}[n]$  clears the respective  $\text{PDOUT}[n]$  to 0. Writing a 0 has no effect. Applicable only when the pin is configured as GPIO output ( $\text{PFUNC}[n] = 1$  and  $\text{PDIR}[n] = 1$ ).

See the register descriptions in [Section 22.4](#) for details on the mapping of each McASP pin to the register bits. [Figure 22-25](#) shows the pin control block diagram.

**Figure 22-25. McASP I/O Pin Control Block Diagram**



#### 22.3.9.6.1 McASP Pin Control—Transmit and Receive

You must correctly set the McASP GPIO registers PFUNC and PDIR, even when McASP pins are used for their serial port (non-GPIO) function.

Serial port functions include:

- Clock pins (ACLKX, ACLKR, AHCLKX, AHCLKR, AFSX, AFSR) used as clock inputs and outputs.
- Serializer data pins (AXRn) used to transmit or receive.
- AMUTE used as a mute output signal.

When using these pins in their serial port function, you must clear PFUNC[n] to 0 for each pin.

Also, certain outputs require  $\text{PDIR}[n] = 1$ , such as clock pins used as clock outputs, serializer data pins used to transmit, and AMUTE used as mute output.

Clock inputs and serializers configured to receive must have  $\text{PDIR}[n] = 0$ .

PFUNC and PDIR do not control the AMUTEIN signal, it is usually tied to a device level interrupt pin (consult device datasheet). If used as a mute input, this pin needs to be configured as an input in the appropriate peripheral.

Finally, there is an important advantage to having separate control of pin direction (by PDIR), and the choice of internal versus external clocking (by CLKRM/CLKXM). Depending on the specific device and usage, you might select an external clock (CLKRM = 0), while enabling the internal clock divider, and the clock pin as an output in the PDIR register (PDIR[ACLKR] = 1). In this case, the bit clock is an output (PDIR[ACLKR] = 1) and, therefore, routed to the ACLKR pin. However, because CLKRM = 0, the bit clock is then routed back to the McASP module as an "external" clock source. This may result in less skew between the clock inside the McASP and the clock in the external device, thus producing more balanced setup and hold times for a particular system. As a result, this may allow a higher serial clock rate interface.

### **22.3.10 Operation**

This section discusses the operation of the McASP.

#### **22.3.10.1 Data Transmission and Reception**

The processor services the McASP by writing data to the XBUF register(s) for transmit operations, and by reading data from the RBUF register(s) for receive operations. The McASP sets status flag and notifies the processor whenever data is ready to be serviced. [Section 22.3.10.1.1](#) discusses data ready status in detail.

The XBUF and RBUF registers can be accessed through one of the two peripheral ports of the device:

- The data port (DAT): This port is dedicated for data transfers on the device.
- The configuration bus (CFG): This port is used for both data transfers and peripheral configuration control on the device.

[Section 22.3.10.1.2](#) and [Section 22.3.10.1.3](#) discuss how to perform transfers through the data port and the configuration bus.

Either the CPU or the DMA can be used to service the McASP through any of these two peripheral ports. The CPU and DMA usages are discussed in [Section 22.3.10.1.4](#) and [Section 22.3.14.2](#).

#### **22.3.10.1.1 Data Ready Status and Event/Interrupt Generation**

##### **22.3.10.1.1.1 Transmit Data Ready**

The transmit data ready flag XDATA bit in the XSTAT register reflects the status of the XBUF register. The XDATA flag is set when data is transferred from the XRBUF[n] buffers to the XRSR[n] shift registers, indicating that the XBUF is empty and ready to accept new data from the processor. This flag is cleared when the XDATA bit is written with a 1, or when all the serializers configured as transmitters are written by the processor.

Whenever XDATA is set, an DMA event AXEVT is automatically generated to notify the DMA of the XBUF empty status. An interrupt AXINT is also generated if XDATA interrupt is enabled in the XINTCTL register (See [Section 22.3.13.2](#) for details).

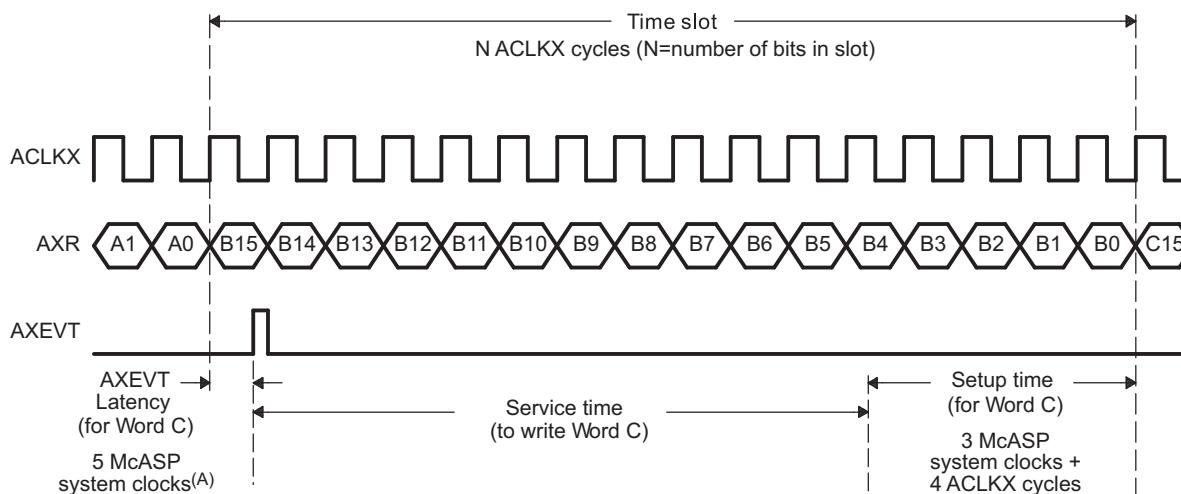
For DMA requests, the McASP does not require XSTAT to be read between DMA events. This means that even if XSTAT already has the XDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Since all serializers act in lockstep, only one DMA event is generated to indicate that all active transmit serializers are ready to be written to with new data.

[Figure 22-26](#) shows the timing details of when AXEVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is transmitted, the McASP sets the XDATA flag and generates an AXEVT event. However, it takes up to 5 McASP system clocks (AXEVT Latency) before AXEVT is active at the McASP boundary. Upon AXEVT, the processor can begin servicing the McASP by writing Word C into the XBUF (Processor Service Time). The processor must write Word C into the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum Processor Service Time ([Figure 22-26](#)) can be calculated as:

$$\text{Processor Service Time} = \text{Time Slot} - \text{AXEVT Latency} - \text{Setup Time}$$

**Figure 22-26. Processor Service Time Upon Transmit DMA Event (AXEVT)**


A Refer to the device-specific data manual for the McASP system clock source. This is not the same as AUXCLK.

**Example 22-1. Processor Service Time Calculation for Transmit DMA Event (AXEVT)**

The following is an example to show how to calculate Processor Service Time. Assume the following setup:

- McASP transmits in I2S format at 192 kHz frame rate. Assume slot size is 32 bit.

With the above setup, we obtain the following parameters corresponding to Figure 22-26:

- Calculation of McASP system clock cycle:
  - System functional clock = 26 MHz
  - Therefore, McASP system clock cycle =  $1/26\text{MHz} = 38.5\text{ ns}$ .
- Calculation of ACLKX clock cycle:
  - This example has two 32-bit slots per frame, for a total of 64 bits per frame.
  - ACLKX clock cycle is  $(1/192\text{ kHz})/64 = 81.4\text{ ns}$ .
- Time Slot between AXEVT events:
  - For I2S format, McASP generates two AXEVT events per 192 kHz frame.
  - Therefore, Time Slot between AXEVT events is  $(1/192\text{ kHz})/2 = 2604\text{ ns}$ .
- AXEVT Latency:
  - = 5 McASP system clocks
  - =  $38.5\text{ ns} \times 5 = 192.5\text{ ns}$
- Setup Time
  - = 3 McASP system clocks + 4 ACLKX cycles
  - =  $(38.5\text{ ns} \times 3) + (81.4\text{ ns} \times 4)$
  - =  $441.1\text{ ns}$
- Processor Service Time
  - = Time Slot - AXEVT Latency - Setup Time
  - =  $2604\text{ ns} - 441.1\text{ ns} - 192.5\text{ ns}$
  - =  $1970.4\text{ ns}$

### 22.3.10.1.1.2 Receive Data Ready

Similarly, the receive data ready flag RDATA in the RSTAT reflects the status of the RBUF register. The RDATA flag is set when data is transferred from the XRSR[n] shift registers to the XRBUF[n] buffers, indicating that the RBUF contains received data and is ready to have the processor read the data. This flag is cleared when the RDATA bit is written with a 1, or when all the serializers configured as receivers are read.

Whenever RDATA is set, an DMA event AREVT is automatically generated to notify the DMA of the RBUF ready status. An interrupt ARINT is also generated if RDATA interrupt is enabled in the RINTCTL register (See [Section 22.3.13.3](#) for details).

For DMA requests, the McASP does not require RSTAT to be read between DMA events. This means that even if RSTAT already has the RDATA flag set to 1 from a previous request, the next transfer triggers another DMA request.

Since all serializers act in lockstep, only one DMA event is generated to indicate that all active receive serializers are ready to receive new data.

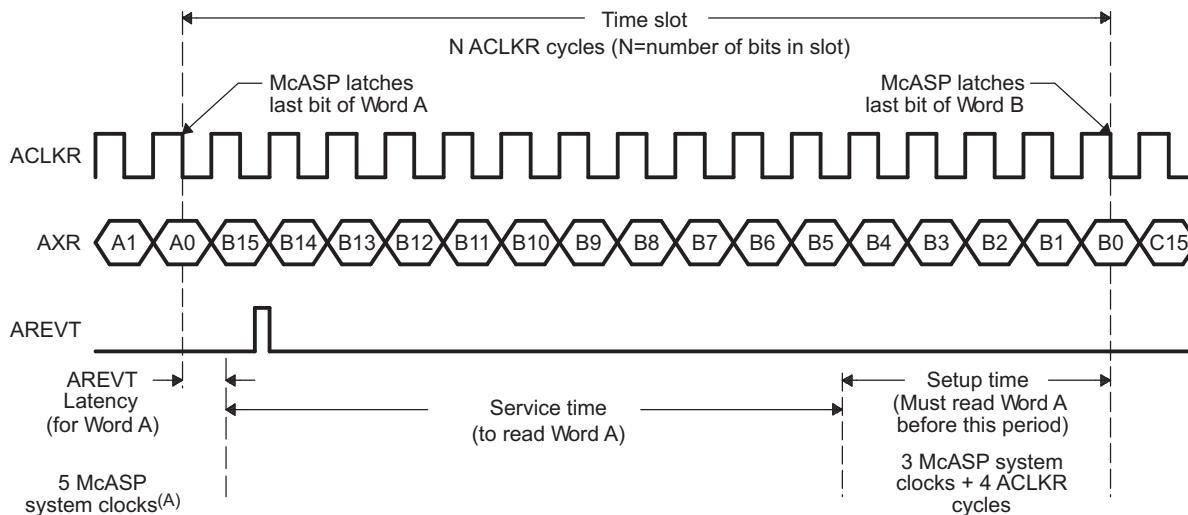
[Figure 22-27](#) shows the timing details of when AREVT is generated at the McASP boundary. In this example, as soon as the last bit (bit A0) of Word A is received, the McASP sets the RDATA flag and generates an AREVT event. However, it takes up to 5 McASP system clocks (AREVT Latency) before AREVT is active at the McASP boundary. Upon AREVT, the processor can begin servicing the McASP by reading Word A from the RBUF (Processor Service Time). The processor must read Word A from the XBUF no later than the setup time required by the McASP (Setup Time).

The maximum Processor Service Time ([Figure 22-27](#)) can be calculated as:

$$\text{Processor Service Time} = \text{Time Slot} - \text{AREVT Latency} - \text{Setup Time}$$

The Processor Service Time calculation for receive is similar to the calculation for transmit. See [Example 22-1](#) for Processor Service Time calculation using transmit as an example.

**Figure 22-27. Processor Service Time Upon Receive DMA Event (AREVT)**



A The device uses SYSCLK2 as the McASP system clock source.

### 22.3.10.1.2 Transfers Through the Data Port (DAT)

---

**NOTE:** To perform internal transfers through the data port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

---

Typically, you will access the McASP XRBUF registers through the data port. To access through the data port, simply have the CPU or DMA access the XRBUF through its data port location. Through the data port, the DMA/CPU can service all the serializers through a single address. The McASP automatically cycles through the appropriate serializers.

For transmit operations through the data port, the DMA/CPU should write to the same XBUF data port address to service all of the active transmit serializers. In addition, the DMA/CPU should write to the XBUF for all active transmit serializers in incremental (although not necessarily consecutive) order. For example, if serializers 0, 4, and 5, are set up as active transmitters, the DMA/CPU should write to the XBUF data port address four times with data for serializers 0, 4, and 5, upon each transmit data ready event. This exact servicing order must be followed so that data appears in the appropriate serializers.

Similarly, for receive operations through the data port, the DMA/CPU should read from the same RBUF data port address to service all of the active receive serializers. In addition, reads from the active receive serializers through the data port return data in incremental (although not necessarily consecutive) order. For example, if serializers 1, 2, and 3, are set up as active receivers, the DMA/CPU should read from the RBUF data port address four times to obtain data for serializers 1, 2, and 3, in this exact order, upon each receive data ready event.

When transmitting, the DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition ([Section 22.3.10.4.2](#)).

Similarly, when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do results in a buffer overrun condition ([Section 22.3.10.4.3](#)).

To perform internal transfers through the data port, clear XBUSEL/RBUSEL bit to 0 in the respective XFMT/RFMT registers.

### 22.3.10.1.3 Transfers Through the Configuration Bus (CFG)

---

**NOTE:** To perform internal transfers through the configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers. Failure to do so will result in software malfunction.

---

In this method, the DMA/CPU accesses the XRBUF registers through the configuration bus address. The exact XRBUF register address for any particular serializer is determined by adding the offset for that particular serializer to the base address for the particular McASP. XRBUF for the serializers configured as transmitters is given the name XBUF $n$ . For example, the XRBUF associated with transmit serializer 2 is named XBUF2. Similarly, XRBUF for the serializers configured as receivers is given the name RBUF $n$ .

Accessing the XRBUF registers through the data port is different because the CPU/DMA only needs to access one single address. When accessing through the configuration bus, the CPU/DMA must provide the exact XBUF $n$  or RBUF $n$  address for each access.

When transmitting, DMA/CPU must write data to each serializer configured as "active" and "transmit" within each time slot. Failure to do so results in a buffer underrun condition ([Section 22.3.10.4.2](#)). Similarly when receiving, data must be read from each serializer configured as "active" and "receive" within each time slot. Failure to do results in a buffer overrun condition ([Section 22.3.10.4.3](#)).

To perform internal transfers through the configuration bus, set XBUSEL/RBUSEL bit to 1 in the respective XFMT/RFMT registers.

### 22.3.10.1.4 Using the CPU for McASP Servicing

The CPU can be used to service the McASP through interrupt (upon AXINT/ARINT interrupts) or through polling the XDATA bit in the XSTAT register. As discussed in [Section 22.3.10.1.2](#) and [Section 22.3.10.1.3](#), the CPU can access through either the data port or through the configuration bus.

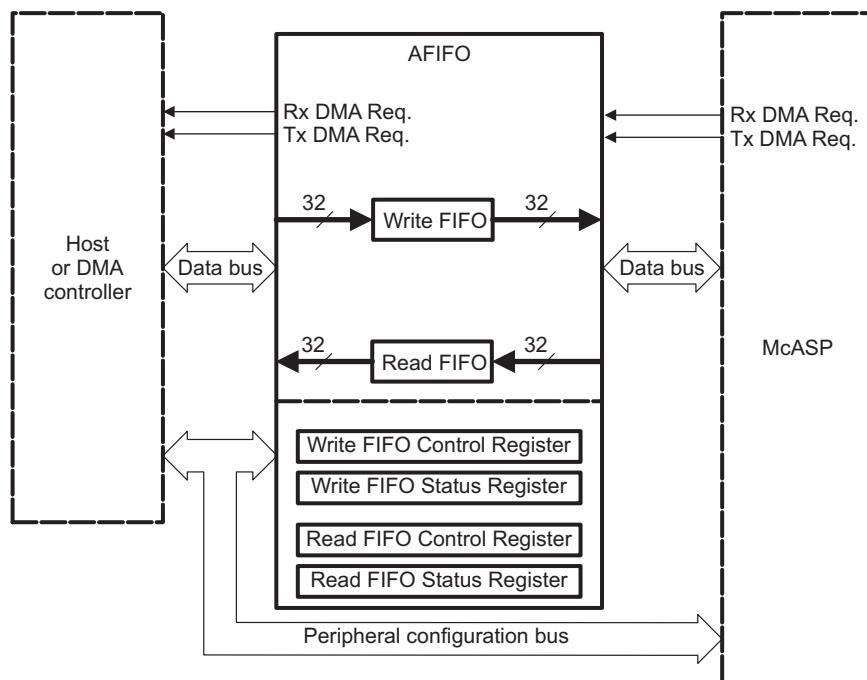
To use the CPU to service the McASP through interrupts, the XDATA/RDATA bit must be enabled in the respective XINTCTL/RINTCTL registers, to generate interrupts AXINT/ARINT to the CPU upon data ready.

### 22.3.10.2 McASP Audio FIFO (AFIFO)

The AFIFO contains two FIFOs: one Read FIFO (RFIFO), and one Write FIFO (WFIFO). To ensure backward compatibility with existing software, both the Read and Write FIFOs are disabled by default. See [Figure 22-28](#) for a high-level block diagram of the AFIFO.

The AFIFO may be enabled/disabled and configured via the WFIFOCTL and RFIFOCTL registers. Note that if the Read or Write FIFO is to be enabled, it must be enabled prior to initializing the receive/transmit section of the McASP (see [Section 22.3.12.2](#) for details).

**Figure 22-28. McASP Audio FIFO (AFIFO) Block Diagram**



#### 22.3.10.2.1 AFIFO Data Transmission

When the Write FIFO is disabled, transmit DMA requests pass through directly from the McASP to the host/DMA controller. Whether the WFIFO is enabled or disabled, the McASP generates transmit DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Write FIFO is enabled, transmit DMA requests from the McASP are sent to the AFIFO, which in turn generates transmit DMA requests to the host/DMA controller.

If the Write FIFO is enabled, upon a transmit DMA request from the McASP, the WFIFO writes *WNUMDMA* 32-bit words to the McASP if and when there are at least *WNUMDMA* words in the Write FIFO. If there are not, the WFIFO waits until this condition has been satisfied. At that point, it writes *WNUMDMA* words to the McASP. (See description for WFIFOCTL.WNUMDMA in [Section 22.4.1.44](#).)

If the host CPU writes to the Write FIFO, independent of a transmit DMA request, the WFIFO will accept host writes until full. After this point, excess data will be discarded.

Note that when the WFIFO is first enabled, it will immediately issue a transmit DMA request to the host. This is because it begins in an empty state, and is therefore ready to accept data.

### 22.3.10.2.1.1 Transmit DMA Event Pacer

The AFIFO may be configured to delay making a transmit DMA request to the host until the Write FIFO has enough space for a specified number of words. In this situation, the number of transmit DMA requests to the host or DMA controller is reduced.

If the Write FIFO has space to accept *RNUMEVT* 32-bit words, it generates a transmit DMA request to the host and then waits for a response. Once *RNUMEVT* words have been written to the FIFO, it checks again to see if there is space for *RNUMEVT* 32-bit words. If there is space, it generates another transmit DMA request to the host, and so on. In this fashion, the Write FIFO will attempt to stay filled.

Note that if transmit DMA event pacing is desired, WFIFOCTL.WNUMEVT should be set to a non-zero integer multiple of the value in WFIFOCTL.WNUMDMA. If transmit DMA event pacing is not desired, then the value in WFIFOCTL.WNUMEVT should be set equal to the value in WFIFOCTL.WNUMDMA.

### 22.3.10.2.2 AFIFO Data Reception

When the Read FIFO is disabled, receive DMA requests pass through directly from McASP to the host/DMA controller. Whether the RFIFO is enabled or disabled, the McASP generates receive DMA requests as needed; the AFIFO is “invisible” to the McASP.

When the Read FIFO is enabled, receive DMA requests from the McASP are sent to the AFIFO, which in turn generates receive DMA requests to the host/DMA controller.

If the Read FIFO is enabled and the McASP makes a receive DMA request, the RFIFO reads *RNUMDMA* 32-bit words from the McASP, if and when the RFIFO has space for *RNUMDMA* words. If it does not, the RFIFO waits until this condition has been satisfied; at that point, it reads *RNUMDMA* words from the McASP. (See description for RFIFOCTL.RNUMDMA in [Section 22.4.1.46](#).)

If the host CPU reads the Read FIFO, independent of a receive DMA request, and the RFIFO at that time contains less than *RNUMEVT* words, those words will be read correctly, emptying the FIFO.

### 22.3.10.2.2.1 Receive DMA Event Pacer

The AFIFO may be configured to delay making a receive DMA request to the host until the Read FIFO contains a specified number of words. In this situation, the number of receive DMA requests to the host or DMA controller is reduced.

If the Read FIFO contains at least *RNUMEVT* 32-bit words, it generates a receive DMA request to the host and then waits for a response. Once *RNUMEVT* 32-bit words have been read from the RFIFO, the RFIFO checks again to see if it contains at least another *RNUMEVT* words. If it does, it generates another receive DMA request to the host, and so on. In this fashion, the Read FIFO will attempt to stay empty.

Note that if receive DMA event pacing is desired, RFIFOCTL.RNUMEVT should be set to a non-zero integer multiple of the value in RFIFOCTL.RNUMDMA. If receive DMA event pacing is not desired, then the value in RFIFOCTL.RNUMEVT should be set equal to the value in RFIFOCTL.RNUMDMA.

### 22.3.10.2.3 Arbitration Between Transmit and Receive DMA Requests

If both the WFIFO and the RFIFO are enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the WFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the transmit DMA request. Once a transfer is in progress, it is allowed to complete.

If only the RFIFO is enabled and a transmit DMA request and receive DMA request occur simultaneously, priority is given to the receive DMA request. Once a transfer is in progress, it is allowed to complete.

## 22.3.10.3 Formatter

### 22.3.10.3.1 Transmit Bit Stream Data Alignment

The McASP transmitter supports serial formats of:

- Slot (or Time slot) size = 8, 12, 16, 20, 24, 28, 32 bits.
- Word size  $\leq$  Slot size.
- Alignment: when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad.
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot.
- Order: order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB.
  - LSB: least-significant bit of word is shifted out last, last bit is MSB.

Hardware support for these serial formats comes from the programmable options in the transmit bitstream format register (XFMT):

- XRVRS: bit reverse (1) or no bit reverse (0).
- XROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits.
- XSSZ: transmit slot size of 8, 12, 16, 20, 24, 28, or 32 bits.

XSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the XROT field.

[Table 22-8](#) and [Figure 22-29](#) show the XRVRS and XROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size (SLOT in [Table 22-8](#)) and word size (WORD in [Table 22-8](#)) options are multiples of 4, since the transmit rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

The transmit bit mask/pad unit operates on data as an initial step of the transmit format unit (see [Figure 22-24](#)), and the data is aligned in the same representation as it is written to the transmitter by the processor (typically Q31 or integer).

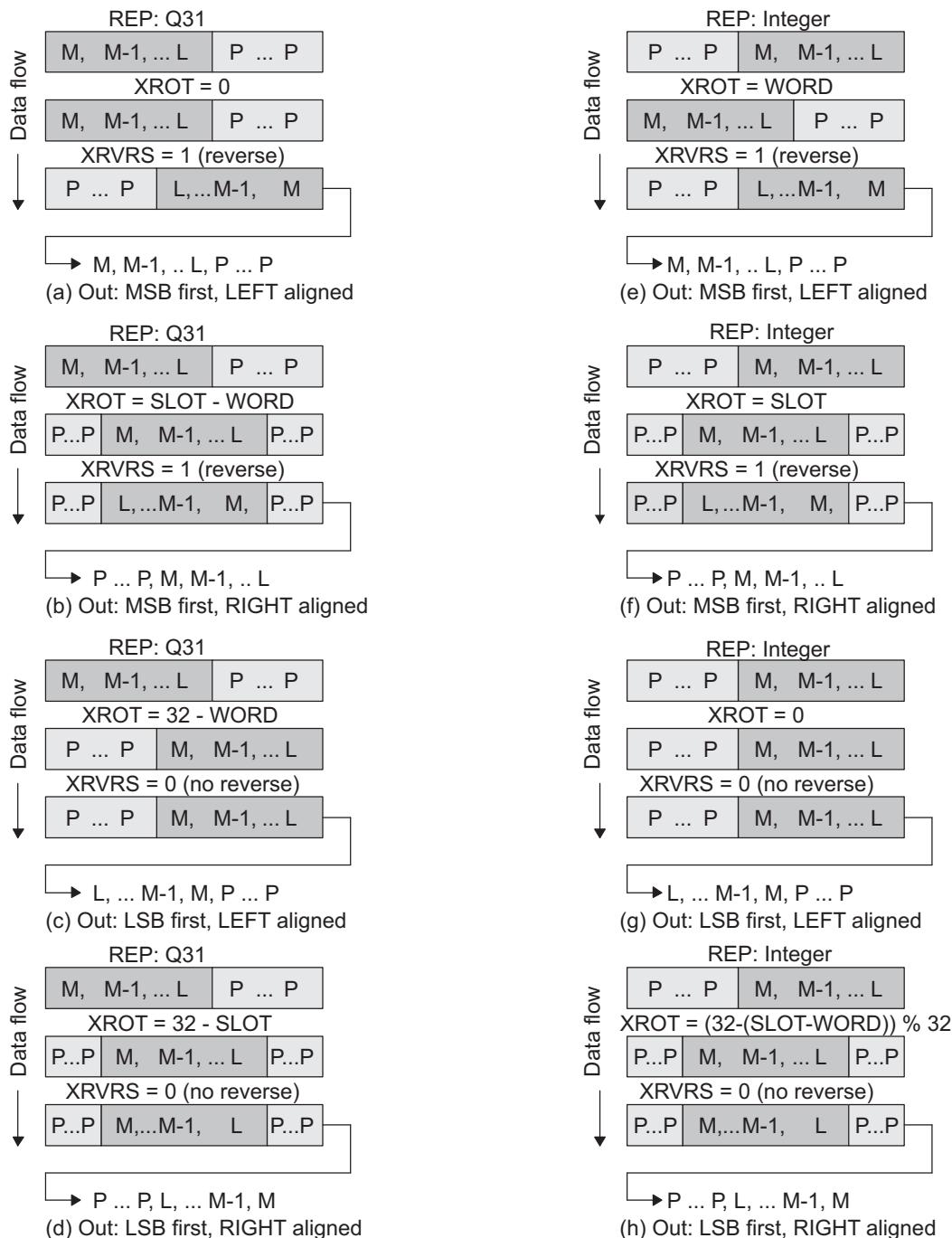
**Table 22-8. Transmit Bitstream Data Alignment**

| <a href="#">Figure 22-29</a> | Bit Stream Order | Bit Stream Alignment | Internal Numeric Representation | XFMT Bit                  |       |
|------------------------------|------------------|----------------------|---------------------------------|---------------------------|-------|
|                              |                  |                      |                                 | XROT <sup>(1)</sup>       | XRVRS |
| (a) <sup>(2)</sup>           | MSB first        | Left aligned         | Q31 fraction                    | 0                         | 1     |
| (b)                          | MSB first        | Right aligned        | Q31 fraction                    | SLOT - WORD               | 1     |
| (c)                          | LSB first        | Left aligned         | Q31 fraction                    | 32 - WORD                 | 0     |
| (d)                          | LSB first        | Right aligned        | Q31 fraction                    | 32 - SLOT                 | 0     |
| (e) <sup>(2)</sup>           | MSB first        | Left aligned         | Integer                         | WORD                      | 1     |
| (f)                          | MSB first        | Right aligned        | Integer                         | SLOT                      | 1     |
| (g)                          | LSB first        | Left aligned         | Integer                         | 0                         | 0     |
| (h)                          | LSB first        | Right aligned        | Integer                         | (32 - (SLOT - WORD)) % 32 | 0     |

<sup>(1)</sup> WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

<sup>(2)</sup> To transmit in I2S format, use MSB first, left aligned, and also select XDATDLY = 01 (1 bit delay)

Figure 22-29. Data Flow Through Transmit Format Unit, Illustrated



### 22.3.10.3.2 Receive Bit Stream Data Alignment

The McASP receiver supports serial formats of:

- Slot or time slot size = 8, 12, 16, 20, 24, 28, 32 bits.
- Word size  $\leq$  Slot size.
- Alignment when more bits/slot than bits/words, then:
  - Left aligned = word shifted first, remaining bits are pad.
  - Right aligned = pad bits are shifted first, word occupies the last bits in slot.
- Order of bits shifted out:
  - MSB: most-significant bit of word is shifted out first, last bit is LSB.
  - LSB: least-significant bit of word is shifted out last, last bit is MSB.

Hardware support for these serial formats comes from the programmable options in the receive bitstream format register (RFMT):

- RRVRS: bit reverse (1) or no bit reverse (0).
- RROT: rotate right by 0, 4, 8, 12, 16, 20, 24, or 28 bits.
- RSSZ: receive slot size of 8, 12, 16, 20, 24, 28, or 32 bits.

RSSZ should always be programmed to match the slot size of the serial stream. The word size is not directly programmed into the McASP, but rather is used to determine the rotation needed in the RROT field.

[Table 22-9](#) and [Figure 22-30](#) show the RRVRS and RROT fields for each serial format and for both integer and Q31 fractional internal representations.

This discussion assumes that all slot size and word size options are multiples of 4; since the receive rotate right unit only supports rotation by multiples of 4. However, the bit mask/pad unit does allow for any number of significant digits. For example, a Q31 number may have 19 significant digits (word) and be transmitted in a 24-bit slot; this would be formatted as a word size of 20 bits and a slot size of 24 bits. However, it is possible to set the bit mask unit to only pass the 19 most-significant digits (program the mask value to FFFF E000h). The digits that are not significant can be set to a selected pad value, which can be any one of the significant digits, a fixed value of 0, or a fixed value of 1.

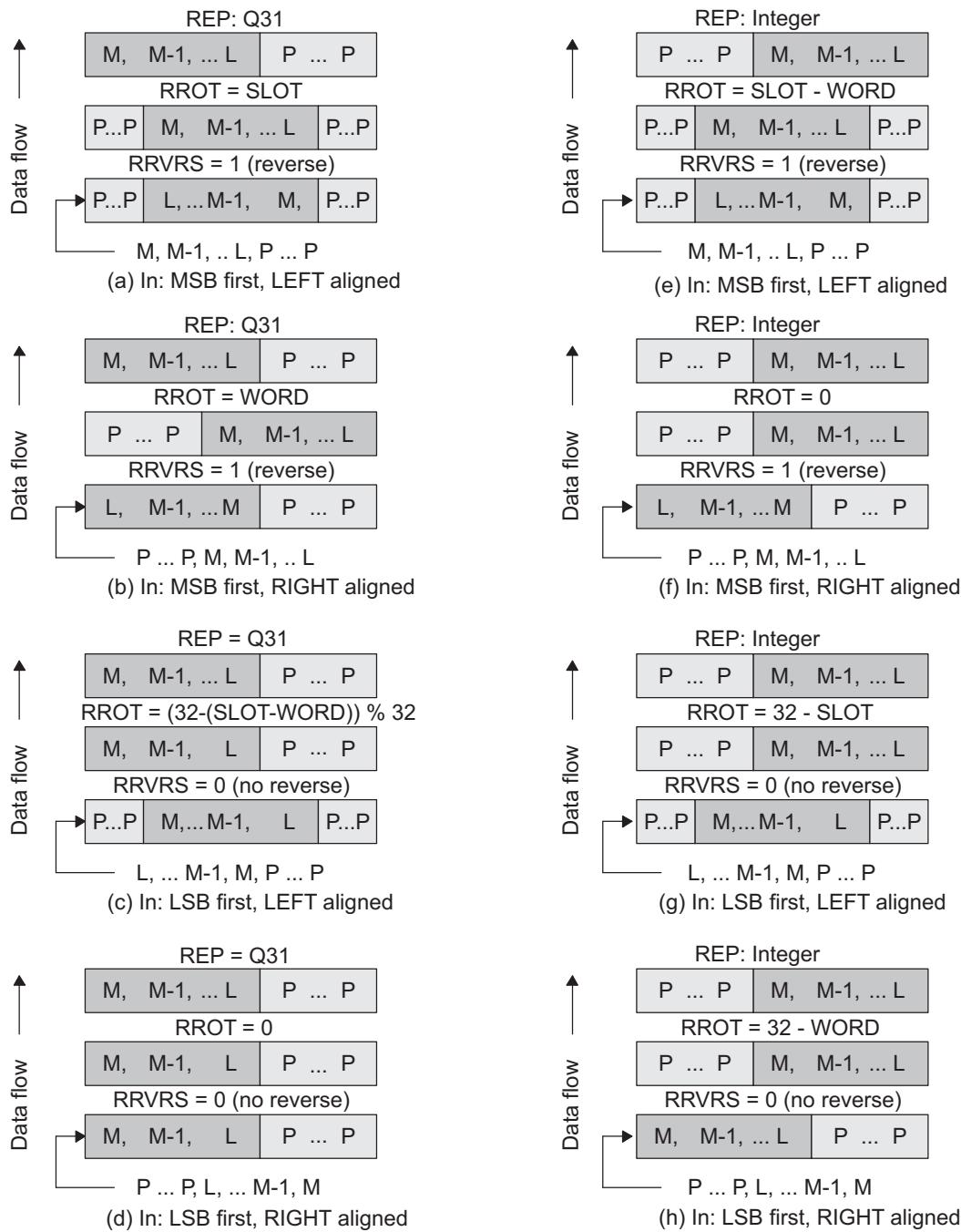
The receive bit mask/pad unit operates on data as the final step of the receive format unit (see [Figure 22-23](#)), and the data is aligned in the same representation as it is read from the receiver by the processor (typically Q31 or integer).

**Table 22-9. Receive Bitstream Data Alignment**

| <a href="#">Figure 22-30</a> | Bit Stream Order | Bit Stream Alignment | Internal Numeric Representation | RFMT Bit                  |       |
|------------------------------|------------------|----------------------|---------------------------------|---------------------------|-------|
|                              |                  |                      |                                 | RROT <sup>(1)</sup>       | RRVRS |
| (a) <sup>(2)</sup>           | MSB first        | Left aligned         | Q31 fraction                    | SLOT                      | 1     |
| (b)                          | MSB first        | Right aligned        | Q31 fraction                    | WORD                      | 1     |
| (c)                          | LSB first        | Left aligned         | Q31 fraction                    | (32 - (SLOT - WORD)) % 32 | 0     |
| (d)                          | LSB first        | Right aligned        | Q31 fraction                    | 0                         | 0     |
| (e) <sup>(2)</sup>           | MSB first        | Left aligned         | Integer                         | SLOT - WORD               | 1     |
| (f)                          | MSB first        | Right aligned        | Integer                         | 0                         | 1     |
| (g)                          | LSB first        | Left aligned         | Integer                         | 32 - SLOT                 | 0     |
| (h)                          | LSB first        | Right aligned        | Integer                         | 32 - WORD                 | 0     |

<sup>(1)</sup> WORD = Word size rounded up to the nearest multiple of 4; SLOT = slot size; % = modulo operator

<sup>(2)</sup> To transmit in I2S format, select MSB first, left aligned, and also select RDATDLY = 01 (1 bit delay)

**Figure 22-30. Data Flow Through Receive Format Unit, Illustrated**


### 22.3.10.4 Error Handling and Management

To support the design of a robust audio system, the McASP includes error-checking capability for the serial protocol, data underrun, and data overrun. In addition, the McASP includes a timer that continually measures the high-frequency master clock every 32 AHCLKX/AHCLKR clock cycles. The timer value can be read to get a measurement of the clock frequency and has a minimum and maximum range setting that can set an error flag if the master clock goes out of a specified range.

Upon the detection of any one or more errors (software selectable), or the assertion of the AMUTEIN input pin, the AMUTE output pin may be asserted to a high or low level to immediately mute the audio output. In addition, an interrupt may be generated if desired, based on any one or more of the error sources.

#### 22.3.10.4.1 Unexpected Frame Sync Error

An unexpected frame sync occurs when:

- In burst mode, when the next active edge of the frame sync occurs early such that the current slot will not be completed by the time the next slot is scheduled to begin.
- In TDM mode, a further constraint is that the frame sync must occur exactly during the correct bit clock (not a cycle earlier or later) and only before slot 0. An unexpected frame sync occurs if this condition is not met.

When an unexpected frame sync occurs, there are two possible actions depending upon when the unexpected frame sync occurs:

1. Early: An early unexpected frame sync occurs when the McASP is in the process of completing the current frame and a new frame sync is detected (not including overlap that occurs due to a 1 or 2 bit frame sync delay). When an early unexpected frame sync occurs:
  - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
  - Current frame is not resynchronized. The number of bits in the current frame is completed. The next frame sync, which occurs after the current frame is completed, will be resynchronized.
2. Late: A late unexpected frame sync occurs when there is a gap or delay between the last bit of the previous frame and the first bit of the next frame. When a late unexpected frame sync occurs (as soon as the gap is detected):
  - Error interrupt flag is set (XSYNCERR, if an unexpected transmit frame sync occurs; RSYNCERR, if an unexpected receive frame sync occurs).
  - Resynchronization occurs upon the arrival of the next frame sync.

Late frame sync is detected the same way in both burst mode and TDM mode; however, in burst mode, late frame sync is not meaningful and its interrupt enable should not be set.

#### 22.3.10.4.2 Buffer Underrun Error - Transmitter

A buffer underrun can only occur for serializers programmed to be transmitters. A buffer underrun occurs when the serializer is instructed by the transmit state machine to transfer data from XRBUF[n] to XRSR[n], but XRBUF[n] has not yet been written with new data since the last time the transfer occurred. When this occurs, the transmit state machine sets the XUNDRN flag.

An underrun is checked only once per time slot. The XUNDRN flag is set when an underrun condition occurs. Once set, the XUNDRN flag remains set until the processor explicitly writes a 1 to the XUNDRN bit to clear the XUNDRN bit.

In DIT mode, a pair of BMC zeros is shifted out when an underrun occurs (four bit times at  $128 \times fs$ ). By shifting out a pair of zeros, a clock may be recovered on the receiver. To recover, reset the McASP and start again with the proper initialization.

In TDM mode, during an underrun case, a long stream of zeros are shifted out causing the DACs to mute. To recover, reset the McASP and start again with the proper initialization.

#### 22.3.10.4.3 Buffer Overrun Error - Receiver

A buffer overrun can only occur for serializers programmed to be receivers. A buffer overrun occurs when the serializer is instructed to transfer data from XRSR[n] to XRBUF[n], but XRBUF[n] has not yet been read by either the DMA or the processor. When this occurs, the receiver state machine sets the ROVRN flag. However, the individual serializer writes over the data in the XRBUF[n] register (destroying the previous sample) and continues shifting.

An overrun is checked only once per time slot. The ROVRN flag is set when an overrun condition occurs. It is possible that an overrun occurs on one time slot but then the processor catches up and does not cause an overrun on the following time slots. However, once the ROVRN flag is set, it remains set until the processor explicitly writes a 1 to the ROVRN bit to clear the ROVRN bit.

#### 22.3.10.4.4 DMA Error - Transmitter

A transmit DMA error, as indicated by the XDMAERR flag in the XSTAT register, occurs when the DMA (or CPU) writes more words to the DAT port of the McASP than it should. For each DMA event, the DMA should write exactly as many words as there are serializers enabled as transmitters.

XDMAERR indicates that the DMA (or CPU) wrote too many words to the McASP for a given transmit DMA event. Writing too few words results in a transmit underrun error setting XUNDRN in XSTAT.

While XDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP transmitter and the DMA to resynchronize them.

#### 22.3.10.4.5 DMA Error - Receiver

A receive DMA error, as indicated by the RDMAERR flag in the RSTAT register, occurs when the DMA (or CPU) reads more words from the DAT port of the McASP than it should. For each DMA event, the DMA should read exactly as many words as there are serializers enabled as receivers.

RDMAERR indicates that the DMA (or CPU) read too many words from the McASP for a given receive DMA event. Reading too few words results in a receiver overrun error setting ROVRN in RSTAT.

While RDMAERR occurs infrequently, an occurrence indicates a serious loss of synchronization between the McASP and the DMA or CPU. You should reinitialize both the McASP receiver and the DMA to resynchronize them.

### 22.3.10.4.6 Clock Failure Detection

#### 22.3.10.4.6.1 Clock-Failure Check Startup

It is expected, initially, that the clock-failure circuits will generate an error until at least one measurement has been taken. Therefore, the clock failure interrupts, clock switch, and mute functions should not immediately be enabled, but be enabled only after a specific startup procedure. The startup procedure is:

1. For the transmit clock failure check:
  - (a) Configure transmit clock failure detect logic (XMIN, XMAX, XPS) in the transmit clock check control register (XCLKCHK).
  - (b) Clear transmit clock failure flag (XCKFAIL) in the transmit status register (XSTAT).
  - (c) Wait until first measurement is taken (> 32 AHCLKX clock periods).
  - (d) Verify no clock failure is detected.
  - (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
  - (f) After the transmit clock is measured and falls within the acceptable range, the following may be enabled:
    - (i) transmit clock failure interrupt enable bit (XCKFAIL) in the transmitter interrupt control register (XINTCTL).
    - (ii) transmit clock failure detect autoswitch enable bit (XCKFAILSW) in the transmit clock check control register (XCLKCHK).
    - (iii) mute option (XCKFAIL) in the mute control register (AMUTE).
2. For the receive clock failure check:
  - (a) Configure receive clock failure detect logic (RMIN, RMAX, RPS) in the receive clock check control register (RCLKCHK).
  - (b) Clear receive clock failure flag (RCKFAIL) in the receive status register (RSTAT).
  - (c) Wait until first measurement is taken (> 32 AHCLKR clock periods).
  - (d) Verify no clock failure is detected.
  - (e) Repeat steps b–d until clock is running and is no longer issuing clock failure errors.
  - (f) After the receive clock is measured and falls within the acceptable range, the following may be enabled:
    - (i) receive clock failure interrupt enable bit (RCKFAIL) in the receiver interrupt control register (RINTCTL).
    - (ii) mute option (RCKFAIL) in the mute control register (AMUTE).

### 22.3.10.4.6.2 Transmit Clock Failure Check and Recovery

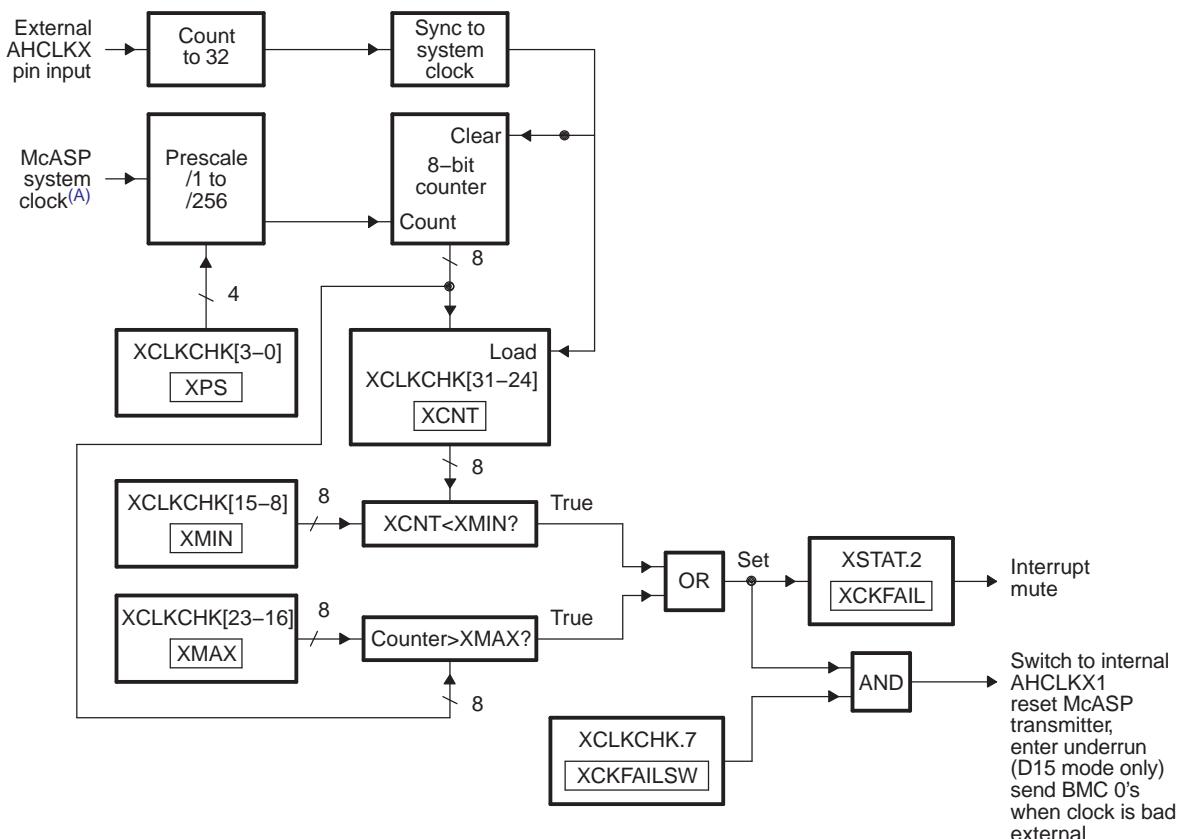
The transmit clock failure check circuit (Figure 22-31) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKX). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKX) periods, and stores the count in XCNT of the transmit clock check control register (XCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (XMIN), and automatically flags an interrupt (XCKFAIL in XSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than XMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (XMAX). This is in case the external clock completely stops, so that the counter value is not copied to XCNT. An out-of-range maximum condition occurs when the count is greater than XMAX. Note that the XMIN and XMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected, or that the audio source has changed and a new sample rate is being used.

In order for the transmit clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKX is internally generated or externally sourced.

**Figure 22-31. Transmit Clock Failure Detection Circuit Block Diagram**



A Refer to device data manual for the McASP system clock source. This is not the same as AUXCLK.

The following actions are taken if a clock failure is detected:

1. Transmit clock failure flag (XCKFAIL) in XSTAT is set. This causes an interrupt if transmit clock failure interrupt enable bit (XCKFAIL) in XINTCTL is set.

In addition (only supported for DIT mode), if the transmit clock failure detect autoswitch enable bit (XCKFAILSW) in XCLKCHK is set, the following additional steps are taken to change the clock source from external to internal:

1. High-frequency transmit clock source bit (HCLKXM) in AHCLKXCTL is set to 1 and internal serial clock divider is selected. However, AHCLKX pin direction does not change to an output while XCKFAIL is set.
2. The internal clock divider is reset, so that the next clock it produces is a full period. However, the transmit clock divide ratio bits (HCLKXDIV) in AHCLKXCTL are not affected, so the internal clock divider generates clocks at the rate configured.
3. The transmit section is reset for a single serial clock period.
4. The transmit section is released from reset and attempts to begin transmitting. If data is available, it begins transmitting immediately; otherwise, it enters the underrun state. An initial underrun is certain to occur, the pattern 1100 (BMC zeroes) should be shifted out initially.

To change back to an external clock, take the following actions:

1. Wait for the external clock to stabilize again. This can be checked by polling the transmit clock count (XCNT) in XCLKCHK.
2. Reset the transmit section according to the startup procedure in [Section 22.3.10.4.6.1](#).

### 22.3.10.4.6.3 Receive Clock Failure Check and Recovery

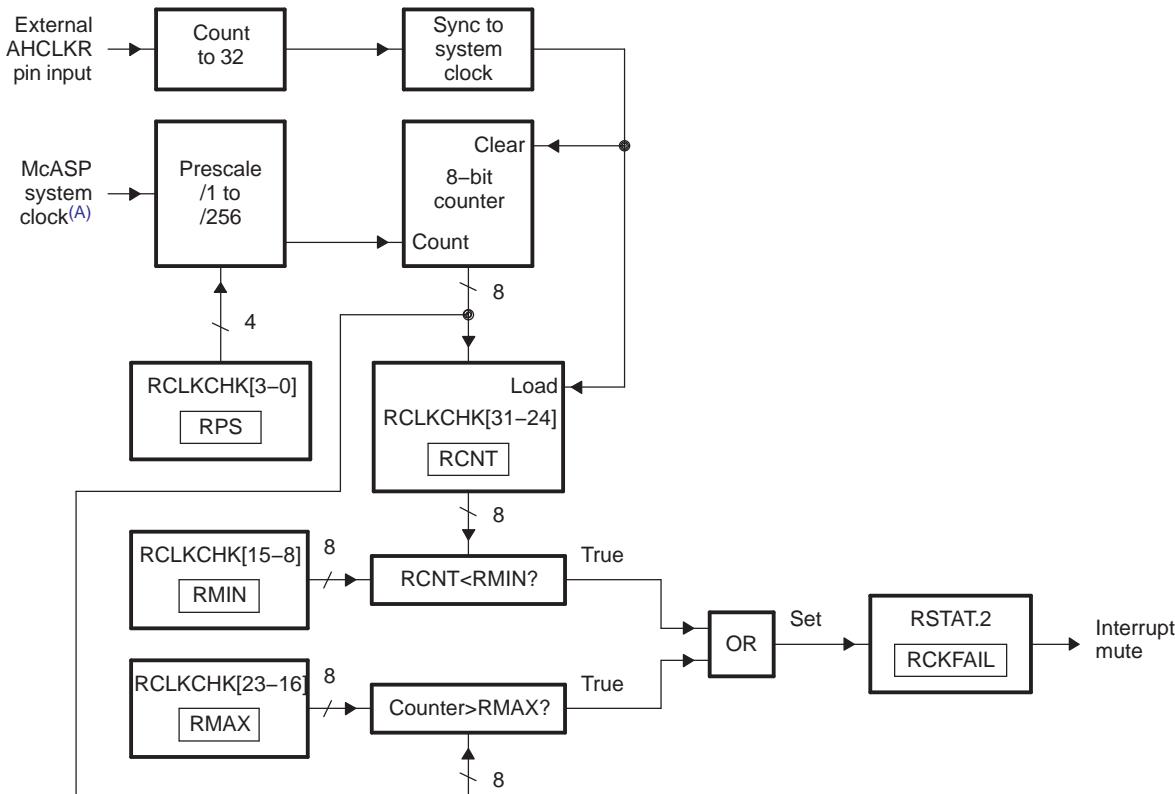
The receive clock failure check circuit (Figure 22-32) works off both the internal McASP system clock and the external high-frequency serial clock (AHCLKR). It continually counts the number of system clocks for every 32 high rate serial clock (AHCLKR) periods, and stores the count in RCNT of the receive clock check control register (RCLKCHK) every 32 high rate serial clock cycles.

The logic compares the count against a user-defined minimum allowable boundary (RMIN) and automatically flags an interrupt (RCKFAIL in RSTAT) when an out-of-range condition occurs. An out-of-range minimum condition occurs when the count is smaller than RMIN. The logic continually compares the current count (from the running system clock counter) against the maximum allowable boundary (RMAX). This is in case the external clock completely stops, so that the counter value is not copied to RCNT. An out-of-range maximum condition occurs when the count is greater than RMAX. Note that the RMIN and RMAX fields are 8-bit unsigned values, and the comparison is performed using unsigned arithmetic.

An out-of-range count may indicate either that an unstable clock was detected or that the audio source has changed and a new sample rate is being used.

In order for the receive clock failure check circuit to operate correctly, the high-frequency serial clock divider must be taken out of reset regardless if AHCLKR is internally generated or externally sourced.

**Figure 22-32. Receive Clock Failure Detection Circuit Block Diagram**



A Refer to device data manual for the McASP system clock source. This is not the same as AUXCLK.

### 22.3.10.5 Loopback Modes

The McASP features a digital loopback mode (DLB) that allows testing of the McASP code in TDM mode with a single processor device. In loopback mode, output of the transmit serializers is connected internally to the input of the receive serializers. Therefore, you can check the receive data against the transmit data to ensure that the McASP settings are correct. Digital loopback mode applies to TDM mode only (2 to 32 slots in a frame). It does not apply to DIT mode ( $XMOD = 180h$ ) or burst mode ( $XMOD = 0$ ).

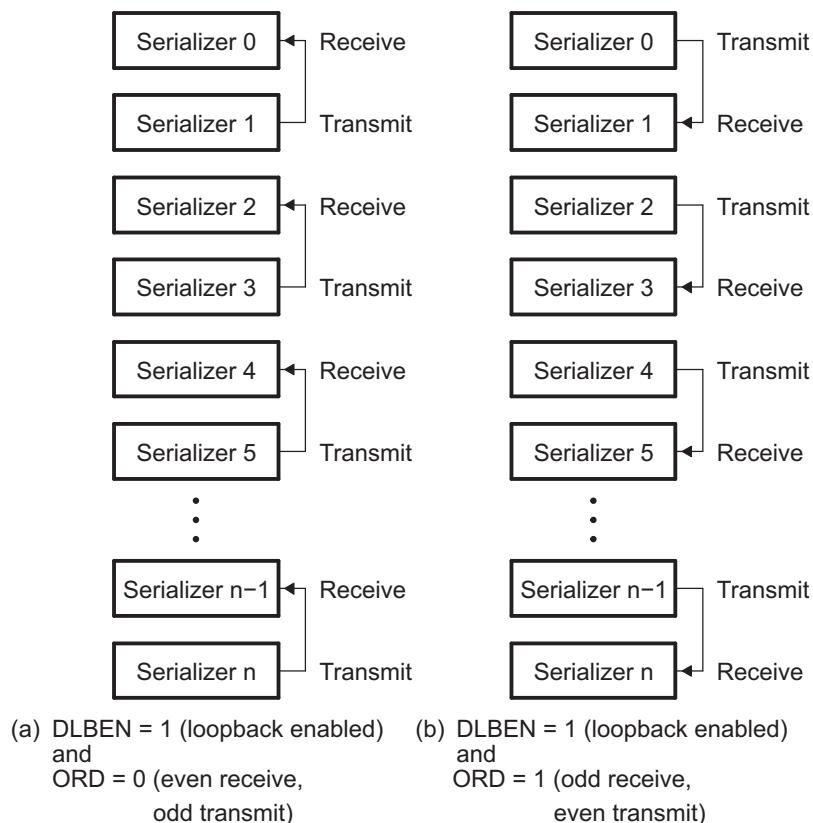
[Figure 22-33](#) shows the basic logical connection of the serializers in loopback mode. Two types of loopback connections are possible, selected by the ORD bit in the digital loopback control register (DLBCTL) as follows:

- ORD = 0: Outputs of odd serializers are connected to inputs of even serializers. If this mode is selected, you should configure odd serializers to be transmitters and even serializers to be receivers.
- ORD = 1: Outputs of even serializers are connected to inputs of odd serializers. If this mode is selected, you should configure even serializers to be transmitters and odd serializers to be receivers.

Data can be externally visible at the I/O pin of the transmit serializer if the pin is configured as a McASP output pin by setting the corresponding PFUNC bit to 0 and PDIR bit to 1.

In loopback mode, the transmit clock and frame sync are used by both the transmit and receive sections of the McASP. The transmit and receive sections operate synchronously. This is achieved by setting the MODE bit of the DLBCTL register to 01b and the ASYNC bit of the ACLKXCTL register to 0.

**Figure 22-33. Serializers in Loopback Mode**



### 22.3.10.5.1 Loopback Mode Configurations

This is a summary of the settings required for digital loopback mode for TDM format:

- The DLBEN bit in DLBCTL must be set to 1 to enable loopback mode.
- The MODE bits in DLBCTL must be set to 01b for both the transmit and receive sections to use the transmit clock and frame sync generator.
- The ORD bit in DLBCTL must be programmed appropriately to select odd or even serializers to be transmitters or receivers. The corresponding serializers must be configured accordingly.
- The ASYNC bit in ACLKXCTL must be cleared to 0 to ensure synchronous transmit and receive operations.
- RMOD field in AFSRCTL and XMOD field in AFSXCTL must be set to 2h to 20h to indicate TDM mode. Loopback mode does not apply to DIT or burst mode.

### 22.3.11 Reset Considerations

The McASP has two reset sources: software reset and hardware reset.

#### 22.3.11.1 Software Reset Considerations

The transmitter and receiver portions of the McASP may be put in reset through the global control register (GBLCTL). Note that a valid serial clock must be supplied to the desired portion of the McASP (transmit and/or receive) in order to assert the software reset bits in GBLCTL. see [Section 22.3.12.2](#)for details on how to ensure reset has occurred.

The entire McASP module may also be reset through the Power and Sleep Controller (PSC). Note that from the McASP perspective, this reset appears as a hardware reset to the entire module.

#### 22.3.11.2 Hardware Reset Considerations

When the McASP is reset due to device reset, the entire serial port (including the transmitter and receiver state machines, and other registers) is reset.

### 22.3.12 Setup and Initialization

This section discusses steps necessary to use the McASP module.

#### 22.3.12.1 Considerations When Using a McASP

The following is a list of things to be considered for systems using a McASP:

##### 22.3.12.1.1 Clocks

For each receive and transmit section:

- External or internal generated bit clock and high frequency clock?
- If internally generated, what is the bit clock speed and the high frequency clock speed?
- Clock polarity?
- External or internal generated frame sync?
- If internally generated, what is frame sync speed?
- Frame sync polarity?
- Frame sync width?
- Transmit and receive sync or asynchronous?

##### 22.3.12.1.2 Data Pins

For each pin of each McASP:

- McASP or GPIO?
- Input or output?

### 22.3.12.1.3 Data Format

For each transmit and receive data:

- Internal numeric representation (integer, Q31 fraction)?
- I2S or DIT (transmit only)?
- Time slot delay (0, 1, or 2 bit)?
- Alignment (left or right)?
- Order (MSB first, LSB first)?
- Pad (if yes, pad with what value)?
- Slot size?
- Rotate?
- Mask?

### 22.3.12.1.4 Data Transfers

- Internal: DMA or CPU?
- External: TDM or burst?
- Bus: configuration bus (CFG) or data port (DAT)?

## 22.3.12.2 Transmit/Receive Section Initialization

You must follow the following steps to properly configure the McASP. If external clocks are used, they should be present prior to the following initialization steps.

1. Reset McASP to default values by setting GBLCTL = 0.
2. Configure all McASP registers except GBLCTL in the following order:
  - (a) Power Idle SYSCONFIG: PWRIDLESYSCONFIG.
  - (b) Receive registers: RMASK, RFMT, AFSRCTL, ACLKRCTL, AHCLKRCTL, RTDM, RINTCTL, RCLKCHK. If external clocks AHCLKR and/or ACLKR are used, they must be running already for proper synchronization of the GBLCTL register.
  - (c) Transmit registers: XMASK, XFMT, AFSXCTL, ACLKXCTL, AHCLKXCTL, XTDM, XINTCTL, XCLKCHK. If external clocks AHCLKX and/or ACLKX are used, they must be running already for proper synchronization of the GBLCTL register.
  - (d) Serializer registers: SRCTL[n].
  - (e) Global registers: Registers PFUNC, PDIR, DITCTL, DLBCTL, AMUTE. Note that PDIR should only be programmed after the clocks and frames are set up in the steps above. This is because the moment a clock pin is configured as an output in PDIR, the clock pin starts toggling at the rate defined in the corresponding clock control register. Therefore you must ensure that the clock control register is configured appropriately before you set the pin to be an output. A similar argument applies to the frame sync pins. Also note that the reset state for the transmit high-frequency clock divide register (HCLKXDIV) is divide-by-1, and the divide-by-1 clocks are not gated by the transmit high-frequency clock divider reset enable (XHCLKRST).
  - (f) DIT registers: For DIT mode operation, set up registers DITCSRA[n], DITCSR[n], DITUDRA[n], and DITUDRB[n].
3. Start the respective high-frequency serial clocks AHCLKX and/or AHCLKR. This step is necessary even if external high-frequency serial clocks are used:
  - (a) Take the respective internal high-frequency serial clock divider(s) out of reset by setting the RHCLKRST bit for the receiver and/or the XHCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be held at 0.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.

4. Start the respective serial clocks ACLKX and/or ACLKR. This step can be skipped if external serial clocks are used and they are running:
  - (a) Take the respective internal serial clock divider(s) out of reset by setting the RCLKRST bit for the receiver and/or the XCLKRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
5. Setup data acquisition as required:
  - (a) If DMA is used to service the McASP, set up data acquisition as desired and start the DMA in this step, before the McASP is taken out of reset.
  - (b) If CPU interrupt is used to service the McASP, enable the transmit and/ or receive interrupt as required.
  - (c) If CPU polling is used to service the McASP, no action is required in this step.
6. Activate serializers.
  - (a) Before starting, clear the respective transmitter and receiver status registers by writing XSTAT = FFFFh and RSTAT = FFFFh.
  - (b) Take the respective serializers out of reset by setting the RSRCRLR bit for the receiver and/or the XSRCLR bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (c) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
7. Verify that all transmit buffers are serviced. Skip this step if the transmitter is not used. Also, skip this step if time slot 0 is selected as inactive (special cases, see [Figure 22-21](#), second waveform). As soon as the transmit serializer is taken out of reset, XDATA in the XSTAT register is set, indicating that XBUF is empty and ready to be serviced. The XDATA status causes an DMA event AXEVT to be generated, and can cause an interrupt AXINT to be generated if it is enabled in the XINTCTL register.
  - (a) If DMA is used to service the McASP, the DMA automatically services the McASP upon receiving AXEVT. Before proceeding in this step, you should verify that the XDATA bit in the XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the DMA.
  - (b) If CPU interrupt is used to service the McASP, interrupt service routine is entered upon the AXINT interrupt. The interrupt service routine should service the XBUF registers. Before proceeding in this step, you should verify that the XDATA bit in XSTAT is cleared to 0, indicating that all transmit buffers are already serviced by the CPU.
  - (c) If CPU polling is used to service the McASP, the XBUF registers should be written to in this step.
8. Release state machines from reset.
  - (a) Take the respective state machine(s) out of reset by setting the RSMRST bit for the receiver and/or the XSMRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
9. Release frame sync generators from reset. Note that it is necessary to release the internal frame sync generators from reset, even if an external frame sync is being used, because the frame sync error detection logic is built into the frame sync generator.
  - (a) Take the respective frame sync generator(s) out of reset by setting the RFRST bit for the receiver, and/or the XFRST bit for the transmitter in GBLCTL. All other bits in GBLCTL should be left at the previous state.
  - (b) Read back from GBLCTL to ensure the bit(s) to which you wrote are successfully latched in GBLCTL before you proceed.
10. Upon the first frame sync signal, McASP transfers begin. The McASP synchronizes to an edge on the frame sync pin, not the level on the frame sync pin. This makes it easy to release the state machine and frame sync generators from reset.
  - (a) For example, if you configure the McASP for a rising edge transmit frame sync, then you do not need to wait for a low level on the frame sync pin before releasing the McASP transmitter state

machine and frame sync generators from reset.

### 22.3.12.3 Separate Transmit and Receive Initialization

In many cases, it is desirable to separately initialize the McASP transmitter and receiver. For example, you may delay the initialization of the transmitter until the type of data coming in on the receiver is recognized. Or a change in the incoming data stream on the receiver may necessitate a reinitialization of the transmitter.

In this case, you may still follow the sequence outlined in [Section 22.3.12.2](#), but use it for each section (transmit, receive) individually. The GBLCTL register is aliased to RGLCTL and XGBLCTL to facilitate separate initialization of transmit and receive sections.

### 22.3.12.4 Importance of Reading Back GBLCTL

In [Section 22.3.12.2](#), steps 3b, 4b, 6c, 8b, and 9b state that GBLCTL should be read back until the bits that were written are successfully latched. This is important, because the transmitter and receiver state machines run off of the respective bit clocks, which are typically about tens to hundreds of times slower than the processor's internal bus clock. Therefore, it takes many cycles between when the processor writes to GBLCTL (or RGLCTL and XGBLCTL), and when the McASP actually recognizes the write operation. If you skip this step, then the McASP may never see the reset bits in the global control registers get asserted and de-asserted; resulting in an uninitialized McASP.

Therefore, the logic in McASP has been implemented such that once the processor writes GBLCTL, RGLCTL, or XGBLCTL, the resulting write is not visible by reading back GBLCTL until the McASP has recognized the change. This typically requires two bit clocks plus two processor bus clocks to occur.

Also, if the bit clocks can be completely stopped, any software that polls GBLCTL should be implemented with a time-out. If GBLCTL does not have a time-out, and the bit clock stops, the changes written to GBLCTL will not be reflected until the bit clock restarts.

Finally, please note that while RGLCTL and XGBLCTL allow separate changing of the receive and transmit halves of GBLCTL, they also immediately reflect the updated value (useful for debug purposes). Only GBLCTL can be used for the read back step.

### 22.3.12.5 Synchronous Transmit and Receive Operation (ASYNC = 0)

When ASYNC = 0 in ACLKXCTL, the transmit and receive sections operate synchronously from the transmit section clock and transmit frame sync signals ([Figure 22-17](#)). The receive section may have a different (but compatible in terms of slot size) data format.

When ASYNC = 0, the receive frame sync generator is internally disabled. If the AFSX pin is configured as an output, it serves as the frame sync signal for both transmit and receive. The AFSR pin should not be used because the transmit frame sync generator output, which is used by both the transmitter and the receiver when ASYNC = 0, is not propagated to the AFSR pin ([Figure 22-19](#)).

When ASYNC = 0, the transmit and receive sections must share some common settings, since they both use the same clock and frame sync signals:

- DITEN = 0 in DITCTL (TDM mode is enabled).
- The total number of bits per frame must be the same (that is, RSSZ × RMOD must equal XSSZ × XMOD).
- Both transmit and receive should either be specified as burst or TDM mode, but not mixed.
- The settings in ACLKRCTL are irrelevant.
- FSXM must match FSRM.
- FXWID must match FRWID.

For all other settings, the transmit and receive sections may be programmed independently.

### 22.3.12.6 Asynchronous Transmit and Receive Operation (ASYNC = 1)

When ASYNC = 1 in ACLKXCTL, the transmit and receive sections operate completely independently and have separate clock and frame sync signals ([Figure 22-17](#), [Figure 22-18](#), and [Figure 22-19](#)). The events generated by each section come asynchronously.

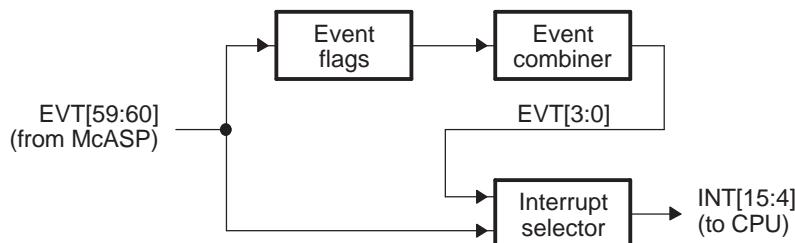
## 22.3.13 Interrupts

### 22.3.13.1 Interrupt Multiplexing

The processor includes an interrupt controller (INTC) to manage CPU interrupts. The INTC maps the device events to 12 CPU interrupts. The McASP generates 4 interrupts to the processor.

The event inputs can be routed to 12 maskable interrupts to the CPU (INT[15:4]). The INTC interrupt selector allows the McASP system events to be routed to any of the 12 CPU interrupt inputs. Furthermore, the INTC provides status flags and allows the combination of events, as shown in [Figure 22-34](#). You must properly configure the INTC by enabling, masking, and routing the McASP system events to the desired CPU interrupts.

**Figure 22-34. Interrupt Multiplexing**



### 22.3.13.2 Transmit Data Ready Interrupt

The transmit data ready interrupt (XDATA) is generated if XDATA is 1 in the XSTAT register and XDATA is also enabled in XINTCTL. [Section 22.3.10.1.1](#) provides details on when XDATA is set in the XSTAT register.

A transmit start of frame interrupt (XSTAFRM) is triggered by the recognition of transmit frame sync. A transmit last slot interrupt (XLAST) is a qualified version of the data ready interrupt (XDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data requested belonging to the last slot (the slot that just ended was next-to-last TDM slot, current slot is last slot).

### 22.3.13.3 Receive Data Ready Interrupt

The receive data ready interrupt (RDATA) is generated if RDATA is 1 in the RSTAT register and RDATA is also enabled in RINTCTL. [Section 22.3.10.1.2](#) provides details on when RDATA is set in the RSTAT register.

A receiver start of frame interrupt (RSTAFRM) is triggered by the recognition of a receiver frame sync. A receiver last slot interrupt (RLAST) is a qualified version of the data ready interrupt (RDATA). It has the same behavior as the data ready interrupt, but is further qualified by having the data in the buffer come from the last TDM time slot (the slot that just ended was last TDM slot).

### 22.3.13.4 Error Interrupts

Upon detection, the following error conditions generate interrupt flags:

- In the receive status register (RSTAT):
  - Receiver overrun (ROVRN).
  - Unexpected receive frame sync (RSYNCERR).
  - Receive clock failure (RCKFAIL).
  - Receive DMA error (RDMAERR).

- In the transmit status register (XSTAT):
  - Transmit underrun (XUNDRN).
  - Unexpected transmit frame sync (XSYNCERR).
  - Transmit clock failure (XCKFAIL).
  - Transmit DMA error (XDMAERR).

Each interrupt source also has a corresponding enable bit in the receive interrupt control register (RINTCTL) and transmit interrupt control register (XINTCTL). If the enable bit is set in RINTCTL or XINTCTL, an interrupt is requested when the interrupt flag is set in RSTAT or XSTAT. If the enable bit is not set, no interrupt request is generated. However, the interrupt flag may be polled.

#### 22.3.13.5 Audio Mute (AMUTE) Function

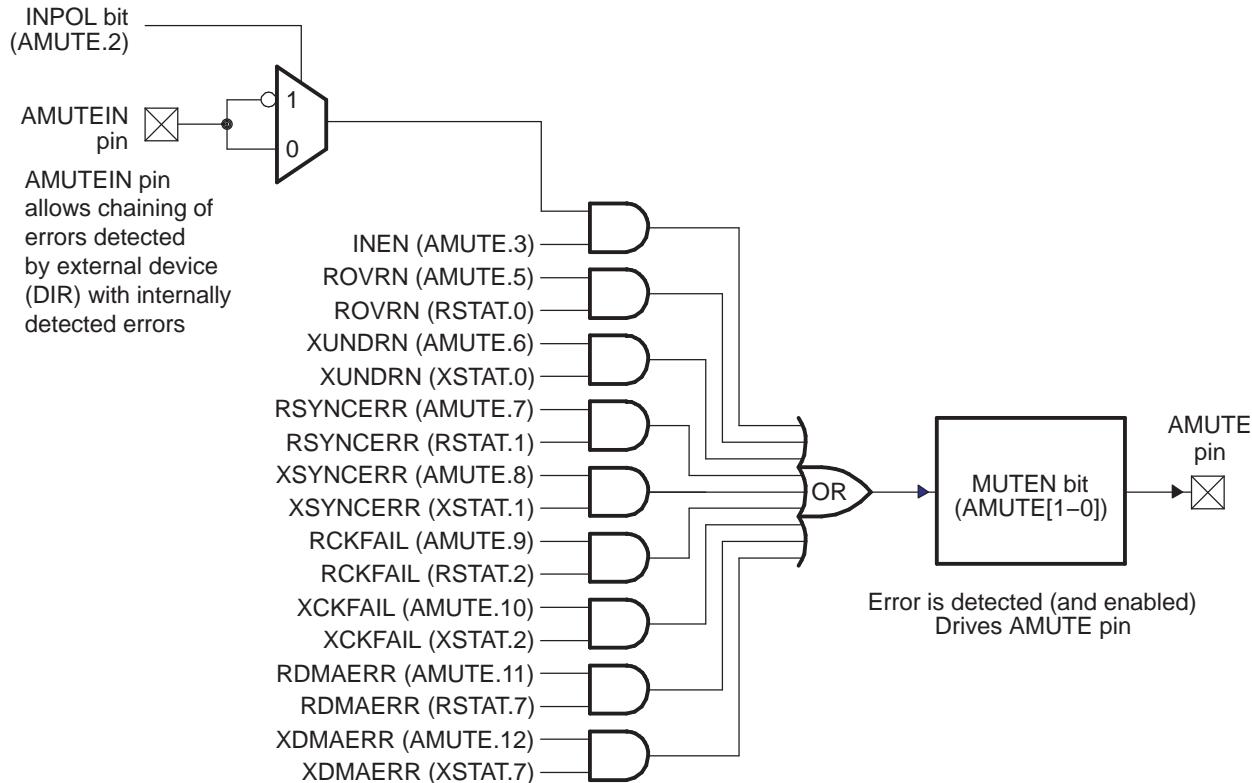
The McASP includes an automatic audio mute function (Figure 22-35) that asserts in hardware the AMUTE pin to a preprogrammed output state, as selected by the MUTEN bit in the audio mute control register (AMUTE). The AMUTE pin is asserted when one of the interrupt flags is set or an external device issues an error signal on the AMUTEIN input. Typically, the AMUTEIN input is shared with a device interrupt pin (for example EXT\_INT4).

The AMUTEIN input allows the on-chip logic to consider a mute input from other devices in the system, so that all errors may be considered. The AMUTEIN input has a programmable polarity to allow it to adapt to different devices, as selected by the INPOL bit in AMUTE, and it must be enabled explicitly.

In addition to the external AMUTEIN input, the AMUTE pin output may be asserted when one of the error interrupt flags is set and its mute function is enabled in AMUTE.

When one or more of the errors is detected and enabled, the AMUTE pin is driven to an active state that is selected by MUTEN in AMUTE. The active polarity of the AMUTE pin is programmable by MUTEN (and the inactive polarity is the opposite of the active polarity). The AMUTE pin remains driven active until software clears all the error interrupt flags that are enabled to mute, and until the AMUTEIN is inactive.

**Figure 22-35. Audio Mute (AMUTE) Block Diagram**



### 22.3.13.6 Multiple Interrupts

This only applies to interrupts and not to DMA requests. The following terms are defined:

- **Active Interrupt Request:** a flag in RSTAT or XSTAT is set and the interrupt is enabled in RINTCTL or XINTCTL.
- **Outstanding Interrupt Request:** An interrupt request has been issued on one of the McASP transmit/receive interrupt ports, but that request has not yet been serviced.
- **Serviced:** The CPU writes to RSTAT or XSTAT to clear one or more of the active interrupt request flags.

The first interrupt request to become active for the transmitter with the interrupt flag set in XSTAT and the interrupt enabled in XINTCTL generates a request on the McASP transmit interrupt port AXINT.

If more than one interrupt request becomes active in the same cycle, a single interrupt request is generated on the McASP transmit interrupt port. Subsequent interrupt requests that become active while the first interrupt request is outstanding do not immediately generate a new request pulse on the McASP transmit interrupt port.

The transmit interrupt is serviced with the CPU writing to XSTAT. If any interrupt requests are active after the write, a new request is generated on the McASP transmit interrupt port.

The receiver operates in a similar way, but using RSTAT, RINTCTL, and the McASP receive interrupt port ARINT.

One outstanding interrupt request is allowed on each port, so a transmit and a receive interrupt request may both be outstanding at the same time.

### 22.3.14 EDMA Event Support

#### 22.3.14.1 EDMA Events

There are 6 EDMA events.

#### 22.3.14.2 Using the DMA for McASP Servicing

The most typical scenario is to use the DMA to service the McASP through the data port, although the DMA can also service the McASP through the configuration bus. Two possibilities exist for using the DMA events to service the McASP:

1. Use **AXEVT/AREVT**: Triggered upon each XDATA/RDATA transition from 0 to 1.
2. Use **AXEVTO/AREVTO** and **AXEVTE/AREVTE**: Alternating AXEVT/AREVT events for odd/even slots. Upon AXEVT/AREVT, AXEVTO/AREVTO is triggered if the event is for an odd channel, and AXEVTE/AREVTE is triggered if the event is for an even channel.

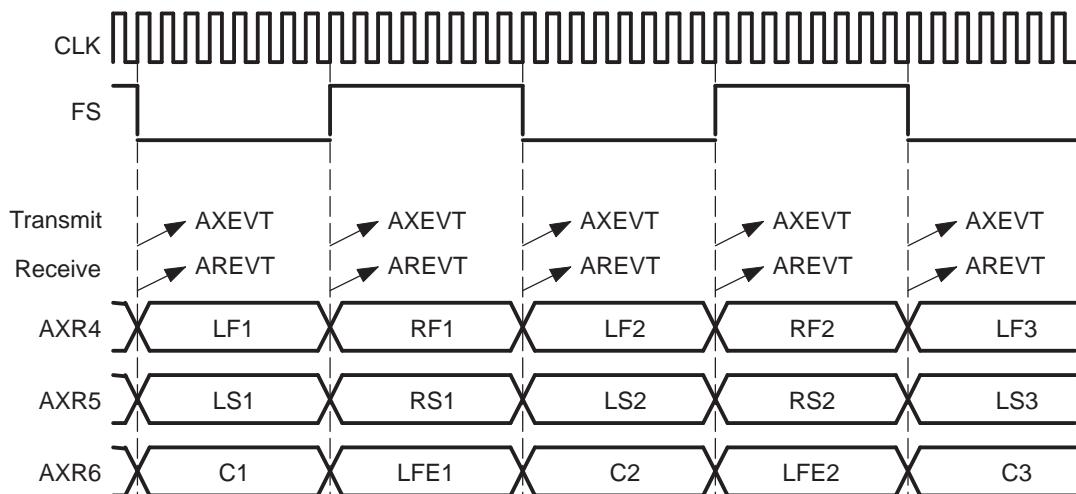
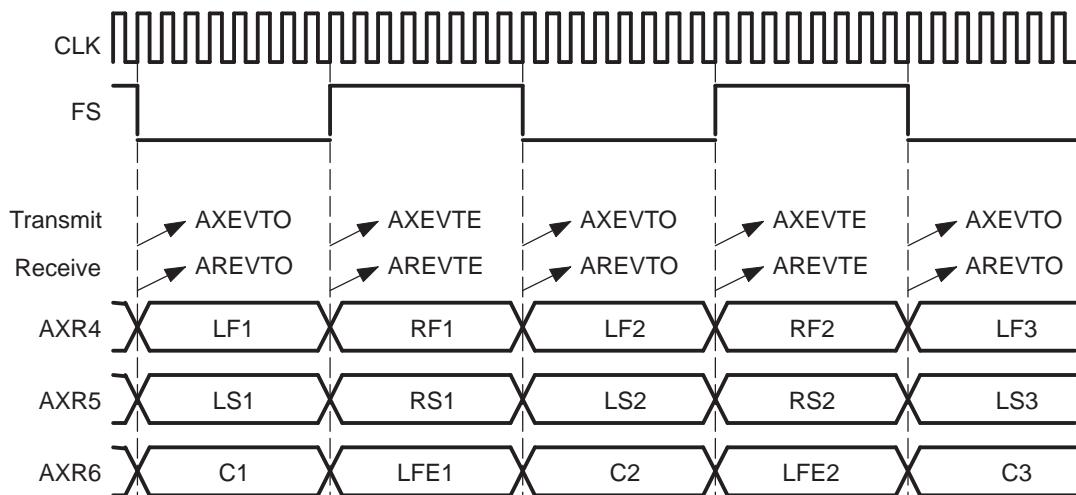
---

**NOTE:** Check the device-specific data manual to see if AXEVTO/AREVTO and AXEVTE/AREVTE are supported. These are optional.

---

[Figure 22-36](#) and [Figure 22-37](#) show an example audio system with six audio channels (LF, RF, LS, RS, C, and LFE) transmitted from three AXRn pins on the McASP. [Figure 22-36](#) and [Figure 22-37](#) show when events AXEVT, AXEVTO, and AXEVTE are triggered. [Figure 22-36](#) and [Figure 22-37](#) also apply for the receive audio channels and show when events AREVT, AREVTO, and AREVTE are triggered.

You can either use the DMA to service the McASP upon events AXEVT and AREVT ([Figure 22-36](#)) or upon events AXEVTO, AREVTO, AXEVTE, and AREVTE ([Figure 22-37](#)).

**Figure 22-36. DMA Events in an Audio Example—Two Events (Scenario 1)**

**Figure 22-37. DMA Events in an Audio Example—Four Events (Scenario 2)**


In scenario 1 (Figure 22-36), a DMA event AXEVT/AREVT is triggered on each time slot. In the example, AXEVT is triggered for each of the transmit audio channel time slot (Time slot for channels LF, LS, and C; and time slot for channels RF, RS, LFE). Similarly, AREVT is triggered for each of the receive audio channel time slot. Scenario 1 allows for the use of a single DMA to transmit all audio channels, and a single DMA to receive all audio channels.

In scenario 2 (Figure 22-37), two alternating DMA events are triggered for each time slot. In the example, AXEVTE (even) is triggered for the time slot for the even audio channels (LF, LS, C) and AXEVTO (odd) is triggered for the time slot for the odd audio channels (RF, RS, LFE). AXEVTO and AXEVTE alternate in time. The same is true in the receive direction with the use of AREVTO and AREVTE. This scenario allows for the use of two DMA channels (odd and even) to transmit all audio channels, and two DMA channels to receive all audio channels.

Here are some guidelines on using the different DMA events:

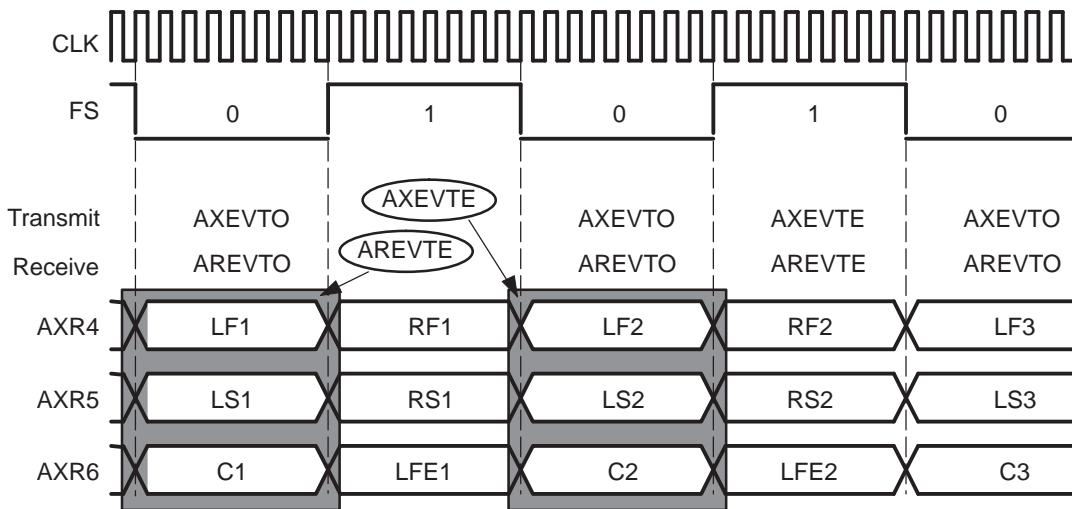
- Either use AXEVT, or the combination of AXEVTO and AXEVTE, to service the McASP. Never use all three at the same time. Similarly for receive, either use AREVT, or the combination of AREVTO and AREVTE.
- The McASP generates transmit DMA events independently from receive DMA events; therefore, separate schemes can be used for transmit and receive DMA. For example, scenario 1 could be used for the transmit data (AXEVT) and scenario 2 could be used for the receive data (AREVTO, AREVTE), and conversely.

Note the difference between DMA event generation and the CPU interrupt generation. DMA events are generated automatically upon data ready; whereas CPU interrupt generation needs to be enabled in the XINTCTL/RINTCTL register.

In [Figure 22-37](#), scenario 2, each transmit DMA request is for data in the next time slot, while each receive DMA request is for data in the previous time slot. For example, [Figure 22-38](#) shows a circled AXEVTE event for an even time slot transmit DMA request. The transmitter always requests a DMA transfer for data it will need to transmit during the next time slot. So, in this example, the circled event AXEVTE is a request for data for samples LF2, LS2, and C2.

On the other hand, the circled AREVTE event is an even time slot receive DMA request. The receiver always requests a DMA transfer for data it received during the previous time slot. In this example, the circled event AREVTE is a request for samples LF1, LS1, and C1.

**Figure 22-38. DMA Events in an Audio Example**



### 22.3.15 Power Management

The McASP can be placed in reduced power modes to conserve power during periods of low activity.

### 22.3.16 Emulation Considerations

---

**NOTE:** The receive buffer registers (RBUF $n$ ) and transmit buffer registers (XBUF $n$ ) should not be accessed by the emulator when the McASP is running. Such an access will cause the RRDY/XRDY bit in the serializer control register  $n$  (SRCTL $n$ ) to be updated.

---

The McASP does not support the emulation suspend.

## 22.4 McASP Registers

### 22.4.1 McASP CFG Registers

Control registers for the McASP are summarized in [Table 22-10](#). The control registers are accessed through the configuration bus of the device. The receive buffer registers (RBUF $n$ ) and transmit buffer registers (XBUF $n$ ) can also be accessed through the data port of the device, as listed in [Table 22-53](#).

Control registers for the McASP Audio FIFO (AFIFO) are summarized in [Table 22-11](#). Note that the AFIFO Write FIFO (WFIFO) and Read FIFO (RFIFO) have independent control and status registers. The AFIFO control registers are accessed through the peripheral configuration port.

**Table 22-10. McASP Registers Accessed Through Configuration Bus**

| Offset | Acronym           | Register Description   | Section                           |
|--------|-------------------|--|-----------------------------------|
| 0h     | REV               | Revision identification register   | <a href="#">Section 22.4.1.1</a>  |
| 04h    | PWRIDLESYS CONFIG | Power Idle SYSCONFIG Register  | <a href="#">Section 22.4.1.2</a>  |
| 10h    | PFUNC             | Pin function register  | <a href="#">Section 22.4.1.3</a>  |
| 14h    | PDIR              | Pin direction register   | <a href="#">Section 22.4.1.4</a>  |
| 18h    | PDOUT             | Pin data output register   | <a href="#">Section 22.4.1.5</a>  |
| 1Ch    | PDIN              | Read returns: Pin data input register  | <a href="#">Section 22.4.1.6</a>  |
| 1Ch    | PDSET             | Writes affect: Pin data set register (alternate write address: PDOUT)  | <a href="#">Section 22.4.1.7</a>  |
| 20h    | PDCLR             | Pin data clear register (alternate write address: PDOUT)   | <a href="#">Section 22.4.1.8</a>  |
| 44h    | GBLCTL            | Global control register  | <a href="#">Section 22.4.1.9</a>  |
| 48h    | AMUTE             | Audio mute control register  | <a href="#">Section 22.4.1.10</a> |
| 4Ch    | DLBCTL            | Digital loopback control register  | <a href="#">Section 22.4.1.11</a> |
| 50h    | DITCTL            | DIT mode control register  | <a href="#">Section 22.4.1.12</a> |
| 60h    | RGBLCTL           | Receiver global control register: Alias of GBLCTL, only receive bits are affected - allows receiver to be reset independently from transmitter     | <a href="#">Section 22.4.1.13</a> |
| 64h    | RMASK             | Receive format unit bit mask register  | <a href="#">Section 22.4.1.14</a> |
| 68h    | RFMT              | Receive bit stream format register   | <a href="#">Section 22.4.1.15</a> |
| 6Ch    | AFSRCTL           | Receive frame sync control register  | <a href="#">Section 22.4.1.16</a> |
| 70h    | ACLKRCTL          | Receive clock control register   | <a href="#">Section 22.4.1.17</a> |
| 74h    | AHCLKRCTL         | Receive high-frequency clock control register  | <a href="#">Section 22.4.1.18</a> |
| 78h    | RTDM              | Receive TDM time slot 0-31 register  | <a href="#">Section 22.4.1.19</a> |
| 7Ch    | RINTCTL           | Receiver interrupt control register  | <a href="#">Section 22.4.1.20</a> |
| 80h    | RSTAT             | Receiver status register   | <a href="#">Section 22.4.1.21</a> |
| 84h    | RSLOT             | Current receive TDM time slot register   | <a href="#">Section 22.4.1.22</a> |
| 88h    | RCLKCHK           | Receive clock check control register   | <a href="#">Section 22.4.1.23</a> |
| 8Ch    | REVTCTL           | Receiver DMA event control register  | <a href="#">Section 22.4.1.24</a> |
| A0h    | XGBLCTL           | Transmitter global control register. Alias of GBLCTL, only transmit bits are affected - allows transmitter to be reset independently from receiver | <a href="#">Section 22.4.1.25</a> |
| A4h    | XMASK             | Transmit format unit bit mask register   | <a href="#">Section 22.4.1.26</a> |
| A8h    | XFMT              | Transmit bit stream format register  | <a href="#">Section 22.4.1.27</a> |
| ACh    | AFSXCTL           | Transmit frame sync control register   | <a href="#">Section 22.4.1.28</a> |
| B0h    | ACLXCTL           | Transmit clock control register  | <a href="#">Section 22.4.1.29</a> |
| B4h    | AHCLKXCTL         | Transmit high-frequency clock control register   | <a href="#">Section 22.4.1.30</a> |
| B8h    | XTDM              | Transmit TDM time slot 0-31 register   | <a href="#">Section 22.4.1.31</a> |
| BCh    | XINTCTL           | Transmitter interrupt control register   | <a href="#">Section 22.4.1.32</a> |
| C0h    | XSTAT             | Transmitter status register  | <a href="#">Section 22.4.1.33</a> |
| C4h    | XSLOT             | Current transmit TDM time slot register  | <a href="#">Section 22.4.1.34</a> |
| C8h    | XCLKCHK           | Transmit clock check control register  | <a href="#">Section 22.4.1.35</a> |
| CCh    | XEVCTL            | Transmitter DMA event control register   | <a href="#">Section 22.4.1.36</a> |

**Table 22-10. McASP Registers Accessed Through Configuration Bus (continued)**

| Offset | Acronym  | Register Description  | Section                           |
|--------|----------|---|-----------------------------------|
| 100h   | DITCSRA0 | Left (even TDM time slot) channel status register (DIT mode) 0    | <a href="#">Section 22.4.1.38</a> |
| 104h   | DITCSRA1 | Left (even TDM time slot) channel status register (DIT mode) 1    | <a href="#">Section 22.4.1.38</a> |
| 108h   | DITCSRA2 | Left (even TDM time slot) channel status register (DIT mode) 2    | <a href="#">Section 22.4.1.38</a> |
| 10Ch   | DITCSRA3 | Left (even TDM time slot) channel status register (DIT mode) 3    | <a href="#">Section 22.4.1.38</a> |
| 110h   | DITCSRA4 | Left (even TDM time slot) channel status register (DIT mode) 4    | <a href="#">Section 22.4.1.38</a> |
| 114h   | DITCSRA5 | Left (even TDM time slot) channel status register (DIT mode) 5    | <a href="#">Section 22.4.1.38</a> |
| 118h   | DITCSR0  | Right (odd TDM time slot) channel status register (DIT mode) 0    | <a href="#">Section 22.4.1.39</a> |
| 11Ch   | DITCSR1  | Right (odd TDM time slot) channel status register (DIT mode) 1    | <a href="#">Section 22.4.1.39</a> |
| 120h   | DITCSR2  | Right (odd TDM time slot) channel status register (DIT mode) 2    | <a href="#">Section 22.4.1.39</a> |
| 124h   | DITCSR3  | Right (odd TDM time slot) channel status register (DIT mode) 3    | <a href="#">Section 22.4.1.39</a> |
| 128h   | DITCSR4  | Right (odd TDM time slot) channel status register (DIT mode) 4    | <a href="#">Section 22.4.1.39</a> |
| 12Ch   | DITCSR5  | Right (odd TDM time slot) channel status register (DIT mode) 5    | <a href="#">Section 22.4.1.39</a> |
| 130h   | DITUDRA0 | Left (even TDM time slot) channel user data register (DIT mode) 0 | <a href="#">Section 22.4.1.40</a> |
| 134h   | DITUDRA1 | Left (even TDM time slot) channel user data register (DIT mode) 1 | <a href="#">Section 22.4.1.40</a> |
| 138h   | DITUDRA2 | Left (even TDM time slot) channel user data register (DIT mode) 2 | <a href="#">Section 22.4.1.40</a> |
| 13Ch   | DITUDRA3 | Left (even TDM time slot) channel user data register (DIT mode) 3 | <a href="#">Section 22.4.1.40</a> |
| 140h   | DITUDRA4 | Left (even TDM time slot) channel user data register (DIT mode) 4 | <a href="#">Section 22.4.1.40</a> |
| 144h   | DITUDRA5 | Left (even TDM time slot) channel user data register (DIT mode) 5 | <a href="#">Section 22.4.1.40</a> |
| 148h   | DITUDRB0 | Right (odd TDM time slot) channel user data register (DIT mode) 0 | <a href="#">Section 22.4.1.41</a> |
| 14Ch   | DITUDRB1 | Right (odd TDM time slot) channel user data register (DIT mode) 1 | <a href="#">Section 22.4.1.41</a> |
| 150h   | DITUDRB2 | Right (odd TDM time slot) channel user data register (DIT mode) 2 | <a href="#">Section 22.4.1.41</a> |
| 154h   | DITUDRB3 | Right (odd TDM time slot) channel user data register (DIT mode) 3 | <a href="#">Section 22.4.1.41</a> |
| 158h   | DITUDRB4 | Right (odd TDM time slot) channel user data register (DIT mode) 4 | <a href="#">Section 22.4.1.41</a> |
| 15Ch   | DITUDRB5 | Right (odd TDM time slot) channel user data register (DIT mode) 5 | <a href="#">Section 22.4.1.41</a> |
| 180h   | SRCTL0   | Serializer control register 0                                     | <a href="#">Section 22.4.1.37</a> |
| 184h   | SRCTL1   | Serializer control register 1                                     | <a href="#">Section 22.4.1.37</a> |
| 188h   | SRCTL2   | Serializer control register 2                                     | <a href="#">Section 22.4.1.37</a> |
| 18Ch   | SRCTL3   | Serializer control register 3                                     | <a href="#">Section 22.4.1.37</a> |
| 200h   | XBUFO    | Transmit buffer register for serializer 0                         | <a href="#">Section 22.4.1.42</a> |
| 204h   | XBUF1    | Transmit buffer register for serializer 1                         | <a href="#">Section 22.4.1.42</a> |
| 208h   | XBUF2    | Transmit buffer register for serializer 2                         | <a href="#">Section 22.4.1.42</a> |
| 20Ch   | XBUF3    | Transmit buffer register for serializer 3                         | <a href="#">Section 22.4.1.42</a> |
| 280h   | RBUFO    | Receive buffer register for serializer 0                          | <a href="#">Section 22.4.1.43</a> |
| 284h   | RBUF1    | Receive buffer register for serializer 1                          | <a href="#">Section 22.4.1.43</a> |
| 288h   | RBUF2    | Receive buffer register for serializer 2                          | <a href="#">Section 22.4.1.43</a> |
| 28Ch   | RBUF3    | Receive buffer register for serializer 3                          | <a href="#">Section 22.4.1.43</a> |

**Table 22-11. McASP AFIFO Registers Accessed Through Peripheral Configuration Port**

| Offset | Acronym  | Register Description        | Section                           |
|--------|----------|-----------------------------|-----------------------------------|
| 1000h  | WFIFOCTL | Write FIFO control register | <a href="#">Section 22.4.1.44</a> |
| 1004h  | WFIFOSTS | Write FIFO status register  | <a href="#">Section 22.4.1.45</a> |
| 1008h  | RFIFOCTL | Read FIFO control register  | <a href="#">Section 22.4.1.46</a> |

**Table 22-11. McASP AFIFO Registers Accessed Through Peripheral Configuration Port (continued)**

| Offset | Acronym  | Register Description      | Section                           |
|--------|----------|---------------------------|-----------------------------------|
| 100Ch  | RFIFOSTS | Read FIFO status register | <a href="#">Section 22.4.1.47</a> |

### 22.4.1.1 Revision Identification Register (REV)

The revision identification register (REV) contains identification data for the peripheral. The REV is shown in [Figure 22-39](#) and described in [Table 22-12](#).

**Figure 22-39. Revision Identification Register (REV)**

|    |              |   |
|----|--------------|---|
| 31 | REV          | 0 |
|    | R-4430 0A02h |   |

LEGEND: R = Read only; -n = value after reset

**Table 22-12. Revision Identification Register (REV) Field Descriptions**

| Bit  | Field | Value      | Description                        |
|------|-------|------------|------------------------------------|
| 31-0 | REV   | 4430 0A02h | Identifies revision of peripheral. |

### 22.4.1.2 Power Idle SYSCONFIG Register (PWRIDLESYSCONFIG)

The PWRIDLESYSCONFIG is shown in [Figure 22-40](#) and described in [Table 22-13](#).

**Figure 22-40. Power Idle SYSCONFIG Register (PWRIDLESYSCONFIG)**

| 31       | 30 | 29    | 28 | 27 | 26 | 25       | 24 |
|----------|----|-------|----|----|----|----------|----|
| Reserved |    |       |    |    |    |          |    |
| R-0      |    |       |    |    |    |          |    |
| 23       |    |       |    |    |    |          | 8  |
| Reserved |    |       |    |    |    |          |    |
| R-0      |    |       |    |    |    |          |    |
| 7        | 6  | 5     | 4  | 3  | 2  | 1        | 0  |
| Reserved |    | Other |    |    |    | IDLEMODE |    |
| R-0      |    | R/W-0 |    |    |    | R/W-10   |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-13. Power Idle SYSCONFIG Register (PWRIDLESYSCONFIG) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-6 | Reserved | 0     | Reserved.  |
| 5-2  | Other    | 0     | Reserved for future programming.   |
| 1-0  | IDLEMODE | 10    | Power management Configuration of the local target state management mode.<br>By definition, target can handle read/write transaction as long as it is out of IDLE state.<br>0x0 = Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, i.e. regardless of the IP module's internal requirements. Backup mode, for debug only.<br>0x1 = No-idle mode: local target never enters idle state. Backup mode, for debug only.<br>0x2 = Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-request-related) wakeup events. |

### 22.4.1.3 Pin Function Register (PFUNC)

The pin function register (PFUNC) specifies the function of AXRn, ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either a McASP pin or a general-purpose input/output (GPIO) pin. The PFUNC is shown in [Figure 22-41](#) and described in [Table 22-14](#).

#### CAUTION

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-41. Pin Function Register (PFUNC)**

| 31       | 30       | 29    | 28    | 27     | 26    | 25    | 24       |
|----------|----------|-------|-------|--------|-------|-------|----------|
| AFSR     | AHCLKR   | ACLKR | AFSX  | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       | Reserved |       |       |        |       |       |          |
|          | R-0      |       |       |        |       |       |          |
| 7        | 6        | 5     | 4     | 3      | 2     | 1     | 0        |
| Reserved | AXR5     | AXR4  | AXR3  | AXR2   | AXR1  | AXR0  |          |
| R-0      | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-14. Pin Function Register (PFUNC) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31   | AFSR     | 0     | Determines if AFSR pin functions as McASP or GPIO.<br>Pin functions as McASP pin.  |
|      |          | 1     | Pin functions as GPIO pin.   |
| 30   | AHCLKR   | 0     | Determines if AHCLKR pin functions as McASP or GPIO.<br>Pin functions as McASP pin.  |
|      |          | 1     | Pin functions as GPIO pin.   |
| 29   | ACLKR    | 0     | Determines if ACLKR pin functions as McASP or GPIO.<br>Pin functions as McASP pin.   |
|      |          | 1     | Pin functions as GPIO pin.   |
| 28   | AFSX     | 0     | Determines if AFSX pin functions as McASP or GPIO.<br>Pin functions as McASP pin.  |
|      |          | 1     | Pin functions as GPIO pin.   |
| 27   | AHCLKX   | 0     | Determines if AHCLKX pin functions as McASP or GPIO.<br>Pin functions as McASP pin.  |
|      |          | 1     | Pin functions as GPIO pin.   |
| 26   | ACLKX    | 0     | Determines if ACLKX pin functions as McASP or GPIO.<br>Pin functions as McASP pin.   |
|      |          | 1     | Pin functions as GPIO pin.   |
| 25   | AMUTE    | 0     | Determines if AMUTE pin functions as McASP or GPIO.<br>Pin functions as McASP pin.   |
|      |          | 1     | Pin functions as GPIO pin.   |
| 24-6 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0  | AXR[5-0] | 0     | Determines if AXRn pin functions as McASP or GPIO.<br>Pin functions as McASP pin.  |
|      |          | 1     | Pin functions as GPIO pin.   |

#### 22.4.1.4 Pin Direction Register (PDIR)

The pin direction register (PDIR) specifies the direction of AXRn, ACLKX, AHCLKX, AFSX, ACLKR, AHCLKR, and AFSR pins as either an input or an output pin. The PDIR is shown in [Figure 22-42](#) and described in [Table 22-15](#).

Regardless of the pin function register (PFUNC) setting, each PDIR bit must be set to 1 for the specified pin to be enabled as an output and each PDIR bit must be cleared to 0 for the specified pin to be an input.

For example, if the McASP is configured to use an internally-generated bit clock and the clock is to be driven out to the system, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output).

When AXRn is configured to transmit, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be set to 1 (an output). Similarly, when AXRn is configured to receive, the PFUNC bit must be cleared to 0 (McASP function) and the PDIR bit must be cleared to 0 (an input).

#### CAUTION

##### Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-42. Pin Direction Register (PDIR)**

| 31       | 30       | 29    | 28    | 27     | 26    | 25    | 24       |
|----------|----------|-------|-------|--------|-------|-------|----------|
| AFSR     | AHCLKR   | ACLKR | AFSX  | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       | Reserved |       |       |        |       |       |          |
| 8        | R-0      |       |       |        |       |       |          |
| 7        | 6        | 5     | 4     | 3      | 2     | 1     | 0        |
| Reserved | AXR5     | AXR4  | AXR3  | AXR2   | AXR1  | AXR0  |          |
| R-0      | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-15. Pin Direction Register (PDIR) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31         | AFSR         | 0            | Determines if AFSR pin functions as an input or output.<br>Pin functions as input.   |
|            |              | 1            | Pin functions as output.   |
| 30         | AHCLKR       | 0            | Determines if AHCLKR pin functions as an input or output.<br>Pin functions as input.   |
|            |              | 1            | Pin functions as output.   |
| 29         | ACLKR        | 0            | Determines if ACLKR pin functions as an input or output.<br>Pin functions as input.  |
|            |              | 1            | Pin functions as output.   |
| 28         | AFSX         | 0            | Determines if AFSX pin functions as an input or output.<br>Pin functions as input.   |
|            |              | 1            | Pin functions as output.   |
| 27         | AHCLKX       | 0            | Determines if AHCLKX pin functions as an input or output.<br>Pin functions as input.   |
|            |              | 1            | Pin functions as output.   |
| 26         | ACLKX        | 0            | Determines if ACLKX pin functions as an input or output.<br>Pin functions as input.  |
|            |              | 1            | Pin functions as output.   |
| 25         | AMUTE        | 0            | Determines if AMUTE pin functions as an input or output.<br>Pin functions as input.  |
|            |              | 1            | Pin functions as output.   |
| 24-6       | Reserved     | 0            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0        | AXR[5-0]     | 0            | Determines if AXRn pin functions as an input or output.<br>Pin functions as input.   |
|            |              | 1            | Pin functions as output.   |

### 22.4.1.5 Pin Data Output Register (PDOUT)

The pin data output register (PDOUT) holds a value for data out at all times, and may be read back at all times. The value held by PDOUT is not affected by writing to PDIR and PFUNC. However, the data value in PDOUT is driven out onto the McASP pin only if the corresponding bit in PFUNC is set to 1 (GPIO function) and the corresponding bit in PDIR is set to 1 (output). When reading data, returns the corresponding bit value in PDOUT[n], does not return input from I/O pin; when writing data, writes to the corresponding PDOUT[n] bit. The PDOUT is shown in [Figure 22-43](#) and described in [Table 22-16](#).

PDOUT has these aliases or alternate addresses:

- PDSET - when written to at this address, writing a 1 to a bit in PDSET sets the corresponding bit in PDOUT to 1; writing a 0 has no effect and keeps the bits in PDOUT unchanged.
- PDCLR - when written to at this address, writing a 1 to a bit in PDCLR clears the corresponding bit in PDOUT to 0; writing a 0 has no effect and keeps the bits in PDOUT unchanged.

There is only one set of data out bits, PDOUT[31-0]. The other registers, PDSET and PDCLR, are just different addresses for the same control bits, with different behaviors during writes.

#### CAUTION

##### Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-43. Pin Data Output Register (PDOUT)**

| 31       | 30     | 29    | 28    | 27     | 26    | 25    | 24       |
|----------|--------|-------|-------|--------|-------|-------|----------|
| AFSR     | AHCLKR | ACLKR | AFSX  | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0  | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       |        |       |       |        |       |       | 8        |
| Reserved |        |       |       |        |       |       |          |
| R-0      |        |       |       |        |       |       |          |
| 7        | 6      | 5     | 4     | 3      | 2     | 1     | 0        |
| Reserved |        | AXR5  | AXR4  | AXR3   | AXR2  | AXR1  | AXR0     |
| R-0      |        | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-16. Pin Data Output Register (PDOUT) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31         | AFSR         | 0            | Determines drive on AFSR output pin when the corresponding PFUNC[31] and PDIR[31] bits are set to 1.<br>Pin drives low.  |
|            |              | 1            | Pin drives high.   |
| 30         | AHCLKR       | 0            | Determines drive on AHCLKR output pin when the corresponding PFUNC[30] and PDIR[30] bits are set to 1.<br>Pin drives low.  |
|            |              | 1            | Pin drives high.   |
| 29         | ACLKR        | 0            | Determines drive on ACLKR output pin when the corresponding PFUNC[29] and PDIR[29] bits are set to 1.<br>Pin drives low.   |
|            |              | 1            | Pin drives high.   |
| 28         | AFSX         | 0            | Determines drive on AFSX output pin when the corresponding PFUNC[28] and PDIR[28] bits are set to 1.<br>Pin drives low.  |
|            |              | 1            | Pin drives high.   |
| 27         | AHCLKX       | 0            | Determines drive on AHCLKX output pin when the corresponding PFUNC[27] and PDIR[27] bits are set to 1.<br>Pin drives low.  |
|            |              | 1            | Pin drives high.   |
| 26         | ACLKX        | 0            | Determines drive on ACLKX output pin when the corresponding PFUNC[26] and PDIR[26] bits are set to 1.<br>Pin drives low.   |
|            |              | 1            | Pin drives high.   |
| 25         | AMUTE        | 0            | Determines drive on AMUTE output pin when the corresponding PFUNC[25] and PDIR[25] bits are set to 1.<br>Pin drives low.   |
|            |              | 1            | Pin drives high.   |
| 24-6       | Reserved     | 0            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0        | AXR[5-0]     | 0            | Determines drive on AXR[n] output pin when the corresponding PFUNC[n] and PDIR[n] bits are set to 1.<br>Pin drives low.  |
|            |              | 1            | Pin drives high.   |

### 22.4.1.6 Pin Data Input Register (PDIN)

The pin data input register (PDIN) holds the I/O pin state of each of the McASP pins. PDIN allows the actual value of the pin to be read, regardless of the state of PFUNC and PDIR. The value after reset for registers 1 through 15 and 24 through 31 depends on how the pins are being driven. The PDIN is shown in [Figure 22-44](#) and described in [Table 22-17](#).

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-44. Pin Data Input Register (PDIN)**

| 31       | 30     | 29    | 28       | 27     | 26    | 25    | 24       |
|----------|--------|-------|----------|--------|-------|-------|----------|
| AFSR     | AHCLKR | ACLKR | AFSX     | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0  | R/W-0 | R/W-0    | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       |        |       |          |        |       |       | 8        |
|          |        |       | Reserved |        |       |       |          |
|          |        |       | R-0      |        |       |       |          |
| 7        | 6      | 5     | 4        | 3      | 2     | 1     | 0        |
| Reserved |        | AXR5  | AXR4     | AXR3   | AXR2  | AXR1  | AXR0     |
| R-0      |        | R/W-0 | R/W-0    | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-17. Pin Data Input Register (PDIN) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31         | AFSR         | 0            | Logic level on AFSR pin.<br>Pin is logic low.  |
|            |              | 1            | Pin is logic high.   |
| 30         | AHCLKR       | 0            | Logic level on AHCLKR pin.<br>Pin is logic low.  |
|            |              | 1            | Pin is logic high.   |
| 29         | ACLKR        | 0            | Logic level on ACLKR pin.<br>Pin is logic low.   |
|            |              | 1            | Pin is logic high.   |
| 28         | AFSX         | 0            | Logic level on AFSX pin.<br>Pin is logic low.  |
|            |              | 1            | Pin is logic high.   |
| 27         | AHCLKX       | 0            | Logic level on AHCLKX pin.<br>Pin is logic low.  |
|            |              | 1            | Pin is logic high.   |
| 26         | ACLKX        | 0            | Logic level on ACLKX pin.<br>Pin is logic low.   |
|            |              | 1            | Pin is logic high.   |
| 25         | AMUTE        | 0            | Logic level on AMUTE pin.<br>Pin is logic low.   |
|            |              | 1            | Pin is logic high.   |
| 24-6       | Reserved     | 0            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0        | AXR[5-0]     | 0            | Logic level on AXR[n] pin.<br>Pin is logic low.  |
|            |              | 1            | Pin is logic high.   |

### 22.4.1.7 Pin Data Set Register (PDSET)

The pin data set register (PDSET) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDSET bit sets the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic high on the pin. PDSET is useful for a multitasking system because it allows you to set to a logic high only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDSET is shown in [Figure 22-45](#) and described in [Table 22-18](#).

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-45. Pin Data Set Register (PDSET)**

| 31       | 30       | 29    | 28    | 27     | 26    | 25    | 24       |
|----------|----------|-------|-------|--------|-------|-------|----------|
| AFSR     | AHCLKR   | ACLKR | AFSX  | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       | Reserved |       |       |        |       |       |          |
|          | R-0      |       |       |        |       |       |          |
| 7        | 6        | 5     | 4     | 3      | 2     | 1     | 0        |
| Reserved |          | AXR5  | AXR4  | AXR3   | AXR2  | AXR1  | AXR0     |
| R-0      |          | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-18. Pin Data Set Register (PDSET) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31         | AFSR         | 0            | Allows the corresponding AFSR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[31] bit is set to 1.   |
| 30         | AHCLKR       | 0            | Allows the corresponding AHCLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[30] bit is set to 1.   |
| 29         | ACLKR        | 0            | Allows the corresponding ACLKR bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[29] bit is set to 1.   |
| 28         | AFSX         | 0            | Allows the corresponding AFSX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[28] bit is set to 1.   |
| 27         | AHCLKX       | 0            | Allows the corresponding AHCLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[27] bit is set to 1.   |
| 26         | ACLKX        | 0            | Allows the corresponding ACLKX bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[26] bit is set to 1.   |
| 25         | AMUTE        | 0            | Allows the corresponding AMUTE bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[25] bit is set to 1.   |
| 24-6       | Reserved     | 0            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0        | AXR[5-0]     | 0            | Allows the corresponding AXR[n] bit in PDOUT to be set to a logic high without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[n] bit is set to 1.  |

### 22.4.1.8 Pin Data Clear Register (PDCLR)

The pin data clear register (PDCLR) is an alias of the pin data output register (PDOUT) for writes only. Writing a 1 to the PDCLR bit clears the corresponding bit in PDOUT and, if PFUNC = 1 (GPIO function) and PDIR = 1 (output), drives a logic low on the pin. PDCLR is useful for a multitasking system because it allows you to clear to a logic low only the desired pin(s) within a system without affecting other I/O pins controlled by the same McASP. The PDCLR is shown in [Figure 22-46](#) and described in [Table 22-19](#).

**CAUTION**

Writing to Reserved Bits

Writing a value other than 0 to reserved bits in this register may cause improper device operation.

**Figure 22-46. Pin Data Clear Register (PDCLR)**

| 31       | 30       | 29    | 28    | 27     | 26    | 25    | 24       |
|----------|----------|-------|-------|--------|-------|-------|----------|
| AFSR     | AHCLKR   | ACLKR | AFSX  | AHCLKX | ACLKX | AMUTE | Reserved |
| R/W-0    | R/W-0    | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R-0      |
| 23       | Reserved |       |       |        |       |       |          |
|          | R-0      |       |       |        |       |       |          |
| 7        | 6        | 5     | 4     | 3      | 2     | 1     | 0        |
| Reserved |          | AXR5  | AXR4  | AXR3   | AXR2  | AXR1  | AXR0     |
| R-0      |          | R/W-0 | R/W-0 | R/W-0  | R/W-0 | R/W-0 | R/W-0    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-19. Pin Data Clear Register (PDCLR) Field Descriptions**

| <b>Bit</b> | <b>Field</b> | <b>Value</b> | <b>Description</b>   |
|------------|--------------|--------------|--|
| 31         | AFSR         | 0            | Allows the corresponding AFSR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[31] bit is cleared to 0.   |
| 30         | AHCLKR       | 0            | Allows the corresponding AHCLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[30] bit is cleared to 0.   |
| 29         | ACLKR        | 0            | Allows the corresponding ACLKR bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[29] bit is cleared to 0.   |
| 28         | AFSX         | 0            | Allows the corresponding AFSX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[28] bit is cleared to 0.   |
| 27         | AHCLKX       | 0            | Allows the corresponding AHCLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[27] bit is cleared to 0.   |
| 26         | ACLKX        | 0            | Allows the corresponding ACLKX bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[26] bit is cleared to 0.   |
| 25         | AMUTE        | 0            | Allows the corresponding AMUTE bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.   |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[25] bit is cleared to 0.   |
| 24-6       | Reserved     | 0            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5-0        | AXR[5-0]     | 0            | Allows the corresponding AXR[n] bit in PDOUT to be cleared to a logic low without affecting other I/O pins controlled by the same port.  |
|            |              |              | No effect.   |
|            |              | 1            | PDOUT[n] bit is cleared to 0.  |

### 22.4.1.9 Global Control Register (GBLCTL)

The global control register (GBLCTL) provides initialization of the transmit and receive sections. The GBLCTL is shown in [Figure 22-47](#) and described in [Table 22-20](#).

The bit fields in GBLCTL are synchronized and latched by the corresponding clocks (ACLKX for bits 12-8 and ACLKR for bits 4-0). Before GBLCTL is programmed, you must ensure that serial clocks are running. If the corresponding external serial clocks, ACLKX and ACLKR, are not yet running, you should select the internal serial clock source in AHCLKXCTL, AHCLKRCTL, ACLKXCTL, and ACLKRCTL before GBLCTL is programmed. Also, after programming any bits in GBLCTL you should not proceed until you have read back from GBLCTL and verified that the bits are latched in GBLCTL.

**Figure 22-47. Global Control Register (GBLCTL)**

| 31       | Reserved |       |        |        |          |         |  | 16 |
|----------|----------|-------|--------|--------|----------|---------|--|----|
| R-0      |          |       |        |        |          |         |  |    |
| 15       | 13       | 12    | 11     | 10     | 9        | 8       |  |    |
| Reserved |          | XFRST | XSMRST | XSRCLR | XHCLKRST | XCLKRST |  |    |
| R-0      |          | R/W-0 | R/W-0  | R/W-0  | R/W-0    | R/W-0   |  |    |
| 7        | 5        | 4     | 3      | 2      | 1        | 0       |  |    |
| Reserved |          | RFRST | RSMRST | RSRCLR | RHCLKRST | RCLKRST |  |    |
| R-0      |          | R/W-0 | R/W-0  | R/W-0  | R/W-0    | R/W-0   |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-20. Global Control Register (GBLCTL) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-13 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.  |
| 12    | XFRST    | 0     | Transmit frame sync generator reset enable bit.<br>Transmit frame sync generator is reset.  |
|       |          | 1     | Transmit frame sync generator is active. When released from reset, the transmit frame sync generator begins counting serial clocks and generating frame sync as programmed.   |
| 11    | XSMRST   | 0     | Transmit state machine reset enable bit.<br>Transmit state machine is held in reset. AXRn pin state:<br>If PFUNC[n] = 0 and PDIR[n] = 1; then the serializer drives the AXRn pin to the state specified for inactive time slot (as determined by DISMOD bits in SRCTL).   |
|       |          | 1     | Transmit state machine is released from reset. When released from reset, the transmit state machine immediately transfers data from XRBUF[n] to XRSR[n]. The transmit state machine sets the underrun flag (XUNDRN) in XSTAT, if XRBUF[n] have not been preloaded with data before reset is released. The transmit state machine also immediately begins detecting frame sync and is ready to transmit.<br>Transmit TDM time slot begins at slot 0 after reset is released. |
| 10    | XSRCLR   | 0     | Transmit serializer clear enable bit. By clearing then setting this bit, the transmit buffer is flushed to an empty state (XDATA = 1). If XSMRST = 1, XSRCLR = 1, XDATA = 1, and XBUF is not loaded with new data before the start of the next active time slot, an underrun will occur.<br>Transmit serializers are cleared.   |
|       |          | 1     | Transmit serializers are active. When the transmit serializers are first taken out of reset (XSRCLR changes from 0 to 1), the transmit data ready bit (XDATA) in XSTAT is set to indicate XBUF is ready to be written.  |
| 9     | XHCLKRST | 0     | Transmit high-frequency clock divider reset enable bit.<br>Transmit high-frequency clock divider is held in reset and passes through its input as divide-by-1..   |
|       |          | 1     | Transmit high-frequency clock divider is running.   |
| 8     | XCLKRST  | 0     | Transmit clock divider reset enable bit.<br>Transmit clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.  |
|       |          | 1     | Transmit clock divider is running.  |

**Table 22-20. Global Control Register (GBLCTL) Field Descriptions (continued)**

| Bit | Field    | Value | Description   |
|-----|----------|-------|---|
| 7-5 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                          |
| 4   | RFRST    | 0     | Receive frame sync generator reset enable bit.<br>Receive frame sync generator is reset.  |
|     |          | 1     | Receive frame sync generator is active. When released from reset, the receive frame sync generator begins counting serial clocks and generating frame sync as programmed.   |
| 3   | RSMRST   | 0     | Receive state machine reset enable bit.<br>Receive state machine is held in reset.  |
|     |          | 1     | Receive state machine is released from reset. When released from reset, the receive state machine immediately begins detecting frame sync and is ready to receive.<br>Receive TDM time slot begins at slot 0 after reset is released. |
| 2   | RSRCLR   | 0     | Receive serializer clear enable bit. By clearing then setting this bit, the receive buffer is flushed.<br>Receive serializers are cleared.  |
|     |          | 1     | Receive serializers are active.   |
| 1   | RHCLKRST | 0     | Receive high-frequency clock divider reset enable bit.<br>Receive high-frequency clock divider is held in reset and passes through its input as divide-by-1.  |
|     |          | 1     | Receive high-frequency clock divider is running.  |
| 0   | RCLKRST  | 0     | Receive high-frequency clock divider reset enable bit.<br>Receive clock divider is held in reset. When the clock divider is in reset, it passes through a divide-by-1 of its input.   |
|     |          | 1     | Receive clock divider is running.   |

### 22.4.1.10 Audio Mute Control Register (AMUTE)

The audio mute control register (AMUTE) controls the McASP audio mute (AMUTE) output pin. The value after reset for register 4 depends on how the pins are being driven. The AMUTE is shown in [Figure 22-48](#) and described in [Table 22-21](#).

**Figure 22-48. Audio Mute Control Register (AMUTE)**

| 31       | Reserved |         |         |         |         |          |       | 16 |
|----------|----------|---------|---------|---------|---------|----------|-------|----|
| 15       | 13       | 12      | 11      | 10      | 9       | 8        |       |    |
|          | Reserved | XDMAERR | RDMAERR | XCKFAIL | RCKFAIL | XSYNCERR |       |    |
|          | R-0      | R/W-0   | R/W-0   | R/W-0   | R/W-0   | R/W-0    | R/W-0 |    |
| 7        | 6        | 5       | 4       | 3       | 2       | 1        | 0     |    |
| RSYNCERR | XUNDRN   | ROVRN   | INSTAT  | INEN    | INPOL   | MUTEN    |       |    |
| R/W-0    | R/W-0    | R/W-0   | R-0     | R/W-0   | R/W-0   | R/W-0    | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-21. Audio Mute Control Register (AMUTE) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-13 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 12    | XDMAERR  | 0     | If transmit DMA error (XDMAERR), drive AMUTE active enable bit.  |
|       |          | 1     | Drive is disabled. Detection of transmit DMA error is ignored by AMUTE.  |
| 11    | RDMAERR  | 0     | If receive DMA error (RDMAERR), drive AMUTE active enable bit.   |
|       |          | 1     | Drive is disabled. Detection of receive DMA error is ignored by AMUTE.   |
| 10    | XCKFAIL  | 0     | If transmit clock failure (XCKFAIL), drive AMUTE active enable bit.  |
|       |          | 1     | Drive is disabled. Detection of transmit clock failure is ignored by AMUTE.  |
| 9     | RCKFAIL  | 0     | If receive clock failure (RCKFAIL), drive AMUTE active enable bit.   |
|       |          | 1     | Drive is disabled. Detection of receive clock failure is ignored by AMUTE.   |
| 8     | XSYNCERR | 0     | If unexpected transmit frame sync error (XSYNCERR), drive AMUTE active enable bit.   |
|       |          | 1     | Drive is disabled. Detection of unexpected transmit frame sync error is ignored by AMUTE.  |
| 7     | RSYNCERR | 0     | If unexpected receive frame sync error (RSYNCERR), drive AMUTE active enable bit.  |
|       |          | 1     | Drive is disabled. Detection of unexpected receive frame sync error is ignored by AMUTE.   |
| 6     | XUNDRN   | 0     | If transmit underrun error (XUNDRN), drive AMUTE active enable bit.  |
|       |          | 1     | Drive is disabled. Detection of transmit underrun error is ignored by AMUTE.   |

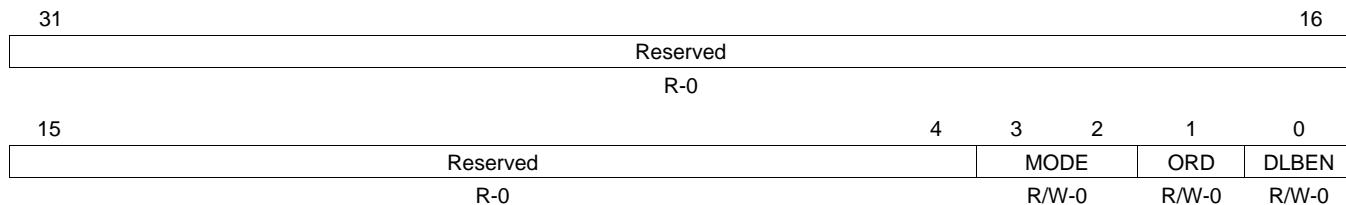
**Table 22-21. Audio Mute Control Register (AMUTE) Field Descriptions (continued)**

| Bit | Field  | Value | Description   |
|-----|--------|-------|---|
| 5   | ROVRN  | 0     | If receiver overrun error (ROVRN), drive AMUTE active enable bit.           |
|     |        | 1     | Drive is disabled. Detection of receiver overrun error is ignored by AMUTE. |
| 4   | INSTAT | 0     | Determines drive on AXRn pin when PFUNC[n] and PDIR[n] bits are set to 1.   |
|     |        | 1     | AMUTEIN pin is inactive.  |
| 3   | INEN   | 0     | Drive AMUTE active when AMUTEIN error is active (INSTAT = 1).               |
|     |        | 1     | Drive is disabled. AMUTEIN is ignored by AMUTE.                             |
| 2   | INPOL  | 0     | Audio mute in (AMUTEIN) polarity select bit.                                |
|     |        | 1     | Polarity is active high. A high on AMUTEIN sets INSTAT to 1.                |
| 1-0 | MUTEN  | 0-3h  | AMUTE pin enable bit (unless overridden by GPIO registers).                 |
|     |        | 0     | AMUTE pin is disabled, pin goes to tri-state condition.                     |
|     |        | 1h    | AMUTE pin is driven high if error is detected.                              |
|     |        | 2h    | AMUTE pin is driven low if error is detected.                               |
|     |        | 3h    | Reserved  |

### 22.4.1.11 Digital Loopback Control Register (DLBCTL)

The digital loopback control register (DLBCTL) controls the internal loopback settings of the McASP in TDM mode. The DLBCTL is shown in [Figure 22-49](#) and described in [Table 22-22](#).

**Figure 22-49. Digital Loopback Control Register (DLBCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-22. Digital Loopback Control Register (DLBCTL) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-4 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.         |
| 3-2  | MODE     | 0-3h  | Loopback generator mode bits. Applies only when loopback mode is enabled (DLBEN = 1).  |
|      |          | 0     | Default and reserved on loopback mode (DLBEN = 1).<br>When in non-loopback mode (DLBEN = 0), MODE should be left at default (00).<br>When in loopback mode (DLBEN = 1), MODE = 00 is reserved and is not applicable. |
|      |          | 1h    | Transmit clock and frame sync generators used by both transmit and receive sections. When in loopback mode (DLBEN = 1), MODE must be 01.   |
|      |          | 2h-3h | Reserved.  |
| 1    | ORD      | 0     | Loopback order bit when loopback mode is enabled (DLBEN = 1).<br>Odd serializers N + 1 transmit to even serializers N that receive. The corresponding serializers must be programmed properly.                       |
|      |          | 1     | Even serializers N transmit to odd serializers N + 1 that receive. The corresponding serializers must be programmed properly.  |
| 0    | DLBEN    | 0     | Loopback mode enable bit.<br>Loopback mode is disabled.  |
|      |          | 1     | Loopback mode is enabled.  |

### 22.4.1.12 Digital Mode Control Register (DITCTL)

The DIT mode control register (DITCTL) controls DIT operations of the McASP. The DITCTL is shown in Figure 22-50 and described in Table 22-23.

**Figure 22-50. Digital Mode Control Register (DITCTL)**

|     |          |                                |
|-----|----------|--------------------------------|
| 31  | Reserved | 16                             |
| R-0 |          |                                |
| 15  | Reserved | 4    3    2    1    0          |
|     | R-0      | R/W-0    R/W-0    R-0    R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-23. Digital Mode Control Register (DITCTL) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-4 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                        |
| 3    | VB       | 0     | Valid bit for odd time slots (DIT right subframe).  |
|      |          | 1     | V bit is 0 during odd DIT subframes.  |
|      |          | 1     | V bit is 1 during odd DIT subframes.  |
| 2    | VA       | 0     | Valid bit for even time slots (DIT left subframe).  |
|      |          | 1     | V bit is 0 during even DIT subframes.   |
|      |          | 1     | V bit is 1 during even DIT subframes.   |
| 1    | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                        |
| 0    | DITEN    | 0     | DIT mode enable bit. DITEN should only be changed while the XSMRST bit in GBLCTL is in reset (and for startup, XSRCLR also in reset). However, it is not necessary to reset the XCLKRST or XHCLKRST bits in GBLCTL to change DITEN. |
|      |          | 1     | DIT mode is disabled. Transmitter operates in TDM or burst mode.  |
|      |          | 1     | DIT mode is enabled. Transmitter operates in DIT encoded mode.  |

### 22.4.1.13 Receiver Global Control Register (RGBLCTL)

Alias of the global control register (GBLCTL). Writing to the receiver global control register (RGBLCTL) affects only the receive bits of GBLCTL (bits 4-0). Reads from RGBLCTL return the value of GBLCTL. RGBLCTL allows the receiver to be reset independently from the transmitter. The RGBLCTL is shown in Figure 22-51 and described in Table 22-24. See Section 22.4.1.9 for a detailed description of GBLCTL.

**Figure 22-51. Receiver Global Control Register (RGBLCTL)**

| 31       | Reserved |       |        |        |          |         |  | 16 |
|----------|----------|-------|--------|--------|----------|---------|--|----|
| R-0      |          |       |        |        |          |         |  |    |
| 15       | 13       | 12    | 11     | 10     | 9        | 8       |  |    |
| Reserved |          | XFRST | XSMRST | XSRCLR | XHCLKRST | XCLKRST |  |    |
| R-0      |          | R-0   | R-0    | R-0    | R-0      | R-0     |  |    |
| 7        | 5        | 4     | 3      | 2      | 1        | 0       |  |    |
| Reserved |          | RFRST | RSMRST | RSRCLR | RHCLKRST | RCLKRST |  |    |
| R-0      |          | R/W-0 | R/W-0  | R/W-0  | R/W-0    | R/W-0   |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-24. Receiver Global Control Register (RGBLCTL) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-13 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 12    | XFRST    | x     | Transmit frame sync generator reset enable bit. A read of this bit returns the XFRST bit value of GBLCTL. Writes have no effect.   |
| 11    | XSMRST   | x     | Transmit state machine reset enable bit. A read of this bit returns the XSMRST bit value of GBLCTL. Writes have no effect.   |
| 10    | XSRCLR   | x     | Transmit serializer clear enable bit. A read of this bit returns the XSRCLR bit value of GBLCTL. Writes have no effect.  |
| 9     | XHCLKRST | x     | Transmit high-frequency clock divider reset enable bit. A read of this bit returns the XHCLKRST bit value of GBLCTL. Writes have no effect.  |
| 8     | XCLKRST  | x     | Transmit clock divider reset enable bit. A read of this bit returns the XCLKRST bit value of GBLCTL. Writes have no effect.  |
| 7-5   | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 4     | RFRST    | 0     | Receive frame sync generator reset enable bit. A write to this bit affects the RFRST bit of GBLCTL.  |
|       |          | 1     | Receive frame sync generator is reset.   |
|       |          | 1     | Receive frame sync generator is active.  |
| 3     | RSMRST   | 0     | Receive state machine reset enable bit. A write to this bit affects the RSMRST bit of GBLCTL.  |
|       |          | 1     | Receive state machine is held in reset.  |
|       |          | 1     | Receive state machine is released from reset.  |
| 2     | RSRCLR   | 0     | Receive serializer clear enable bit. A write to this bit affects the RSRCLR bit of GBLCTL.   |
|       |          | 1     | Receive serializers are cleared.   |
|       |          | 1     | Receive serializers are active.  |
| 1     | RHCLKRST | 0     | Receive high-frequency clock divider reset enable bit. A write to this bit affects the RHCLKRST bit of GBLCTL.   |
|       |          | 1     | Receive high-frequency clock divider is held in reset and passes through its input as divide-by-1.   |
|       |          | 1     | Receive high-frequency clock divider is running.   |
| 0     | RCLKRST  | 0     | Receive clock divider reset enable bit. A write to this bit affects the RCLKRST bit of GBLCTL.   |
|       |          | 1     | Receive clock divider is held in reset.  |
|       |          | 1     | Receive clock divider is running.  |

#### 22.4.1.14 Receive Format Unit Bit Mask Register (RMASK)

The receive format unit bit mask register (RMASK) determines which bits of the received data are masked off and padded with a known value before being read by the CPU or DMA. The RMASK is shown in Figure 22-52 and described in Table 22-25.

**Figure 22-52. Receive Format Unit Bit Mask Register (RMASK)**

|    |          |   |
|----|----------|---|
| 31 | RMASK[n] | 0 |
|    | R/W-0    |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-25. Receive Format Unit Bit Mask Register (RMASK) Field Descriptions**

| Bit  | Field       | Value  | Description  |
|------|-------------|--------|--|
| 31-0 | RMASK[31-0] | 0<br>1 | Receive data mask <i>n</i> enable bit.<br>Corresponding bit of receive data (after passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (RPAD and RPBIT bits in RFMT).<br>Corresponding bit of receive data (after passing through reverse and rotate units) is returned to CPU or DMA. |

### 22.4.1.15 Receive Bit Stream Format Register (RFMT)

The receive bit stream format register (RFMT) configures the receive data format. The RFMT is shown in Figure 22-53 and described in Table 22-26.

**Figure 22-53. Receive Bit Stream Format Register (RFMT)**

|       |          |    |    |       |   |       |  |        |   |       |         |    |    |  |
|-------|----------|----|----|-------|---|-------|--|--------|---|-------|---------|----|----|--|
| 31    | Reserved |    |    |       |   |       |  |        |   |       | 18      | 17 | 16 |  |
|       | R-0      |    |    |       |   |       |  |        |   |       | RDATDLY |    |    |  |
| 15    | 14       | 13 | 12 |       | 8 | 7     |  | 4      | 3 | 2     | 0       |    |    |  |
| RRVRS | RPAD     |    |    | RPBIT |   | RSSZ  |  | RBUSEL |   | RROT  |         |    |    |  |
| R/W-0 | R/W-0    |    |    | R/W-0 |   | R/W-0 |  | R/W-0  |   | R/W-0 |         |    |    |  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-26. Receive Bit Stream Format Register (RFMT) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 31-18 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 17-16 | RDATDLY  | 0-3h   | Receive bit delay.   |
|       |          | 0      | 0-bit delay. The first receive data bit, AXRn, occurs in same ACLKR cycle as the receive frame sync (AFSR).  |
|       |          | 1h     | 1-bit delay. The first receive data bit, AXRn, occurs one ACLKR cycle after the receive frame sync (AFSR).   |
|       |          | 2h     | 2-bit delay. The first receive data bit, AXRn, occurs two ACLKR cycles after the receive frame sync (AFSR).  |
|       |          | 3h     | Reserved.  |
| 15    | RRVRS    | 0      | Receive serial bitstream order.  |
|       |          | 1      | Bitstream is LSB first. No bit reversal is performed in receive format bit reverse unit.   |
| 14-13 | RPAD     | 0-3h   | Pad value for extra bits in slot not belonging to the word. This field only applies to bits when RMASK[n] = 0.   |
|       |          | 0      | Pad extra bits with 0.   |
|       |          | 1h     | Pad extra bits with 1.   |
|       |          | 2h     | Pad extra bits with one of the bits from the word as specified by RPBIT bits.  |
|       |          | 3h     | Reserved.  |
| 12-8  | RPBIT    | 0-1Fh  | RPBIT value determines which bit (as read by the CPU or DMA from RBUF[n]) is used to pad the extra bits. This field only applies when RPAD = 2h.   |
|       |          | 0      | Pad with bit 0 value.  |
|       |          | 1h-1Fh | Pad with bit 1 to bit 31 value.  |

**Table 22-26. Receive Bit Stream Format Register (RFMT) Field Descriptions (continued)**

| Bit | Field  | Value | Description  |
|-----|--------|-------|--|
| 7-4 | RSSZ   | 0-Fh  | Receive slot size.   |
|     |        | 0-2h  | Reserved   |
|     |        | 3h    | Slot size is 8 bits.   |
|     |        | 4h    | Reserved   |
|     |        | 5h    | Slot size is 12 bits.  |
|     |        | 6h    | Reserved   |
|     |        | 7h    | Slot size is 16 bits.  |
|     |        | 8h    | Reserved   |
|     |        | 9h    | Slot size is 20 bits.  |
|     |        | Ah    | Reserved   |
|     |        | Bh    | Slot size is 24 bits   |
|     |        | Ch    | Reserved   |
|     |        | Dh    | Slot size is 28 bits.  |
|     |        | Eh    | Reserved   |
|     |        | Fh    | Slot size is 32 bits.  |
| 3   | RBUSEL | 0     | Selects whether reads from serializer buffer XRBUF[n] originate from the configuration bus (CFG) or the data (DAT) port. |
|     |        | 1     | Reads from XRBUF[n] originate on data port. Reads from XRBUF[n] on configuration bus are ignored.                        |
| 2-0 | RROT   | 0-7h  | Right-rotation value for receive rotate right format unit.   |
|     |        | 0     | Rotate right by 0 (no rotation).   |
|     |        | 1h    | Rotate right by 4 bit positions.   |
|     |        | 2h    | Rotate right by 8 bit positions.   |
|     |        | 3h    | Rotate right by 12 bit positions.  |
|     |        | 4h    | Rotate right by 16 bit positions.  |
|     |        | 5h    | Rotate right by 20 bit positions.  |
|     |        | 6h    | Rotate right by 24 bit positions.  |
|     |        | 7h    | Rotate right by 28 bit positions.  |

### 22.4.1.16 Receive Frame Sync Control Register (AFSRCTL)

The receive frame sync control register (AFSRCTL) configures the receive frame sync (AFSR). The AFSRCTL is shown in [Figure 22-54](#) and described in [Table 22-27](#).

**Figure 22-54. Receive Frame Sync Control Register (AFSRCTL)**

| 31    | Reserved |     |          |       |          |       |       |       | 16 |
|-------|----------|-----|----------|-------|----------|-------|-------|-------|----|
| R-0   |          |     |          |       |          |       |       |       |    |
| 15    | 7        | 6   | 5        | 4     | 3        | 2     | 1     | 0     |    |
|       | RMOD     |     | Reserved | FRWID | Reserved | FSRM  | FSRP  |       |    |
| R/W-0 |          | R-0 | R/W-0    | R-0   | R/W-0    | R/W-0 | R/W-0 | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-27. Receive Frame Sync Control Register (AFSRCTL) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--|--|
| 31-16 | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                     |
| 15-7  | RMOD     | 0-1FFh<br>0<br>1h<br>2h-20h<br>21h-17Fh<br>180h<br>181h-1FFh | Receive frame sync mode select bits.<br>Burst mode.<br>Reserved.<br>2-slot TDM (I2S mode) to 32-slot TDM.<br>Reserved.<br>384-slot TDM (external DIR IC inputting 384-slot DIR frames to McASP over I2S interface).<br>Reserved. |
| 6-5   | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                     |
| 4     | FRWID    | 0<br>1   | Receive frame sync width select bit indicates the width of the receive frame sync (AFSR) during its active period.<br>Single bit.<br>Single word.  |
| 3-2   | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                     |
| 1     | FSRM     | 0<br>1   | Receive frame sync generation select bit.<br>Externally-generated receive frame sync.<br>Internally-generated receive frame sync.  |
| 0     | FSRP     | 0<br>1   | Receive frame sync polarity select bit.<br>A rising edge on receive frame sync (AFSR) indicates the beginning of a frame.<br>A falling edge on receive frame sync (AFSR) indicates the beginning of a frame.                     |

### 22.4.1.17 Receive Clock Control Register (ACLRKCTL)

The receive clock control register (ACLRKCTL) configures the receive bit clock (ACLRK) and the receive clock generator. The ACLRKCTL is shown in [Figure 22-55](#) and described in [Table 22-28](#).

**Figure 22-55. Receive Clock Control Register (ACLRKCTL)**

|          |          |      |       |         |   |   |  |    |
|----------|----------|------|-------|---------|---|---|--|----|
| 31       | Reserved |      |       |         |   |   |  | 16 |
| R-0      |          |      |       |         |   |   |  |    |
| 15       | 8        | 7    | 6     | 5       | 4 | 0 |  |    |
| Reserved | CLKRP    | Rsvd | CLKRM | CLKRDIV |   |   |  |    |
| R-0      | R/W-0    | R-0  | R/W-1 | R/W-0   |   |   |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-28. Receive Clock Control Register (ACLRKCTL) Field Descriptions**

| Bit  | Field    | Value  | Description  |
|------|----------|--------|--|
| 31-8 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                               |
| 7    | CLKRP    | 0      | Receive bitstream clock polarity select bit.<br>Falling edge. Receiver samples data on the falling edge of the serial clock, so the external transmitter driving this receiver must shift data out on the rising edge of the serial clock. |
|      |          | 1      | Rising edge. Receiver samples data on the rising edge of the serial clock, so the external transmitter driving this receiver must shift data out on the falling edge of the serial clock.  |
| 6    | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                               |
| 5    | CLKRM    | 0      | Receive bit clock source bit. Note that this bit does not have any effect, if ACLKXCTL.ASYNC = 0.<br>External receive clock source from ACLKR pin.   |
|      |          | 1      | Internal receive clock source from output of programmable bit clock divider.   |
| 4-0  | CLKRDIV  | 0-1Fh  | Receive bit clock divide ratio bits determine the divide-down ratio from AHCLKR to ACLKR. Note that this bit does not have any effect, if ACLKXCTL.ASYNC = 0.  |
|      |          | 0      | Divide-by-1.   |
|      |          | 1h     | Divide-by-2.   |
|      |          | 2h-1Fh | Divide-by-3 to divide-by-32.   |

### 22.4.1.18 Receive High-Frequency Clock Control Register (AHCLKRCTL)

The receive high-frequency clock control register (AHCLKRCTL) configures the receive high-frequency master clock (AHCLKR) and the receive clock generator. The AHCLKRCTL is shown in [Figure 22-56](#) and described in [Table 22-29](#).

**Figure 22-56. Receive High-Frequency Clock Control Register (AHCLKRCTL)**

|        |          |          |    |    |          |    |
|--------|----------|----------|----|----|----------|----|
| 31     | Reserved |          |    |    |          | 16 |
|        | R-0      |          |    |    |          |    |
| 15     | 14       | 13       | 12 | 11 |          | 0  |
| HCLKRM | HCLKRP   | Reserved |    |    | HCLKRDIV |    |
| R/W-1  | R/W-0    | R/W-0    |    |    | R/W-0    |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

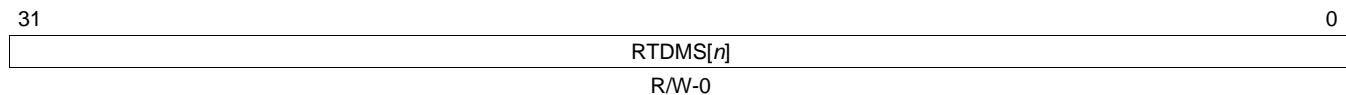
**Table 22-29. Receive High-Frequency Clock Control Register (AHCLKRCTL) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 31-16 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 15    | HCLKRM   | 0      | Receive high-frequency clock source bit.<br>External receive high-frequency clock source from AHCLKR pin.  |
|       |          | 1      | Internal receive high-frequency clock source from output of programmable high clock divider.   |
| 14    | HCLKRP   | 0      | Receive bitstream high-frequency clock polarity select bit.<br>AHCLKR is not inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin. |
|       |          | 1      | AHCLKR is inverted before programmable bit clock divider. In the special case where the receive bit clock (ACLKR) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKRDIV = 0 in ACLKRCTL), AHCLKR is directly passed through to the ACLKR pin.  |
| 13-12 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 11-0  | HCLKRDIV | 0-FFFh | Receive high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKR.<br>0: Divide-by-1.<br>1h: Divide-by-2.<br>2h-FFFh: Divide-by-3 to divide-by-4096.  |

### 22.4.1.19 Receive TDM Time Slot Register (RTDM)

The receive TDM time slot register (RTDM) specifies which TDM time slot the receiver is active. The RTDM is shown in [Figure 22-57](#) and described in [Table 22-30](#).

**Figure 22-57. Receive TDM Time Slot Register (RTDM)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-30. Receive TDM Time Slot Register (RTDM) Field Descriptions**

| Bit  | Field       | Value  | Description   |
|------|-------------|--------|---|
| 31-0 | RTDMS[31-0] | 0<br>1 | Receiver mode during TDM time slot n.<br>Receive TDM time slot n is inactive. The receive serializer does not shift in data during this slot.<br>Receive TDM time slot n is active. The receive serializer shifts in data during this slot. |

### 22.4.1.20 Receiver Interrupt Control Register (RINTCTL)

The receiver interrupt control register (RINTCTL) controls generation of the McASP receive interrupt (RINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates RINT. The RINTCTL is shown in [Figure 22-58](#) and described in [Table 22-31](#). See [Section 22.4.1.21](#) for a description of the interrupt conditions.

**Figure 22-58. Receiver Interrupt Control Register (RINTCTL)**

|                 |          |       |       |         |         |          |       |  |       |  |  |  |  |  |  |  |  |
|-----------------|----------|-------|-------|---------|---------|----------|-------|--|-------|--|--|--|--|--|--|--|--|
| 31              | Reserved |       |       |         |         |          |       |  | 8     |  |  |  |  |  |  |  |  |
| R-0             |          |       |       |         |         |          |       |  |       |  |  |  |  |  |  |  |  |
| 7 6 5 4 3 2 1 0 |          |       |       |         |         |          |       |  |       |  |  |  |  |  |  |  |  |
| RSTAFRM         | Reserved | RDATA | RLAST | RDMAERR | RCKFAIL | RSYNCERR | ROVRN |  |       |  |  |  |  |  |  |  |  |
| R/W-0           | R-0      | R/W-0 | R/W-0 | R/W-0   | R/W-0   | R/W-0    | R/W-0 |  | R/W-0 |  |  |  |  |  |  |  |  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-31. Receiver Interrupt Control Register (RINTCTL) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-8 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 7    | RSTAFRM  | 0     | Receive start of frame interrupt enable bit.<br>Interrupt is disabled. A receive start of frame interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receive start of frame interrupt generates a McASP receive interrupt (RINT).   |
| 6    | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5    | RDATA    | 0     | Receive data ready interrupt enable bit.<br>Interrupt is disabled. A receive data ready interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receive data ready interrupt generates a McASP receive interrupt (RINT).   |
| 4    | RLAST    | 0     | Receive last slot interrupt enable bit.<br>Interrupt is disabled. A receive last slot interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receive last slot interrupt generates a McASP receive interrupt (RINT).  |
| 3    | RDMAERR  | 0     | Receive DMA error interrupt enable bit.<br>Interrupt is disabled. A receive DMA error interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receive DMA error interrupt generates a McASP receive interrupt (RINT).  |
| 2    | RCKFAIL  | 0     | Receive clock failure interrupt enable bit.<br>Interrupt is disabled. A receive clock failure interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receive clock failure interrupt generates a McASP receive interrupt (RINT).  |
| 1    | RSYNCERR | 0     | Unexpected receive frame sync interrupt enable bit.<br>Interrupt is disabled. An unexpected receive frame sync interrupt does not generate a McASP receive interrupt (RINT).                                 |
|      |          | 1     | Interrupt is enabled. An unexpected receive frame sync interrupt generates a McASP receive interrupt (RINT).   |
| 0    | ROVRN    | 0     | Receiver overrun interrupt enable bit.<br>Interrupt is disabled. A receiver overrun interrupt does not generate a McASP receive interrupt (RINT).  |
|      |          | 1     | Interrupt is enabled. A receiver overrun interrupt generates a McASP receive interrupt (RINT).   |

### 22.4.1.21 Receiver Status Register (RSTAT)

The receiver status register (RSTAT) provides the receiver status and receive TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The RSTAT is shown in [Figure 22-59](#) and described in [Table 22-32](#).

**Figure 22-59. Receiver Status Register (RSTAT)**

| 31      | Reserved |         |         |          |         |          |         |       | 9 | 8 |
|---------|----------|---------|---------|----------|---------|----------|---------|-------|---|---|
| R-0     |          |         |         |          |         |          |         | R/W-0 |   |   |
| 7       | 6        | 5       | 4       | 3        | 2       | 1        | 0       |       |   |   |
| RDMAERR | RSTAfrm  | RDATA   | RLAST   | RTDMSLOT | RCKFAIL | RSYNCERR | ROVRN   |       |   |   |
| R/W1C-0 | R/W1C-0  | R/W1C-0 | R/W1C-0 | R-0      | R/W1C-0 | R/W1C-0  | R/W1C-0 |       |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1C = Writing a 1 clears this bit; writing a 0 has no effect.; -n = value after reset

**Table 22-32. Receiver Status Register (RSTAT) Field Descriptions**

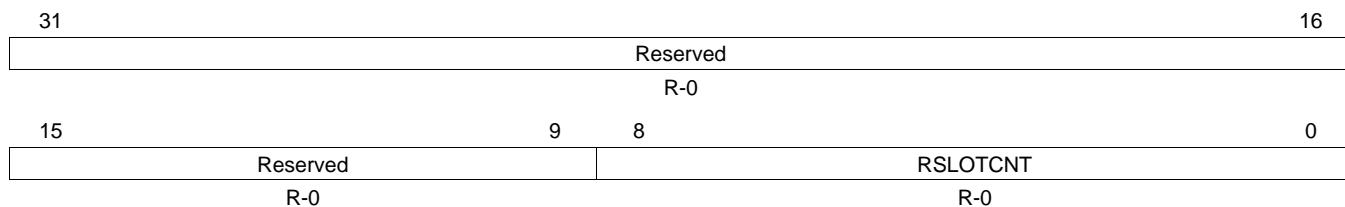
| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-9 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.  |
| 8    | RERR     | 0     | RERR bit always returns a logic-OR of:ROVRN   RSYNCERR   RCKFAIL   RDMAERR<br>Allows a single bit to be checked to determine if a receiver error interrupt has occurred.  |
|      |          | 1     | No errors have occurred.<br>An error has occurred.  |
| 7    | RDMAERR  | 0     | Receive DMA error flag. RDMAERR is set when the CPU or DMA reads more serializers through the data port in a given time slot than were programmed as receivers. Causes a receive interrupt (RINT), if this bit is set and RDMAERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. |
|      |          | 1     | Receive DMA error did not occur.<br>Receive DMA error did occur.  |
| 6    | RSTAfrm  | 0     | Receive start of frame flag. Causes a receive interrupt (RINT), if this bit is set and RSTAfrm in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.  |
|      |          | 1     | No new receive frame sync (AFSR) is detected.<br>A new receive frame sync (AFSR) is detected.   |
| 5    | RDATA    | 0     | Receive data ready flag. Causes a receive interrupt (RINT), if this bit is set and RDATA in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.  |
|      |          | 1     | No new data in RBUF.<br>Data is transferred from XRSR to RBUF and ready to be serviced by the CPU or DMA. When RDATA is set, it always causes a DMA event (AREVT).  |
| 4    | RLAST    | 0     | Receive last slot flag. RLAST is set along with RDATA, if the current slot is the last slot in a frame. Causes a receive interrupt (RINT), if this bit is set and RLAST in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.   |
|      |          | 1     | Current slot is not the last slot in a frame.<br>Current slot is the last slot in a frame. RDATA is also set.   |
| 3    | RTDMSLOT | 0     | Returns the LSB of RSLOT. Allows a single read of RSTAT to determine whether the current TDM time slot is even or odd.  |
|      |          | 1     | Current TDM time slot is odd.<br>Current TDM time slot is even.   |
| 2    | RCKFAIL  | 0     | Receive clock failure flag. RCKFAIL is set when the receive clock failure detection circuit reports an error (see <i>Clock Failure Detection</i> ). Causes a receive interrupt (RINT), if this bit is set and RCKFAIL in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.             |
|      |          | 1     | Receive clock failure did not occur.<br>Receive clock failure did occur.  |

**Table 22-32. Receiver Status Register (RSTAT) Field Descriptions (continued)**

| Bit | Field    | Value | Description  |
|-----|----------|-------|--|
| 1   | RSYNCERR | 0     | Unexpected receive frame sync flag. RSYNCERR is set when a new receive frame sync (AFSR) occurs before it is expected. Causes a receive interrupt (RINT), if this bit is set and RSYNCERR in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect.  |
|     |          |       | Unexpected receive frame sync did not occur.   |
|     |          | 1     | Unexpected receive frame sync did occur.   |
| 0   | ROVRN    | 0     | Receiver overrun flag. ROVRN is set when the receive serializer is instructed to transfer data from XRSR to RBUF, but the former data in RBUF has not yet been read by the CPU or DMA. Causes a receive interrupt (RINT), if this bit is set and ROVRN in RINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 to this bit has no effect. |
|     |          |       | Receiver overrun did not occur.  |
|     |          | 1     | Receiver overrun did occur.  |

#### 22.4.1.22 Current Receive TDM Time Slot Registers (RSLOT)

The current receive TDM time slot register (RSLOT) indicates the current time slot for the receive data frame. The RSLOT is shown in [Figure 22-60](#) and described in [Table 22-33](#).

**Figure 22-60. Current Receive TDM Time Slot Registers (RSLOT)**

LEGEND: R = Read only; -n = value after reset

**Table 22-33. Current Receive TDM Time Slot Registers (RSLOT) Field Descriptions**

| Bit  | Field    | Value  | Description  |
|------|----------|--------|--|
| 31-9 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                       |
| 8-0  | RSLOTCNT | 0-17Fh | Current receive time slot count. Legal values: 0 to 383 (17Fh). TDM function is not supported for > 32 time slots. However, TDM time slot counter may count to 383 when used to receive a DIR block (transferred over TDM format). |

### 22.4.1.23 Receive Clock Check Control Register (RCLKCHK)

The receive clock check control register (RCLKCHK) configures the receive clock failure detection circuit. The RCLKCHK is shown in [Figure 22-61](#) and described in [Table 22-34](#).

**Figure 22-61. Receive Clock Check Control Register (RCLKCHK)**

| 31    | 24 | 23       | 16    |
|-------|----|----------|-------|
| RCNT  |    | RMAX     |       |
| R-0   |    |          | R/W-0 |
| 15    | 8  | 7        | 4 3 0 |
| RMIN  |    | Reserved | RPS   |
| R/W-0 |    | R-0      | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-34. Receive Clock Check Control Register (RCLKCHK) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-24 | RCNT     | 0-FFh | Receive clock count value (from previous measurement). The clock circuit continually counts the number of system clocks for every 32 receive high-frequency master clock (AHCLKR) signals, and stores the count in RCNT until the next measurement is taken.   |
| 23-16 | RMAX     | 0-FFh | Receive clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If the current counter value is greater than RMAX after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.  |
| 15-8  | RMIN     | 0-FFh | Receive clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 receive high-frequency master clock (AHCLKR) signals have been received. If RCNT is less than RMIN after counting 32 AHCLKR signals, RCKFAIL in RSTAT is set. The comparison is performed using unsigned arithmetic.  |
| 7-4   | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 3-0   | RPS      | 0-Fh  | Receive clock check prescaler value.<br>0 McASP system clock divided by 1.<br>1h McASP system clock divided by 2.<br>2h McASP system clock divided by 4.<br>3h McASP system clock divided by 8.<br>4h McASP system clock divided by 16.<br>5h McASP system clock divided by 32.<br>6h McASP system clock divided by 64.<br>7h McASP system clock divided by 128.<br>8h McASP system clock divided by 256.<br>9h-Fh Reserved. |

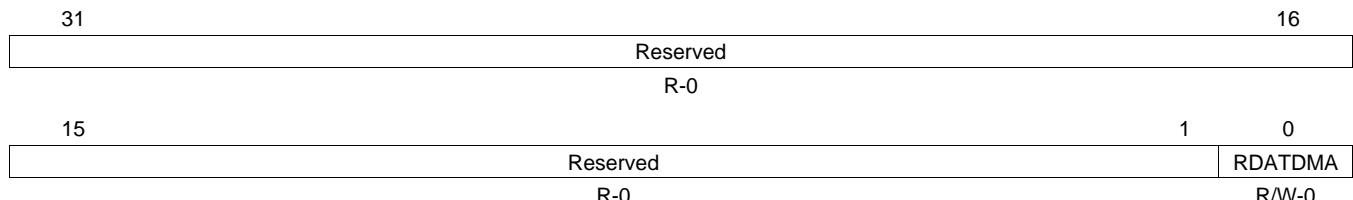
#### 22.4.1.24 Receiver DMA Event Control Register (REVCTL)

The receiver DMA event control register (REVCTL) contains a disable bit for the receiver DMA event. The REVCTL is shown in [Figure 22-62](#) and described in [Table 22-35](#).

**NOTE: Device-specific registers**

Accessing REVCTL not implemented on a specific device may cause improper device operation.

**Figure 22-62. Receiver DMA Event Control Register (REVCTL)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-35. Receiver DMA Event Control Register (REVCTL) Field Descriptions**

| Bit  | Field    | Value  | Description  |
|------|----------|--------|--|
| 31-1 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 0    | RDATDMA  | 0<br>1 | Receive data DMA request enable bit. If writing to this bit, always write the default value of 0.<br>Receive data DMA request is enabled.<br>Reserved  |

### 22.4.1.25 Transmitter Global Control Register (XGBLCTL)

Alias of the global control register (GBLCTL). Writing to the transmitter global control register (XGBLCTL) affects only the transmit bits of GBLCTL (bits 12-8). Reads from XGBLCTL return the value of GBLCTL. XGBLCTL allows the transmitter to be reset independently from the receiver. The XGBLCTL is shown in Figure 22-63 and described in Table 22-36. See Section 22.4.1.9 for a detailed description of GBLCTL.

**Figure 22-63. Transmitter Global Control Register (XGBLCTL)**

|     |          |       |        |        |          |         |  |    |
|-----|----------|-------|--------|--------|----------|---------|--|----|
| 31  | Reserved |       |        |        |          |         |  | 16 |
| R-0 |          |       |        |        |          |         |  |    |
| 15  | 13       | 12    | 11     | 10     | 9        | 8       |  |    |
|     | Reserved | XFRST | XSMRST | XSRCLR | XHCLKRST | XCLKRST |  |    |
| R-0 |          | R/W-0 | R/W-0  | R/W-0  | R/W-0    | R/W-0   |  |    |
| 7   | 5        | 4     | 3      | 2      | 1        | 0       |  |    |
|     | Reserved | RFRST | RSMRST | RSRCLR | RHCLKRST | RCLKRST |  |    |
| R-0 |          | R-0   | R-0    | R-0    | R-0      | R-0     |  |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-36. Transmitter Global Control Register (XGBLCTL) Field Descriptions**

| Bit   | Field    | Value  | Description   |
|-------|----------|--------|---|
| 31-13 | Reserved | 0-FFh  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                    |
| 12    | XFRST    | 0<br>1 | Transmit frame sync generator reset enable bit. A write to this bit affects the XFRST bit of GBLCTL.<br>Transmit frame sync generator is reset.<br>Transmit frame sync generator is active.                                     |
| 11    | XSMRST   | 0<br>1 | Transmit state machine reset enable bit. A write to this bit affects the XSMRST bit of GBLCTL.<br>Transmit state machine is held in reset.<br>Transmit state machine is released from reset.                                    |
| 10    | XSRCLR   | 0<br>1 | Transmit serializer clear enable bit. A write to this bit affects the XSRCLR bit of GBLCTL.<br>Transmit serializers are cleared.<br>Transmit serializers are active.  |
| 9     | XHCLKRST | 0<br>1 | Transmit high-frequency clock divider reset enable bit. A write to this bit affects the XHCLKRST bit of GBLCTL.<br>Transmit high-frequency clock divider is held in reset.<br>Transmit high-frequency clock divider is running. |
| 8     | XCLKRST  | 0<br>1 | Transmit clock divider reset enable bit. A write to this bit affects the XCLKRST bit of GBLCTL.<br>Transmit clock divider is held in reset.<br>Transmit clock divider is running.   |
| 7-5   | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.                    |
| 4     | RFRST    | x      | Receive frame sync generator reset enable bit. A read of this bit returns the RFRST bit value of GBLCTL. Writes have no effect.   |
| 3     | RSMRST   | x      | Receive state machine reset enable bit. A read of this bit returns the RSMRST bit value of GBLCTL. Writes have no effect.   |
| 2     | RSRCLR   | x      | Receive serializer clear enable bit. A read of this bit returns the RSRCLR bit value of GBLCTL. Writes have no effect.  |
| 1     | RHCLKRST | x      | Receive high-frequency clock divider reset enable bit. A read of this bit returns the RHCLKRST bit value of GBLCTL. Writes have no effect.  |
| 0     | RCLKRST  | x      | Receive clock divider reset enable bit. A read of this bit returns the RCLKRST bit value of GBLCTL. Writes have no effect.  |

### 22.4.1.26 Transmit Format Unit Bit Mask Register (XMASK)

The transmit format unit bit mask register (XMASK) determines which bits of the transmitted data are masked off and padded with a known value before being shifted out the McASP. The XMASK is shown in Figure 22-64 and described in Table 22-37.

**Figure 22-64. Transmit Format Unit Bit Mask Register (XMASK)**

|    |          |   |
|----|----------|---|
| 31 | XMASK[n] | 0 |
|    | R/W-0    |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-37. Transmit Format Unit Bit Mask Register (XMASK) Field Descriptions**

| Bit  | Field       | Value | Description  |
|------|-------------|-------|--|
| 31-0 | XMASK[31-0] | 0     | Transmit data mask <i>n</i> enable bit.<br><br>Corresponding bit of transmit data (before passing through reverse and rotate units) is masked out and then padded with the selected bit pad value (XPAD and XBIT bits in XFMT), which is transmitted out the McASP in place of the original bit. |
|      |             | 1     | Corresponding bit of transmit data (before passing through reverse and rotate units) is transmitted out the McASP.   |

### 22.4.1.27 Transmit Bit Stream Format Register (XFMT)

The transmit bit stream format register (XFMT) configures the transmit data format. The XFMT is shown in Figure 22-65 and described in Table 22-38.

**Figure 22-65. Transmit Bit Stream Format Register (XFMT)**

|       |          |    |       |   |       |   |        |       |       |    |    |  |
|-------|----------|----|-------|---|-------|---|--------|-------|-------|----|----|--|
| 31    | Reserved |    |       |   |       |   |        |       | 18    | 17 | 16 |  |
| R-0   |          |    |       |   |       |   |        | R/W-0 |       |    |    |  |
| 15    | 14       | 13 | 12    | 8 | 7     | 4 | 3      | 2     | 0     |    |    |  |
| XRVRS | XPAD     |    | XPBIT |   | XSSZ  |   | XBUSEL |       | XROT  |    |    |  |
| R/W-0 | R/W-0    |    | R/W-0 |   | R/W-0 |   | R/W-0  |       | R/W-0 |    |    |  |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-38. Transmit Bit Stream Format Register (XFMT) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-18 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 17-16 | XDATDLY  | 0-3h  | Transmit sync bit delay.   |
|       |          | 0     | 0-bit delay. The first transmit data bit, AXRn, occurs in same ACLKX cycle as the transmit frame sync (AFSX).  |
|       |          | 1h    | 1-bit delay. The first transmit data bit, AXRn, occurs one ACLKX cycle after the transmit frame sync (AFSX).   |
|       |          | 2h    | 2-bit delay. The first transmit data bit, AXRn, occurs two ACLKX cycles after the transmit frame sync (AFSX).  |
|       |          | 3h    | Reserved.  |
| 15    | XRVRS    | 0     | Transmit serial bitstream order.   |
|       |          | 1     | Bitstream is LSB first. No bit reversal is performed in transmit format bit reverse unit.<br>Bitstream is MSB first. Bit reversal is performed in transmit format bit reverse unit.                          |
| 14-13 | XPAD     | 0-3h  | Pad value for extra bits in slot not belonging to word defined by XMASK. This field only applies to bits when XMASK[n] = 0.  |
|       |          | 0     | Pad extra bits with 0.   |
|       |          | 1h    | Pad extra bits with 1.   |
|       |          | 2h    | Pad extra bits with one of the bits from the word as specified by XPBIT bits.  |
|       |          | 3h    | Reserved.  |
| 12-8  | XPBIT    | 0-1Fh | XPBIT value determines which bit (as written by the CPU or DMA to XBUF[n]) is used to pad the extra bits before shifting. This field only applies when XPAD = 2h.  |
|       |          | 0     | Pad with bit 0 value.  |
|       |          | 1-1Fh | Pad with bit 1 to bit 31 value.  |

**Table 22-38. Transmit Bit Stream Format Register (XFMT) Field Descriptions (continued)**

| Bit | Field  | Value | Description   |
|-----|--------|-------|---|
| 7-4 | XSSZ   | 0-Fh  | Transmit slot size.   |
|     |        | 0-2h  | Reserved.   |
|     |        | 3h    | Slot size is 8 bits.  |
|     |        | 4h    | Reserved.   |
|     |        | 5h    | Slot size is 12 bits.   |
|     |        | 6h    | Reserved.   |
|     |        | 7h    | Slot size is 16 bits.   |
|     |        | 8h    | Reserved.   |
|     |        | 9h    | Slot size is 20 bits.   |
|     |        | Ah    | Reserved.   |
|     |        | Bh    | Slot size is 24 bits.   |
|     |        | Ch    | Reserved.   |
|     |        | Dh    | Slot size is 28 bits.   |
|     |        | Eh    | Reserved.   |
|     |        | Fh    | Slot size is 32 bits.   |
| 3   | XBUSEL | 0     | Selects whether writes to serializer buffer XRBUF[n] originate from the configuration bus (CFG) or the data (DAT) port.                 |
|     |        | 1     | Writes to XRBUF[n] originate from the data port. Writes to XRBUF[n] from the configuration bus are ignored with no effect to the McASP. |
| 2-0 | XROT   | 0-7h  | Right-rotation value for transmit rotate right format unit.   |
|     |        | 0     | Rotate right by 0 (no rotation).  |
|     |        | 1h    | Rotate right by 4 bit positions.  |
|     |        | 2h    | Rotate right by 8 bit positions.  |
|     |        | 3h    | Rotate right by 12 bit positions.   |
|     |        | 4h    | Rotate right by 16 bit positions.   |
|     |        | 5h    | Rotate right by 20 bit positions.   |
|     |        | 6h    | Rotate right by 24 bit positions.   |
|     |        | 7h    | Rotate right by 28 bit positions.   |

### 22.4.1.28 Transmit Frame Sync Control Register (AFSXCTL)

The transmit frame sync control register (AFSXCTL) configures the transmit frame sync (AFSX). The AFSXCTL is shown in [Figure 22-66](#) and described in [Table 22-39](#).

**Figure 22-66. Transmit Frame Sync Control Register (AFSXCTL)**

| 31 | Reserved |   |     |   |       |   |     |       | 16    |
|----|----------|---|-----|---|-------|---|-----|-------|-------|
|    |          |   |     |   |       |   |     |       | R-0   |
| 15 | XMOD     | 7 | 6   | 5 | 4     | 3 | 2   | 1     | 0     |
|    | R/W-0    |   | R-0 |   | R/W-0 |   | R-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-39. Transmit Frame Sync Control Register (AFSXCTL) Field Descriptions**

| Bit   | Field    | Value  | Description   |
|-------|----------|--|---|
| 31-16 | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.    |
| 15-7  | XMOD     | 0-1FFh<br>0<br>1h<br>2h-20h<br>21h-17Fh<br>180h<br>181h-1FFh | Transmit frame sync mode select bits.<br>Burst mode.<br>Reserved.<br>2-slot TDM (I2S mode) to 32-slot TDM.<br>Reserved.<br>384-slot DIT mode.<br>Reserved.  |
| 6-5   | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.    |
| 4     | FXWID    | 0<br>1   | Transmit frame sync width select bit indicates the width of the transmit frame sync (AFSX) during its active period.<br>Single bit.<br>Single word.   |
| 3-2   | Reserved | 0  | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.    |
| 1     | FSXM     | 0<br>1   | Transmit frame sync generation select bit.<br>Externally-generated transmit frame sync.<br>Internally-generated transmit frame sync.  |
| 0     | FSXP     | 0<br>1   | Transmit frame sync polarity select bit.<br>A rising edge on transmit frame sync (AFSX) indicates the beginning of a frame.<br>A falling edge on transmit frame sync (AFSX) indicates the beginning of a frame. |

### 22.4.1.29 Transmit Clock Control Register (ACLKXCTL)

The transmit clock control register (ACLKXCTL) configures the transmit bit clock (ACLKX) and the transmit clock generator. The ACLKXCTL is shown in [Figure 22-67](#) and described in [Table 22-40](#).

**Figure 22-67. Transmit Clock Control Register (ACLKXCTL)**

|                         |          |  |   |   |   |   |   |        |    |
|-------------------------|----------|--|---|---|---|---|---|--------|----|
| 31                      | Reserved |  |   |   |   |   |   |        | 16 |
| R-0                     |          |  |   |   |   |   |   |        |    |
| 15                      | Reserved |  |   |   |   |   |   |        | 0  |
| R-0                     |          |  | 8 | 7 | 6 | 5 | 4 | CLKDIV |    |
| R/W-0 R/W-1 R/W-1 R/W-0 |          |  |   |   |   |   |   |        |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-40. Transmit Clock Control Register (ACLKXCTL) Field Descriptions**

| Bit  | Field    | Value  | Description  |
|------|----------|--------|--|
| 31-8 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.         |
| 7    | CLKXP    | 0      | Transmit bitstream clock polarity select bit.<br>Rising edge. External receiver samples data on the falling edge of the serial clock, so the transmitter must shift data out on the rising edge of the serial clock. |
|      |          | 1      | Falling edge. External receiver samples data on the rising edge of the serial clock, so the transmitter must shift data out on the falling edge of the serial clock.   |
| 6    | ASYNC    | 0      | Transmit/receive operation asynchronous enable bit.<br>Synchronous. Transmit clock and frame sync provides the source for both the transmit and receive sections.  |
|      |          | 1      | Asynchronous. Separate clock and frame sync used by transmit and receive sections.   |
| 5    | CLKXM    | 0      | Transmit bit clock source bit.<br>External transmit clock source from ACLKX pin.   |
|      |          | 1      | Internal transmit clock source from output of programmable bit clock divider.  |
| 4-0  | CLKDIV   | 0-1Fh  | Transmit bit clock divide ratio bits determine the divide-down ratio from AHCLKX to ACLKX.   |
|      |          | 0      | Divide-by-1.   |
|      |          | 1h     | Divide-by-2.   |
|      |          | 2h-1Fh | Divide-by-3 to divide-by-32.   |

### 22.4.1.30 Transmit High-Frequency Clock Control Register (AHCLKXCTL)

The transmit high-frequency clock control register (AHCLKXCTL) configures the transmit high-frequency master clock (AHCLKX) and the transmit clock generator. The AHCLKXCTL is shown in [Figure 22-68](#) and described in [Table 22-41](#).

**Figure 22-68. Transmit High-Frequency Clock Control Register (AHCLKXCTL)**

|                                    |          |          |  |  |          |    |
|------------------------------------|----------|----------|--|--|----------|----|
| 31                                 |          |          |  |  |          | 16 |
|                                    | Reserved |          |  |  |          |    |
|                                    | R-0      |          |  |  |          |    |
| 15      14      13      12      11 |          |          |  |  |          | 0  |
| HCLKXM                             | HCLKXP   | Reserved |  |  | HCLKXDIV |    |
| R/W-1                              | R/W-0    | R/W-0    |  |  | R/W-0    |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-41. Transmit High-Frequency Clock Control Register (AHCLKXCTL) Field Descriptions**

| Bit   | Field    | Value                        | Description  |
|-------|----------|------------------------------|--|
| 31-16 | Reserved | 0                            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 15    | HCLKXM   | 0<br>1                       | Transmit high-frequency clock source bit.<br>External transmit high-frequency clock source from AHCLKX pin.<br>Internal transmit high-frequency clock source from output of programmable high clock divider.   |
| 14    | HCLKXP   | 0<br>1                       | Transmit bitstream high-frequency clock polarity select bit.<br>AHCLKX is not inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin.<br>AHCLKX is inverted before programmable bit clock divider. In the special case where the transmit bit clock (ACLKX) is internally generated and the programmable bit clock divider is set to divide-by-1 (CLKXDIV = 0 in ACLKXCTL), AHCLKX is directly passed through to the ACLKX pin. |
| 13-12 | Reserved | 0                            | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 11-0  | HCLKXDIV | 0-FFFh<br>0<br>1h<br>2h-FFFh | Transmit high-frequency clock divide ratio bits determine the divide-down ratio from AUXCLK to AHCLKX.<br>Divide-by-1.<br>Divide-by-2.<br>Divide-by-3 to divide-by-4096.   |

### 22.4.1.31 Transmit TDM Time Slot Register (XTDM)

The transmit TDM time slot register (XTDM) specifies in which TDM time slot the transmitter is active. TDM time slot counter range is extended to 384 slots (to support SPDIF blocks of 384 subframes). XTDM operates modulo 32, that is, XTDMS specifies the TDM activity for time slots 0, 32, 64, 96, 128, etc. The XTDM is shown in Figure 22-69 and described in Table 22-42.

**Figure 22-69. Transmit TDM Time Slot Register (XTDM)**

|    |          |   |
|----|----------|---|
| 31 | XTDMS[n] | 0 |
|    | R/W-0    |   |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 22-42. Transmit TDM Time Slot Register (XTDM) Field Descriptions**

| Bit  | Field       | Value  | Description   |
|------|-------------|--------|---|
| 31-0 | XTDMS[31-0] | 0<br>1 | Transmitter mode during TDM time slot n.<br>Transmit TDM time slot n is inactive. The transmit serializer does not shift out data during this slot.<br>Transmit TDM time slot n is active. The transmit serializer shifts out data during this slot according to the serializer control register (SRCTL). |

### 22.4.1.32 Transmitter Interrupt Control Register (XINTCTL)

The transmitter interrupt control register (XINTCTL) controls generation of the McASP transmit interrupt (XINT). When the register bit(s) is set to 1, the occurrence of the enabled McASP condition(s) generates XINT. The XINTCTL is shown in [Figure 22-70](#) and described in [Table 22-43](#). See [Section 22.4.1.33](#) for a description of the interrupt conditions.

**Figure 22-70. Transmitter Interrupt Control Register (XINTCTL)**

|         |          |       |       |         |         |          |        |  |   |
|---------|----------|-------|-------|---------|---------|----------|--------|--|---|
| 31      | Reserved |       |       |         |         |          |        |  | 8 |
| R-0     |          |       |       |         |         |          |        |  |   |
| 7       | 6        | 5     | 4     | 3       | 2       | 1        | 0      |  |   |
| XSTAFRM | Reserved | XDATA | XLAST | XDMAERR | XCKFAIL | XSYNCERR | XUNDRN |  |   |

R/W-0      R-0      R/W-0      R/W-0      R/W-0      R/W-0      R/W-0      R/W-0      R/W-0

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-43. Transmitter Interrupt Control Register (XINTCTL) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-8 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 7    | XSTAFRM  | 0     | Transmit start of frame interrupt enable bit.<br>Interrupt is disabled. A transmit start of frame interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmit start of frame interrupt generates a McASP transmit interrupt (XINT).   |
| 6    | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 5    | XDATA    | 0     | Transmit data ready interrupt enable bit.<br>Interrupt is disabled. A transmit data ready interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmit data ready interrupt generates a McASP transmit interrupt (XINT).   |
| 4    | XLAST    | 0     | Transmit last slot interrupt enable bit.<br>Interrupt is disabled. A transmit last slot interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmit last slot interrupt generates a McASP transmit interrupt (XINT).  |
| 3    | XDMAERR  | 0     | Transmit DMA error interrupt enable bit.<br>Interrupt is disabled. A transmit DMA error interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmit DMA error interrupt generates a McASP transmit interrupt (XINT).  |
| 2    | XCKFAIL  | 0     | Transmit clock failure interrupt enable bit.<br>Interrupt is disabled. A transmit clock failure interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmit clock failure interrupt generates a McASP transmit interrupt (XINT).  |
| 1    | XSYNCERR | 0     | Unexpected transmit frame sync interrupt enable bit.<br>Interrupt is disabled. An unexpected transmit frame sync interrupt does not generate a McASP transmit interrupt (XINT).                              |
|      |          | 1     | Interrupt is enabled. An unexpected transmit frame sync interrupt generates a McASP transmit interrupt (XINT).   |
| 0    | XUNDRN   | 0     | Transmitter underrun interrupt enable bit.<br>Interrupt is disabled. A transmitter underrun interrupt does not generate a McASP transmit interrupt (XINT).   |
|      |          | 1     | Interrupt is enabled. A transmitter underrun interrupt generates a McASP transmit interrupt (XINT).  |

### 22.4.1.33 Transmitter Status Register (XSTAT)

The transmitter status register (XSTAT) provides the transmitter status and transmit TDM time slot number. If the McASP logic attempts to set an interrupt flag in the same cycle that the CPU writes to the flag to clear it, the McASP logic has priority and the flag remains set. This also causes a new interrupt request to be generated. The XSTAT is shown in [Figure 22-71](#) and described in [Table 22-44](#).

**Figure 22-71. Transmitter Status Register (XSTAT)**

| 31      | Reserved |         |         |     |         |         |         |         | 9       | 8       |         |          |         |          |        |
|---------|----------|---------|---------|-----|---------|---------|---------|---------|---------|---------|---------|----------|---------|----------|--------|
| R-0     |          |         |         |     |         |         |         | R/W-0   |         |         |         |          |         |          |        |
| 7       | 6        | 5       | 4       | 3   | 2       | 1       | 0       | XDMAERR | XSTAFRM | XDATA   | XLAST   | XTDMSLOT | XCKFAIL | XSYNCERR | XUNDRN |
| R/W1C-0 | R/W1C-0  | R/W1C-0 | R/W1C-0 | R-0 | R/W1C-0  | R/W1C-0 | R/W1C-0  |        |

LEGEND: R/W = Read/Write; R = Read only; W1C = Writing a 1 clears this bit; writing a 0 has no effect; -n = value after reset

**Table 22-44. Transmitter Status Register (XSTAT) Field Descriptions**

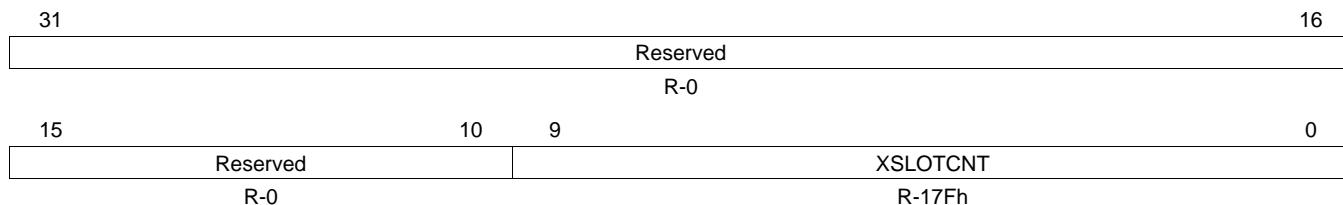
| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-9 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.  |
| 8    | XERR     | 0     | XERR bit always returns a logic-OR of:XUNDRN   XSYNCERR   XCKFAIL   XDMAERR Allows a single bit to be checked to determine if a transmitter error interrupt has occurred.   |
|      |          | 1     | No errors have occurred.<br>An error has occurred.  |
| 7    | XDMAERR  | 0     | Transmit DMA error flag. XDMAERR is set when the CPU or DMA writes more serializers through the data port in a given time slot than were programmed as transmitters. Causes a transmit interrupt (XINT), if this bit is set and XDMAERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. |
|      |          | 1     | Transmit DMA error did not occur.<br>Transmit DMA error did occur.  |
| 6    | XSTAFRM  | 0     | Transmit start of frame flag. Causes a transmit interrupt (XINT), if this bit is set and XSTAFRM in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.  |
|      |          | 1     | No new transmit frame sync (AFSX) is detected.<br>A new transmit frame sync (AFSX) is detected.   |
| 5    | XDATA    | 0     | Transmit data ready flag. Causes a transmit interrupt (XINT), if this bit is set and XDATA in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.  |
|      |          | 1     | XBUF is written and is full.<br>Data is copied from XBUF to XRSR. XBUF is empty and ready to be written. XDATA is also set when the transmit serializers are taken out of reset. When XDATA is set, it always causes a DMA event (AXEVT).   |
| 4    | XLAST    | 0     | Transmit last slot flag. XLAST is set along with XDATA, if the current slot is the last slot in a frame. Causes a transmit interrupt (XINT), if this bit is set and XLAST in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.   |
|      |          | 1     | Current slot is not the last slot in a frame.<br>Current slot is the last slot in a frame. XDATA is also set.   |
| 3    | XTDMSLOT | 0     | Returns the LSB of XSLOT. Allows a single read of XSTAT to determine whether the current TDM time slot is even or odd.  |
|      |          | 1     | Current TDM time slot is odd.<br>Current TDM time slot is even.   |
| 2    | XCKFAIL  | 0     | Transmit clock failure flag. XCKFAIL is set when the transmit clock failure detection circuit reports an error (see <i>Clock Failure Detection</i> ). Causes a transmit interrupt (XINT), if this bit is set and XCKFAIL in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.                |
|      |          | 1     | Transmit clock failure did not occur.<br>Transmit clock failure did occur.  |

**Table 22-44. Transmitter Status Register (XSTAT) Field Descriptions (continued)**

| Bit | Field    | Value | Description   |
|-----|----------|-------|---|
| 1   | XSYNCERR | 0     | Unexpected transmit frame sync flag. XSYNCERR is set when a new transmit frame sync (AFSX) occurs before it is expected. Causes a transmit interrupt (XINT), if this bit is set and XSYNCERR in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect.  |
|     |          |       | Unexpected transmit frame sync did not occur.   |
|     |          | 1     | Unexpected transmit frame sync did occur.   |
| 0   | XUNDRN   | 0     | Transmitter underrun flag. XUNDRN is set when the transmit serializer is instructed to transfer data from XBUF to XRSR, but XBUF has not yet been serviced with new data since the last transfer. Causes a transmit interrupt (XINT), if this bit is set and XUNDRN in XINTCTL is set. This bit is cleared by writing a 1 to this bit. Writing a 0 has no effect. |
|     |          |       | Transmitter underrun did not occur.   |
|     |          | 1     | Transmitter underrun did occur. For details on McASP action upon underrun conditions, see <i>Buffer Underrun Error - Transmitter</i> .  |

#### 22.4.1.34 Current Transmit TDM Time Slot Register (XSLOT)

The current transmit TDM time slot register (XSLOT) indicates the current time slot for the transmit data frame. The XSLOT is shown in [Figure 22-72](#) and described in [Table 22-45](#).

**Figure 22-72. Current Transmit TDM Time Slot Register (XSLOT)**

LEGEND: R = Read only; -n = value after reset

**Table 22-45. Current Transmit TDM Time Slot Register (XSLOT) Field Descriptions**

| Bit   | Field    | Value  | Description   |
|-------|----------|--------|---|
| 31-10 | Reserved | 0      | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.  |
| 9-0   | XSLOTCNT | 0-17Fh | Current transmit time slot count. Legal values: 0 to 383 (17Fh). During reset, this counter value is 383 so the next count value, which is used to encode the first DIT group of data, will be 0 and encodes the B preamble. TDM function is not supported for >32 time slots. However, TDM time slot counter may count to 383 when used to transmit a DIT block. |

### 22.4.1.35 Transmit Clock Check Control Register (XCLKCHK)

The transmit clock check control register (XCLKCHK) configures the transmit clock failure detection circuit. The XCLKCHK is shown in [Figure 22-73](#) and described in [Table 22-46](#).

**Figure 22-73. Transmit Clock Check Control Register (XCLKCHK)**

| 31    | 24       | 23   | 16    |
|-------|----------|------|-------|
| XCNT  |          | XMAX |       |
| R-0   |          |      | R/W-0 |
| 15    | 8        | 7    | 4 3 0 |
| XMIN  | Reserved |      | XPS   |
| R/W-0 | R-0      |      | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-46. Transmit Clock Check Control Register (XCLKCHK) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-24 | XCNT     | 0     | Transmit clock count value (from previous measurement). The clock circuit continually counts the number of system clocks for every 32 transmit high-frequency master clock (AHCLKX) signals, and stores the count in XCNT until the next measurement is taken.  |
| 23-16 | XMAX     | 0-FFh | Transmit clock maximum boundary. This 8-bit unsigned value sets the maximum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If the current counter value is greater than XMAX after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.   |
| 15-8  | XMIN     | 0-FFh | Transmit clock minimum boundary. This 8-bit unsigned value sets the minimum allowed boundary for the clock check counter after 32 transmit high-frequency master clock (AHCLKX) signals have been received. If XCNT is less than XMIN after counting 32 AHCLKX signals, XCKFAIL in XSTAT is set. The comparison is performed using unsigned arithmetic.   |
| 7-4   | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.  |
| 3-0   | XPS      | 0-Fh  | Transmit clock check prescaler value.<br>0 McASP system clock divided by 1.<br>1h McASP system clock divided by 2.<br>2h McASP system clock divided by 4.<br>3h McASP system clock divided by 8.<br>4h McASP system clock divided by 16.<br>5h McASP system clock divided by 32.<br>6h McASP system clock divided by 64.<br>7h McASP system clock divided by 128.<br>8h McASP system clock divided by 256.<br>9h-Fh Reserved. |

#### 22.4.1.36 Transmitter DMA Event Control Register (XEVCTL)

The transmitter DMA event control register (XEVTCTL) contains a disable bit for the transmit DMA event. The XEVTCCTL is shown in Figure 22-74 and described in Table 22-47.

**NOTE:** Device-specific registers

Accessing REVCTL not implemented on a specific device may cause improper device operation.

**Figure 22-74. Transmitter DMA Event Control Register (XEVCTL)**

|          |  |   |         |
|----------|--|---|---------|
| 31       |  |   | 16      |
| Reserved |  |   |         |
| R-0      |  |   |         |
| 15       |  | 1 | 0       |
| Reserved |  |   | XDATDMA |
| R-0      |  |   | R/W-0   |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-47. Transmitter DMA Event Control Register (XEVCTL) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-1 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility. |
| 0    | XDATDMA  | 0     | Transmit data DMA request enable bit. If writing to this bit, always write the default value of 0.   |
|      |          | 1     | Transmit data DMA request is enabled.  |
|      |          |       | Reserved   |

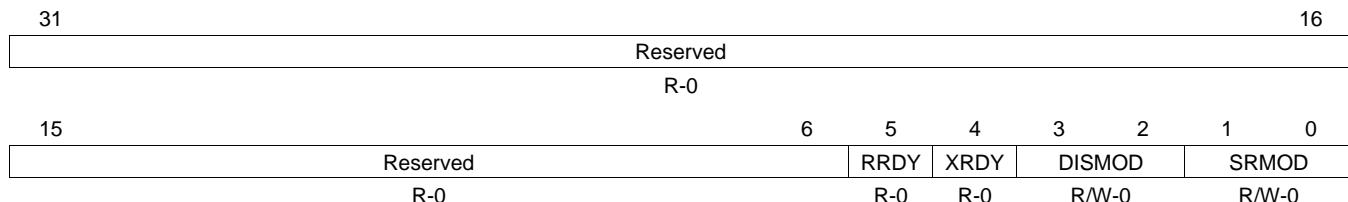
### 22.4.1.37 Serializer Control Registers (SRCTL $n$ )

Each serializer on the McASP has a serializer control register (SRCTL). There are up to 6 serializers per McASP. The SRCTL is shown in [Figure 22-75](#) and described in [Table 22-48](#).

**NOTE: Device-specific registers**

Accessing SRCTL $n$  not implemented on a specific device may cause improper device operation.

**Figure 22-75. Serializer Control Registers (SRCTL $n$ )**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 22-48. Serializer Control Registers (SRCTL $n$ ) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-6 | Reserved | 0     | Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.   |
| 5    | RRDY     | 0     | Receive buffer ready bit. RRDY indicates the current receive buffer state. Always reads 0 when programmed as a transmitter or as inactive. If SRMOD bit is set to receive (2h), RRDY switches from 0 to 1 whenever data is transferred from XRSR to RBUF.  |
|      |          | 1     | Receive buffer (RBUF) is empty.  |
|      |          | 1     | Receive buffer (RBUF) contains data and needs to be read before the start of the next time slot or a receiver overrun occurs.  |
| 4    | XRDY     | 0     | Transmit buffer ready bit. XRDY indicates the current transmit buffer state. Always reads 0 when programmed as a receiver or as inactive. If SRMOD bit is set to transmit (1h), XRDY switches from 0 to 1 when XSRCLR in GBLCTL is switched from 0 to 1 to indicate an empty transmitter. XRDY remains set until XSRCLR is forced to 0, data is written to the corresponding transmit buffer, or SRMOD bit is changed to receive (2h) or inactive (0). |
|      |          | 0     | Transmit buffer (XBUF) contains data.  |
|      |          | 1     | Transmit buffer (XBUF) is empty and needs to be written before the start of the next time slot or a transmit underrun occurs.  |
| 3-2  | DISMOD   | 0-3h  | Serializer pin drive mode bit. Drive on pin when in inactive TDM slot of transmit mode or when serializer is inactive. This field only applies if the pin is configured as a McASP pin (PFUNC = 0).  |
|      |          | 0     | Drive on pin is 3-state.   |
|      |          | 1h    | Reserved.  |
|      |          | 2h    | Drive on pin is logic low.   |
|      |          | 3h    | Drive on pin is logic high.  |
| 1-0  | SRMOD    | 0-3h  | Serializer mode bit.   |
|      |          | 0     | Serializer is inactive.  |
|      |          | 1h    | Serializer is transmitter.   |
|      |          | 2h    | Serializer is receiver.  |
|      |          | 3h    | Reserved.  |

#### 22.4.1.38 DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)

The DIT left channel status registers (DITCSRA) provide the status of each left channel (even TDM time slot). Each of the six 32-bit registers ([Figure 22-76](#)) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

**Figure 22-76. DIT Left Channel Status Registers (DITCSRA0-DITCSRA5)**

|    |                     |   |
|----|---------------------|---|
| 31 | DITCSRA[ <i>n</i> ] | 0 |
|    | R/W-0               |   |

LEGEND: R/W = Read/Write; -*n* = value after reset

#### 22.4.1.39 DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)

The DIT right channel status registers (DITCSRB) provide the status of each right channel (odd TDM time slot). Each of the six 32-bit registers ([Figure 22-77](#)) can store 192 bits of channel status data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register file in time, if a different set of data need to be sent.

**Figure 22-77. DIT Right Channel Status Registers (DITCSRB0-DITCSRB5)**

|    |                     |   |
|----|---------------------|---|
| 31 | DITCSRB[ <i>n</i> ] | 0 |
|    | R/W-0               |   |

LEGEND: R/W = Read/Write; -*n* = value after reset

#### 22.4.1.40 DIT Left Channel User Data Registers (DITUUDRA0-DITUUDRA5)

The DIT left channel user data registers (DITUUDRA) provides the user data of each left channel (even TDM time slot). Each of the six 32-bit registers ([Figure 22-78](#)) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

**Figure 22-78. DIT Left Channel User Data Registers (DITUUDRA0-DITUUDRA5)**

|    |                      |   |
|----|----------------------|---|
| 31 | DITUUDRA[ <i>n</i> ] | 0 |
|    | R/W-0                |   |

LEGEND: R/W = Read/Write; -*n* = value after reset

#### 22.4.1.41 DIT Right Channel User Data Registers (DITUUDRB0-DITUUDRB5)

The DIT right channel user data registers (DITUUDRB) provides the user data of each right channel (odd TDM time slot). Each of the six 32-bit registers ([Figure 22-79](#)) can store 192 bits of user data for a complete block of transmission. The DIT reuses the same data for the next block. It is your responsibility to update the register in time, if a different set of data need to be sent.

**Figure 22-79. DIT Right Channel User Data Registers (DITUUDRB0-DITUUDRB5)**

|    |                      |   |
|----|----------------------|---|
| 31 | DITUUDRB[ <i>n</i> ] | 0 |
|    | R/W-0                |   |

LEGEND: R/W = Read/Write; -*n* = value after reset

#### 22.4.1.42 Transmit Buffer Registers (XBUF $n$ )

The transmit buffers for the serializers (XBUF) hold data from the transmit format unit. For transmit operations, the XBUF ([Figure 22-80](#)) is an alias of the XRBUF in the serializer.

---

**NOTE: Device-specific registers**

Accessing XBUF registers not implemented on a specific device may cause improper device operation.

---

**Figure 22-80. Transmit Buffer Registers (XBUF $n$ )**

|    |             |   |
|----|-------------|---|
| 31 | XBUF[ $n$ ] | 0 |
|    | R/W-0       |   |

LEGEND: R/W = Read/Write; - $n$  = value after reset

#### 22.4.1.43 Receive Buffer Registers (RBUF $n$ )

The receive buffers for the serializers (RBUF) hold data from the serializer before the data goes to the receive format unit. For receive operations, the RBUF ([Figure 22-81](#)) is an alias of the XRBUF in the serializer.

---

**NOTE: Device-specific registers**

Accessing XBUF registers not implemented on a specific device may cause improper device operation.

---

**Figure 22-81. Receive Buffer Registers (RBUF $n$ )**

|    |             |   |
|----|-------------|---|
| 31 | RBUF[ $n$ ] | 0 |
|    | R/W-0       |   |

LEGEND: R/W = Read/Write; - $n$  = value after reset

#### 22.4.1.44 Write FIFO Control Register (WFIFOCTL)

The Write FIFO control register (WFIFOCTL) is shown in [Figure 22-82](#) and described in [Table 22-49](#).

**NOTE:** The WNUMEVT and WNUMDMA values must be set prior to enabling the Write FIFO.

If the Write FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

**Figure 22-82. Write FIFO Control Register (WFIFOCTL)**

|    |          |   |         |       |
|----|----------|---|---------|-------|
| 31 | Reserved |   | 17      | 16    |
|    |          |   | R-0     | R/W-0 |
| 15 | 8        | 7 | 0       |       |
|    | WNUMEVT  |   | WNUMDMA |       |
|    | R/W-10h  |   | R/W-4h  |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

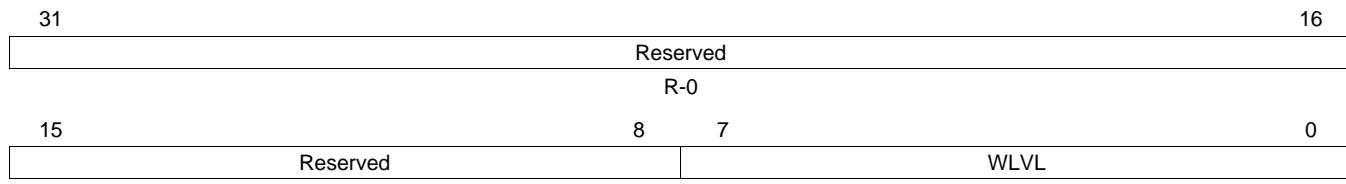
**Table 22-49. Write FIFO Control Register (WFIFOCTL) Field Descriptions**

| Bit   | Field    | Value                                       | Description  |
|-------|----------|---|--|
| 31-17 | Reserved | 0   | Reserved   |
| 16    | WENA     | 0<br>1                                      | Write FIFO enable bit.<br>Write FIFO is disabled. The WLVL bit in the Write FIFO status register (WFIFOSTS) is reset to 0 and pointers are initialized, that is, the Write FIFO is “flushed.”<br>Write FIFO is enabled. If Write FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.   |
| 15-8  | WNUMEVT  | 0-FFh<br>0<br>1h<br>2h<br>3h-40h<br>41h-FFh | Write word count per DMA event (32-bit). When the Write FIFO has space for at least WNUMEVT words of data, then an AXEVT (transmit DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as transmitters. This value must be set prior to enabling the Write FIFO.<br>0 words<br>1 word<br>2 words<br>3 to 64 words<br>Reserved |
| 7-0   | WNUMDMA  | 0-FFh<br>0<br>1h<br>2h<br>3h-10h<br>11h-FFh | Write word count per transfer (32-bit words). Upon a transmit DMA event from the McASP, WNUMDMA words are transferred from the Write FIFO to the McASP. This value must equal the number of McASP serializers used as transmitters. This value must be set prior to enabling the Write FIFO.<br>0 words<br>1 word<br>2 words<br>3-16 words<br>Reserved   |

#### 22.4.1.45 Write FIFO Status Register (WFIFOSTS)

The Write FIFO status register (WFIFOSTS) is shown in [Figure 22-83](#) and described in [Table 22-50](#).

**Figure 22-83. Write FIFO Status Register (WFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 22-50. Write FIFO Status Register (WFIFOSTS) Field Descriptions**

| Bit  | Field    | Value   | Description  |
|------|----------|---------|--|
| 31-8 | Reserved | 0       | Reserved   |
| 7-0  | WLVL     | 0-FFh   | Write level (read-only). Number of 32-bit words currently in the Write FIFO. |
|      |          | 0       | 0 words currently in Write FIFO.   |
|      |          | 1h      | 1 word currently in Write FIFO.  |
|      |          | 2h      | 2 words currently in Write FIFO.   |
|      |          | 3h-40h  | 3 to 64 words currently in Write FIFO.                                       |
|      |          | 41h-FFh | Reserved   |

#### 22.4.1.46 Read FIFO Control Register (RFIFOCTL)

The Read FIFO control register (RFIFOCTL) is shown in [Figure 22-84](#) and described in [Table 22-51](#).

**NOTE:** The RNUMEVT and RNUMDMA values must be set prior to enabling the Read FIFO.

If the Read FIFO is to be enabled, it must be enabled prior to taking the McASP out of reset.

**Figure 22-84. Read FIFO Control Register (RFIFOCTL)**

|    |          |   |         |       |
|----|----------|---|---------|-------|
| 31 | Reserved |   | 17      | 16    |
|    |          |   | R-0     | R/W-0 |
| 15 | 8        | 7 | 0       |       |
|    | RNUMEVT  |   | RNUMDMA |       |
|    | R/W-10h  |   | R/W-4h  |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

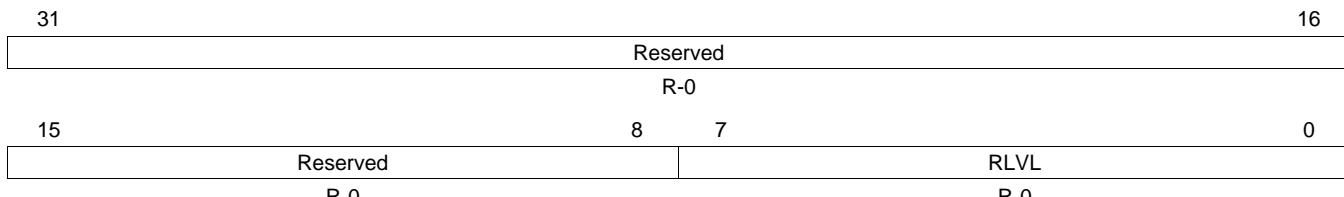
**Table 22-51. Read FIFO Control Register (RFIFOCTL) Field Descriptions**

| Bit   | Field    | Value                                       | Description  |
|-------|----------|---|--|
| 31-17 | Reserved | 0   | Reserved   |
| 16    | RENA     | 0<br>1                                      | Read FIFO enable bit.<br>Read FIFO is disabled. The RLVL bit in the Read FIFO status register (RFIFOSTS) is reset to 0 and pointers are initialized, that is, the Read FIFO is “flushed.”<br>Read FIFO is enabled. If Read FIFO is to be enabled, it must be enabled prior to taking McASP out of reset.   |
| 15-8  | RNUMEVT  | 0-FFh<br>0<br>1h<br>2h<br>3h-40h<br>41h-FFh | Read word count per DMA event (32-bit). When the Read FIFO contains at least RNUMEVT words of data, then an AREVT (receive DMA event) is generated to the host/DMA controller. This value should be set to a non-zero integer multiple of the number of serializers enabled as receivers. This value must be set prior to enabling the Read FIFO.<br>0 words<br>1 word<br>2 words<br>3 to 64 words<br>Reserved |
| 7-0   | RNUMDMA  | 0-FFh<br>0<br>1<br>2<br>3h-10h<br>11h-FFh   | Read word count per transfer (32-bit words). Upon a receive DMA event from the McASP, the Read FIFO reads RNUMDMA words from the McASP. This value must equal the number of McASP serializers used as receivers. This value must be set prior to enabling the Read FIFO.<br>0 words<br>1 word<br>2 words<br>3-16 words<br>Reserved   |

#### 22.4.1.47 Read FIFO Status Register (RFIFOSTS)

The Read FIFO status register (RFIFOSTS) is shown in [Figure 22-85](#) and described in [Table 22-52](#).

**Figure 22-85. Read FIFO Status Register (RFIFOSTS)**



LEGEND: R = Read only; -n = value after reset

**Table 22-52. Read FIFO Status Register (RFIFOSTS) Field Descriptions**

| Bit  | Field    | Value   | Description  |
|------|----------|---------|--|
| 31-8 | Reserved | 0       | Reserved   |
| 7-0  | RLVL     | 0-FFh   | Read level (read-only). Number of 32-bit words currently in the Read FIFO. |
|      |          | 0       | 0 words currently in Read FIFO.  |
|      |          | 1h      | 1 word currently in Read FIFO.   |
|      |          | 2h      | 2 words currently in Read FIFO.  |
|      |          | 3h-40h  | 3 to 64 words currently in Read FIFO.                                      |
|      |          | 41h-FFh | Reserved   |

#### 22.4.2 McASP Data Port Registers

**Table 22-53. McASP Registers Accessed Through Data Port**

| Hex Address    | Register Name | Register Description   |
|----------------|---------------|--|
| Read Accesses  | RBUF          | Receive buffer data port address. Cycles through receive serializers, skipping over transmit serializers and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. DAT BUS only if XBUSEL = 0. |
| Write Accesses | XBUF          | Transmit buffer data port address. Cycles through transmit serializers, skipping over receive and inactive serializers. Starts at the lowest serializer at the beginning of each time slot. DAT BUS only if RBUSEL = 0.            |

## **Controller Area Network (CAN)**

This chapter describes the controller area network for the device.

| Topic                                    | Page        |
|--|-------------|
| <b>23.1 Introduction .....</b>           | <b>3864</b> |
| <b>23.2 Integration .....</b>            | <b>3865</b> |
| <b>23.3 Functional Description .....</b> | <b>3867</b> |
| <b>23.4 DCAN Registers .....</b>         | <b>3905</b> |

## 23.1 Introduction

### 23.1.1 DCAN Features

The general features of the DCAN controller are:

- Supports CAN protocol version 2.0 part A, B (ISO 11898-1)
- Bit rates up to 1 MBit/s
- Dual clock source
- 16, 32, 64 or 128 message objects (instantiated as 64 on this device)
- Individual identifier mask for each message object
- Programmable FIFO mode for message objects
- Programmable loop-back modes for self-test operation
- Suspend mode for debug support
- Software module reset
- Automatic bus on after Bus-Off state by a programmable 32-bit timer
- Message RAM parity check mechanism
- Direct access to Message RAM during test mode
- CAN Rx / Tx pins configurable as general purpose IO pins
- Two interrupt lines (plus additional parity-error interrupt line)
- RAM initialization
- DMA support

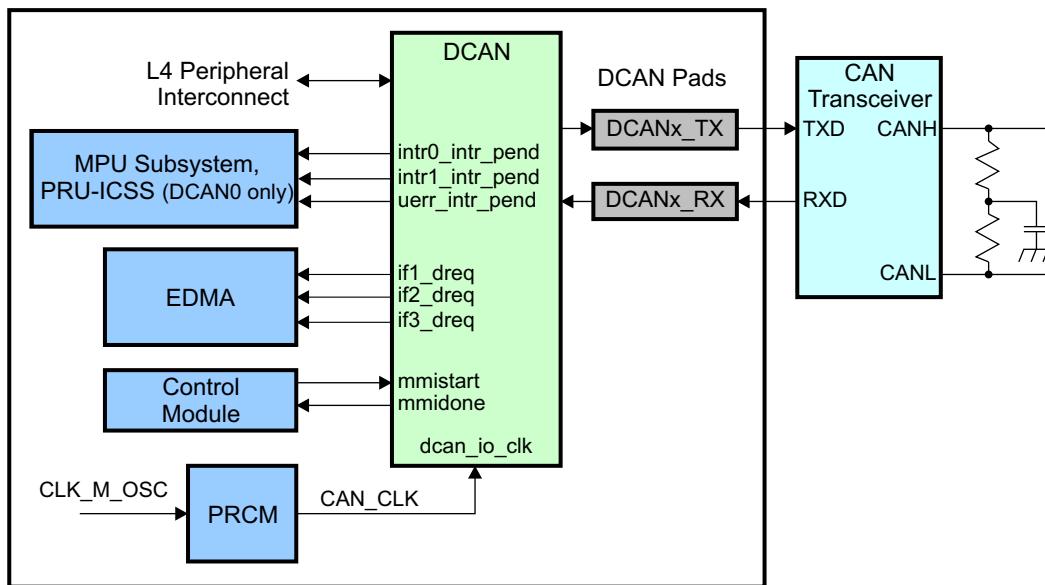
### 23.1.2 Unsupported DCAN Features

The DCAN module in this device does not support GPIO pin mode. All GPIO functionality is mapped through the GPIO modules and muxed at the pins. GPIO pin control signals from the DCAN modules are not connected.

## 23.2 Integration

The Controller Area Network is a serial communications protocol which efficiently supports distributed realtime control with a high level of security. The DCAN module supports bitrates up to 1 Mbit/s and is compliant to the CAN 2.0B protocol specification. The core IP within DCAN is provided by Bosch.

This device includes two instantiations of the DCAN controller: DCAN0 and DCAN1. [Figure 23-1](#) shows the DCAN module integration.



**Figure 23-1. DCAN Integration**

### 23.2.1 DCAN Connectivity Attributes

The general connectivity attributes for the DCAN module are shown in [Table 23-1](#).

**Table 23-1. DCAN Connectivity Attributes**

| Attributes          | Type  |
|---------------------|---|
| Power Domain        | Peripheral Domain   |
| Clock Domain        | PD_PER_L4LS_GCLK (OCP)<br>PD_PER_CAN_CLK (Func)   |
| Reset Signals       | PER_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle  |
| Interrupt Requests  | 3 Interrupts per instance <ul style="list-style-type: none"> <li>Intr0 (DCANx_INT0) – Error, Status, Msg Object interrupt</li> <li>Intr1 (DCANx_INT1) – Msg Object interrupt</li> <li>Uerr (DCANx_PARITY) – Parity error interrupt</li> <li>All DCAN0 interrupts to MPU Subsystem and PRU-ICSS</li> <li>All DCAN1 interrupts to only MPU Subsystem</li> </ul> |
| DMA Requests        | 3 DMA requests per instance to EDMA (CAN_IFxDMA)  |
| Physical Address    | L4 Peripheral slave port  |

### 23.2.2 DCAN Clock and Reset Management

The DCAN controllers have separate bus interface and functional clocks.

**Table 23-2. DCAN Clock Signals**

| Clock Signal                    | Max Freq | Reference / Source | Comments                      |
|---------------------------------|----------|--------------------|-------------------------------|
| DCAN_ocp_clk<br>Interface clock | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>from PRCM |
| DCAN_io_clk<br>Functional clock | 26 MHz   | CLK_M_OSC          | pd_per_can_clk<br>from PRCM   |

### 23.2.3 DCAN Pin List

The external signals for the DCAN module are shown in the following table.

**Table 23-3. DCAN Pin List**

| Pin      | Type | Description        |
|----------|------|--------------------|
| DCANx_TX | O    | DCAN transmit line |
| DCANx_RX | I    | DCAN receive line  |

## 23.3 Functional Description

The DCAN module performs CAN protocol communication according to ISO 11898-1. The bit rate can be programmed to values up to 1 MBit/s. Additional transceiver hardware is required for the connection to the physical layer (CAN bus).

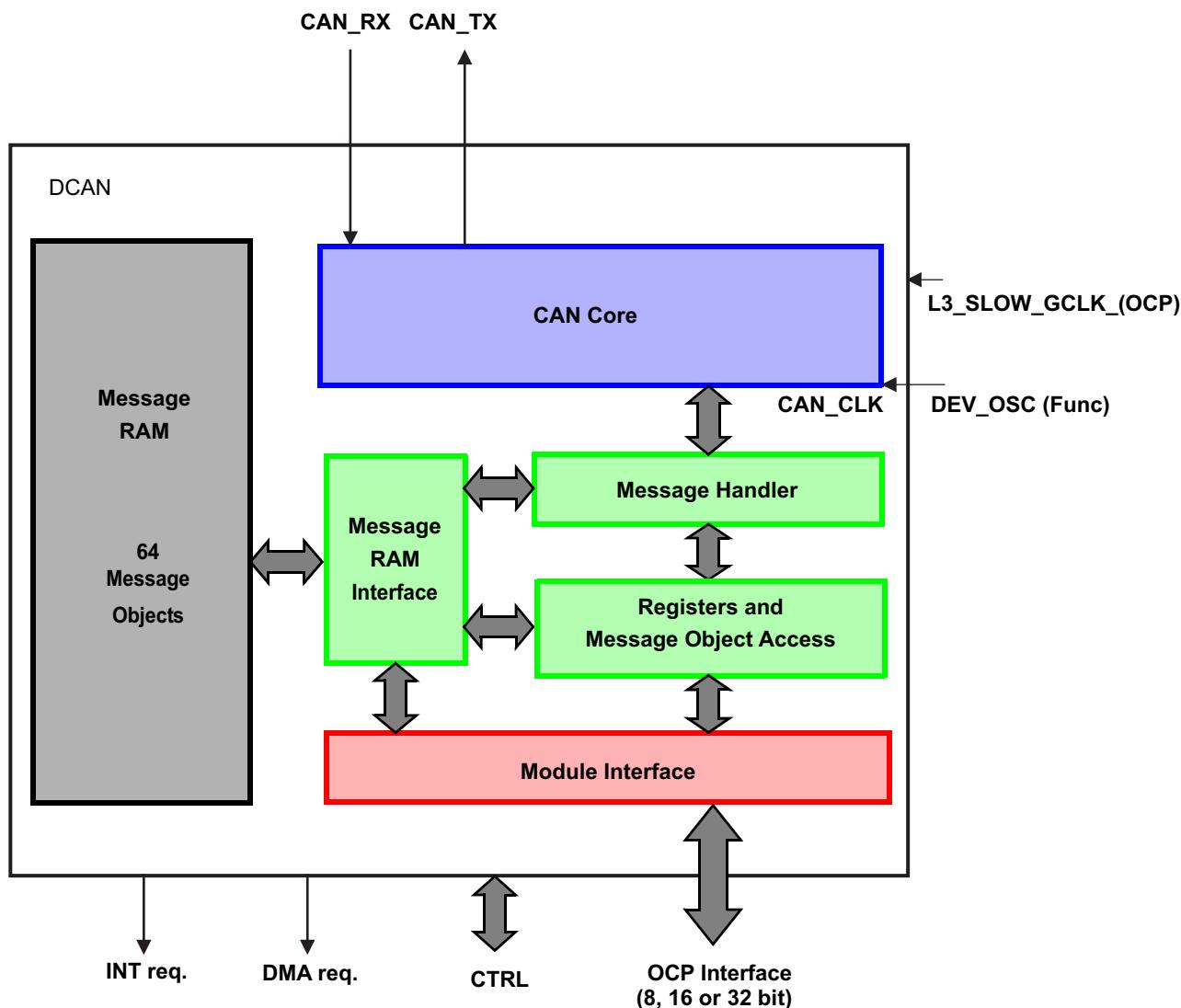
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the message RAM.

All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN core and the message RAM, and the handling of transmission requests, as well as the generation of interrupts or DMA requests.

The register set of the DCAN module can be accessed directly by the CPU via the module interface. These registers are used to control and configure the CAN core and the message handler, and to access the message RAM.

Figure 23-2 shows the DCAN block diagram and its features are described below.

**Figure 23-2. DCAN Block Diagram**



### 23.3.1 CAN Core

The CAN core consists of the CAN protocol controller and the Rx/Tx shift register. It handles all ISO 11898-1 protocol functions.

### 23.3.2 Message Handler

The message handler is a state machine that controls the data transfer between the single-ported message RAM and the CAN core's Rx/Tx shift register. It also handles acceptance filtering and the interrupt/DMA request generation as programmed in the control registers.

### 23.3.3 Message RAM

The DCAN0 and DCAN1 enables a storage of 64 CAN messages.

### 23.3.4 Message RAM Interface

Three interface register sets control the CPU read and write accesses to the message RAM. There are two interface registers sets for read and write access, IF1 and IF2, and one interface register set for read access only, IF3. Additional information can be found in [Section 23.3.15.12](#).

The interface registers have the same word-length as the message RAM.

### 23.3.5 Registers and Message Object Access

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the message RAM are done through interface registers. In a dedicated test mode, the message RAM is memory mapped and can be directly accessed by either CPU or DMA.

### 23.3.6 Module Interface

The DCAN module registers are accessed by the CPU or user software through a 32-bit peripheral bus interface.

### 23.3.7 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (L3\_SLOW\_GCLK) and the peripheral asynchronous clock source domain (CLK\_M\_OSC) for CAN\_CLK.

### 23.3.8 CAN Operation

After hardware reset, the [Init](#) bit in the CAN control register (DCAN CTL) is set and all CAN protocol functions are disabled. The CAN module must be initialized before operating it. [Figure 23-3](#) illustrates the basic initialization flow for the CAN module.

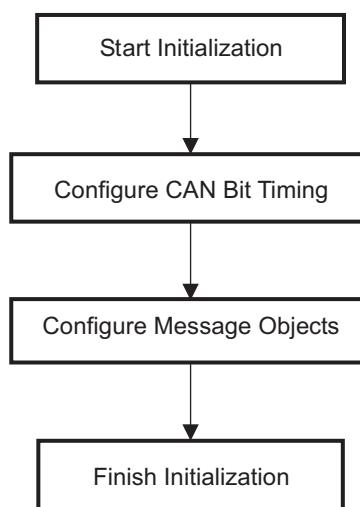
#### 23.3.8.1 CAN Module Initialization

A general CAN module initialization would mean the following two critical steps:

- Configuration of the CAN bit timing
- Configuration of message objects

To initialize the CAN controller, the CPU has to set up the CAN bit timing and those message objects that have to be used for CAN communication. Message objects that are not needed, can be deactivated.

**Figure 23-3. CAN Module General Initialization Flow**



##### 23.3.8.1.1 Configuration of CAN Bit Timing

The CAN module must be in initialization mode to configure the CAN bit timing.

For CAN bit timing software configuration flow, see [Figure 23-4](#).

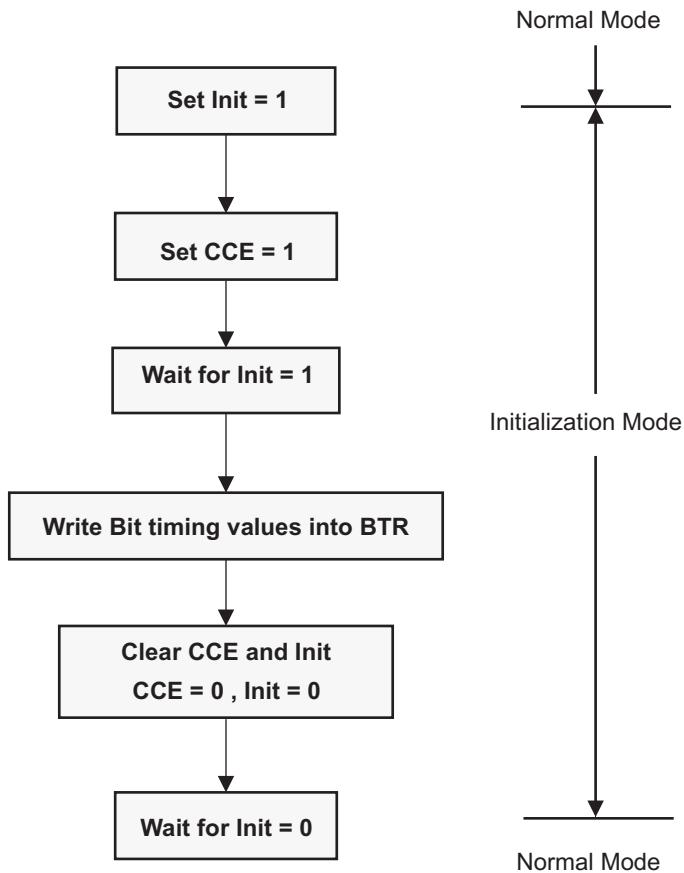
**Step 1:** Enter *initialization mode* by setting the [Init](#) (Initialization) bit in the CAN control register.

While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN\_TX output is recessive (high).

The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

Also, note that the CAN module is in initialization mode on hardware reset and during Bus-Off.

Figure 23-4. CAN Bit-Timing Configuration



**Step 2:** Set the Configure Change Enable ( **CCE** ) bit in the CAN control register.

The access to the Bit Timing register (BTR) for the configuration of the bit timing is enabled when both **Init** and **CCE** bits in the CAN Control register are set.

**Step 3:** Wait for the Init bit to get set. This would make sure that the module has entered *Initialization mode*.

**Step 4:** Write the bit timing values into the bit timing register. See [Section 23.3.16.2](#) for the BTR value calculation for a given bit timing.

**Step 5:** Clear the CCE and Init bit.

**Step 6:** Wait for the Init bit to clear. This would ensure that the module has come out of *initialization mode*.

Following these steps, the module comes to operation by synchronizing itself to the CAN bus, provided the BTR is configured as per the CAN bus baud rate, although the message objects have to be configured before carrying out any communication.

---

**NOTE:** The module will not come out of the *initialization mode* if any incorrect BTR values are written in step 4.

---



---

**NOTE:** The required message objects should be configured as transmit or receive objects before the start of data transfer as explained in [Section 23.3.8.1](#).

---

### 23.3.8.1.2 Configuration of Message Objects

The message objects can be configured only through the interface registers; the CPU does not have direct access to the message object (message RAM). Familiarize yourself with the interface register set (IFx) usage (see [Section 23.3.17](#)) and the message object structure (see [Section 23.3.18](#)) before configuring the message objects.

For more information regarding the procedure to configure the message objects, see [Section 23.3.14](#). All the message objects should be configured to particular identifiers or set to not valid before the message transfer is started. It is possible to change the configuration of message objects during normal operation (that is between data transfers).

---

**NOTE:** The message objects initialization is independent of the bit-timing configuration.

---

### 23.3.8.1.3 DCAN RAM Hardware Initialization

The memory hardware initialization for the DCAN module is enabled in the device control register, [DCAN\\_RAMINIT](#), which initializes the RAM with zeros and sets parity bits accordingly. Wait for the RAMINIT\_DONE bit to be set to ensure successful RAM initialization. Ensure the clock to the DCAN module is enabled before starting this initialization.

For more details on RAM hardware initialization, see [Chapter 9, Control Module](#).

### 23.3.8.2 CAN Message Transfer (Normal Operation)

Once the DCAN is initialized and [Init](#) bit is reset to zero, the CAN core synchronizes itself to the CAN bus and is ready for message transfer as per the configured message objects.

The CPU may enable the interrupt lines (setting IE0 and IE1 to '1') at the same time when it clears [Init](#) and [CCE](#). The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication can be carried out in any of the following two modes: interrupt and polling.

The interrupt register points to those message objects with [IntPnd](#) = '1'. It is updated even if the interrupt lines to the CPU are disabled (IE0/IE1 are zero).

The CPU may poll all [MessageObject](#)'s [NewDat](#) and [TxRqst](#) bits in parallel from the [NewData X](#) registers and the [Transmission Request X Registers](#) (DCAN TXRQ X). Polling can be made easier if all transmit objects are grouped at the low numbers and all receive objects are grouped at the high numbers.

Received messages are stored into their appropriate message objects if they pass acceptance filtering.

The whole message (including all arbitration bits, DLC and up to eight data bytes) is stored into the message object. As a consequence (e.g., when the identifier mask is used), the arbitration bits which are masked to "don't care" may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the interface registers, as the message handler guarantees data consistency in case of concurrent accesses.

If a permanent message object (arbitration and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes.

If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority.

Messages may be updated or set to not valid at any time, even if a requested transmission is still pending. However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

### 23.3.8.2.1 Automatic Retransmission

According to the CAN Specification (ISO11898), the DCAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting bit disable automatic retransmission ([DAR](#)) in the CAN Control register. Further details to this mode are provided in [Section 23.3.15.3](#).

### 23.3.8.2.2 Auto-Bus-On

By default, after the DCAN has entered Bus-Off state, the CPU can start a Bus-Off-Recovery sequence by resetting the Init bit. If this is not done, the module will stay in Bus-Off state.

The DCAN provides an automatic Auto-Bus-On feature which is enabled by bit [ABO](#) in the CAN control register. If set, the DCAN will automatically start the Bus-Off-Recovery sequence. The sequence can be delayed by a user-defined number of L3\_SLOW\_GCLK cycles which can be defined in the Auto-Bus-On Time register (DCAN ABOTR).

---

**NOTE:** If the DCAN goes to Bus-Off state due to a massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit has been reset by the CPU or due to the Auto-Bus-On feature, the device will wait for 129 occurrences of bus Idle (equal to  $129 * 11$  consecutive recessive bits) before resuming normal operation. At the end of the Bus-Off recovery sequence, the error counters will be reset.

---

### 23.3.8.3 Test Modes

The DCAN module provides several test modes which are mainly intended for production tests or self test.

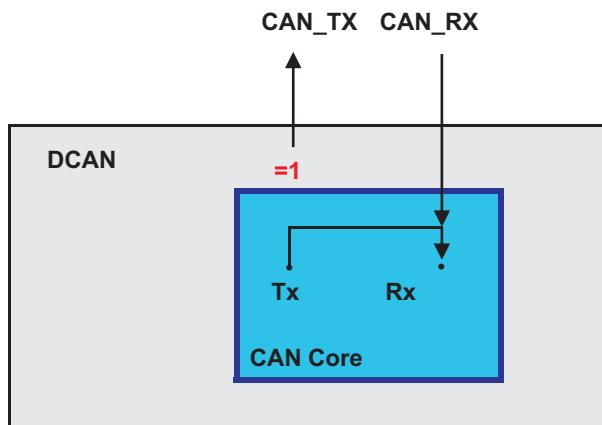
For all test modes, the [Test](#) bit in the CAN control register needs to be set to one. This enables write access to the test register (DCAN TEST).

#### 23.3.8.3.1 Silent Mode

The silent mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (e.g., acknowledge bit, overload flag, active error flag). The DCAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, these are internally routed to the CAN core.

[Figure 23-5](#) shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in silent mode. Silent mode can be activated by setting the [Silent](#) bit in the Test register to one. In ISO 11898-1, the silent mode is called the bus monitoring mode.

**Figure 23-5. CAN Core in Silent Mode**



### 23.3.8.3.2 Loopback Mode

The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN\_RX input pin is disregarded by the CAN core. Transmitted messages can still be monitored at the CAN\_TX pin.

In order to be independent from external stimulation, the CAN core ignores acknowledge sampled in the acknowledge slot of a data/remote frame.

[Figure 23-6](#) shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in loopback mode.

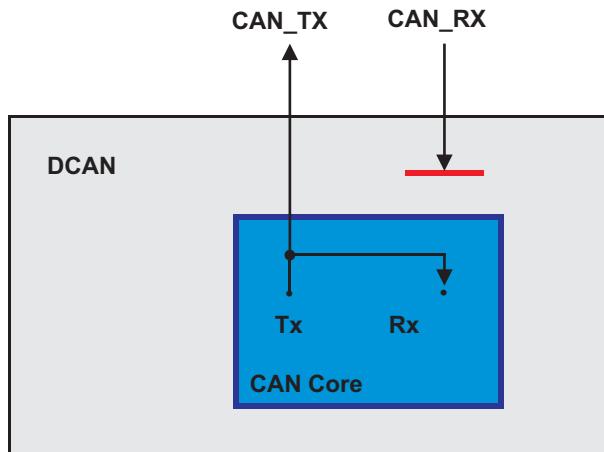
Loopback mode can be activated by setting bit [LBack](#) in the test register to one.

---

**NOTE:** In loopback mode, the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN core are disregarded. For including these into the testing, see [Section 23.3.8.3.3](#).

---

**Figure 23-6. CAN Core in Loopback Mode**



### 23.3.8.3.3 External Loopback Mode

The external loopback mode is similar to the loopback mode; however, it includes the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN core. When external loopback mode is selected, the input of the CAN core is connected to the input buffer of the Tx pin.

With this configuration, the Tx pin IO circuit can be tested.

External loopback mode can be activated by setting bit **ExL** in test register to one.

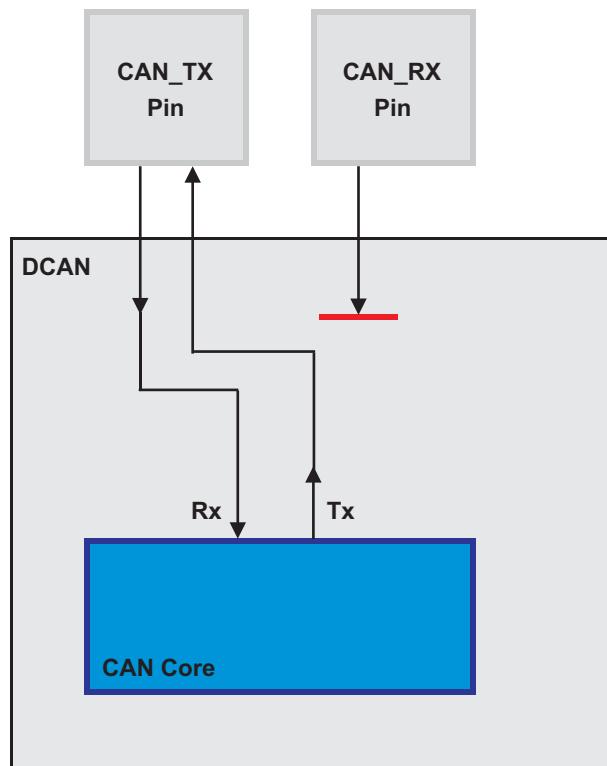
[Figure 23-7](#) shows the connection of signals CAN\_TX and CAN\_RX to the CAN core in external loopback mode.

---

**NOTE:** When loopback mode is active (LBack bit set), the ExL bit will be ignored.

---

**Figure 23-7. CAN Core in External Loopback Mode**

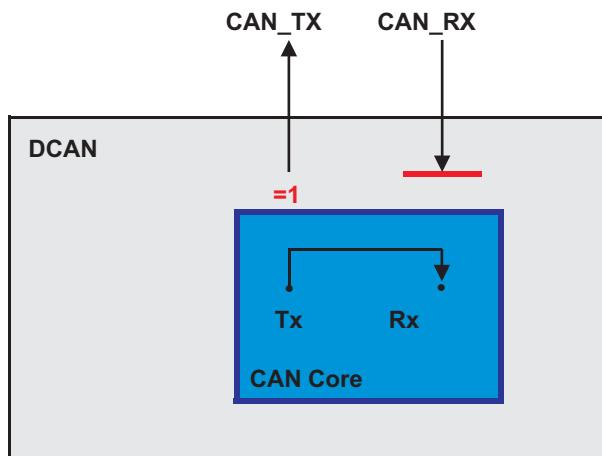


### 23.3.8.3.4 Loopback Mode Combined With Silent Mode

It is also possible to combine loopback mode and silent mode by setting bits **LBack** and **Silent** at the same time. This mode can be used for a “Hot Selftest,” i.e., the DCAN hardware can be tested without affecting the CAN network. In this mode, the CAN\_RX pin is disconnected from the CAN core and no dominant bits will be sent on the CAN\_TX pin.

[Figure 23-8](#) shows the connection of the signals CAN\_TX and CAN\_RX to the CAN core in case of the combination of loopback mode with silent mode.

**Figure 23-8. CAN Core in Loop Back Combined With Silent Mode**



### 23.3.8.3.5 Software Control of CAN\_TX pin

Four output functions are available for the CAN transmit pin CAN\_TX. In addition to its default function (serial data output), the CAN\_TX pin can drive constant dominant or recessive values, or it can drive the CAN sample point signal to monitor the CAN core's bit timing.

Combined with the readable value of the CAN\_RX pin, this function can be used to check the physical layer of the CAN bus.

The output mode of pin CAN\_TX is selected by programming the test register bits Tx[1:0] as described in section [Section 23.4.1.6](#).

---

**NOTE:** The software control for the CAN\_TX pin interferes with CAN protocol functions. For CAN message transfer or any of the test modes (loopback mode, external loopback mode or silent mode), the CAN\_TX pin should operate in its default functionality.

---

### 23.3.9 Dual Clock Source

Two clock domains are provided to the DCAN module: the peripheral synchronous clock domain (L3\_SLOW\_GCLK) as the general module clock source, and the peripheral asynchronous clock source domain (CLK\_M\_OSC) provided to the CAN core (as clock source CAN\_CLK) for generating the CAN bit timing.

Both clock domains can be derived from the same clock source (so that L3\_SLOW\_GCLK = CLK\_M\_OSC).

For more information on how to configure the relevant clock source registers in the system module, see [Chapter 8, Power and Clock Management](#).

Between the two clock domains, a synchronization mechanism is implemented in the DCAN module in order to ensure correct data transfer.

---

**NOTE:** If the dual clock functionality is used, then L3\_SLOW\_GCLK must always be higher or equal to CAN\_CLK (CLK\_M\_OSC) (derived from the asynchronous clock source), in order to achieve a stable functionality of the DCAN. Here also the frequency shift of the modulated L3\_SLOW\_GCLK has to be considered:

$$f_{0, \text{L3\_SLOW\_GCLK(OCP)}} \pm \Delta f_{\text{FM}, \text{L3\_SLOW\_GCLK(OCP)}} \geq f_{\text{CANCLK}}$$


---

---

**NOTE:** The CAN core has to be programmed to at least 8 clock cycles per bit time. To achieve a transfer rate of 1 MBaud when using the asynchronous clock domain as the clock source for CAN\_CLK (CLK\_M\_OSC), an oscillator frequency of 8 MHz or higher has to be used.

---

### 23.3.10 Interrupt Functionality

Interrupts can be generated on two interrupt lines: DCANINT0 and DCANINT1. These lines can be enabled by setting the **IE0** and **IE1** bits, respectively, in the CAN control register. The interrupts are level triggered at the chip level.

The DCAN provides three groups of interrupt sources: message object interrupts, status change interrupts, and error interrupts (see [Figure 23-9](#) and [Figure 23-10](#)).

The source of an interrupt can be determined by the interrupt identifiers Int0ID/Int1ID in the interrupt register (see [Section 23.4.1.5](#)). When no interrupt is pending, the register will hold the value zero.

Each interrupt line remains active until the dedicated field in the interrupt register **DCAN INT** (Int0ID / Int1ID) again reach zero (this means the cause of the interrupt is reset), or until **IE0** / **IE1** are reset.

The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN core has updated (not necessarily changed) the Error and Status register (DCAN ES). This interrupt has the highest priority. The CPU can update (reset) the status bits WakeUpPnd, RxOk, TxOk and LEC by reading the error and status register **DCAN ES**, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID resp. Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority; the last message object has the lowest priority.

An interrupt service routine that reads the message that is the source of the interrupt may read the message and reset the message object's IntPnd at the same time (ClrlntPnd bit in the IF1/IF2 command register). When IntPnd is cleared, the interrupt register will point to the next message object with a pending interrupt.

#### 23.3.10.1 Message Object Interrupts

Message object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxE and RxE that are described in [Section 23.3.18.1](#).

Message object interrupts can be routed to either DCANINT0 or DCANINT1 line, controlled by the interrupt multiplexer register (DCAN INTMUX12 to DCAN INTMUX78), see [Section 23.4.1.17](#).

#### 23.3.10.2 Status Change Interrupts

The events WakeUpPnd, RxOk, TxOk and LEC in error and status register (**DCAN ES**) belong to the status change interrupts. The status change interrupt group can be enabled by bit in CAN control register.

If **SIE** is set, a status change interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the message RAM configuration.

Status change interrupts can only be routed to interrupt line DCANOINT, which has to be enabled by setting **IE0** in the CAN control register.

---

**NOTE:** Reading the error and status register will clear the WakeUpPnd flag. If in global power-down mode, the WakeUpPnd flag is cleared by such a read access before the DCAN module has been waken up by the system, the DCAN may re-assert the WakeUpPnd flag, and a second interrupt may occur.

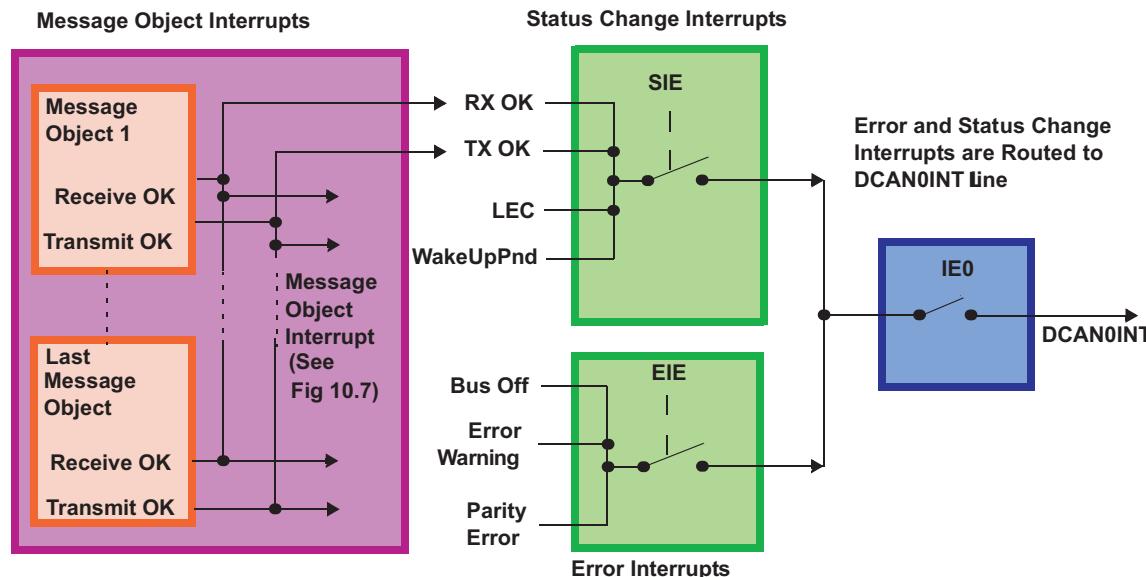
---

### 23.3.10.3 Error Interrupts

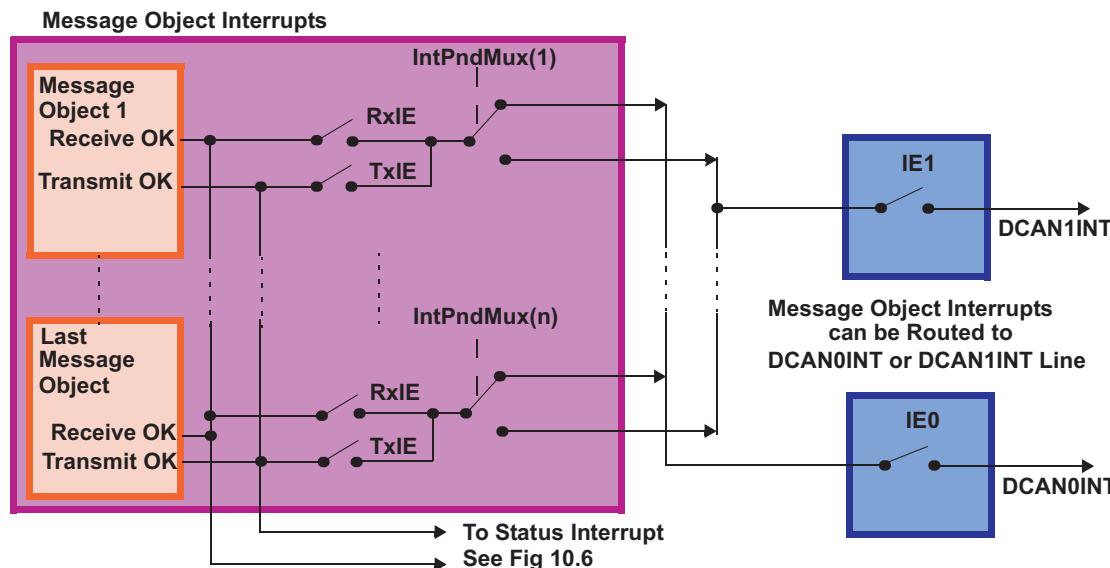
The events PER, BOff and EWarn, monitored in the Error and Status register, **DCAN ES**, belong to the error interrupts. The error interrupt group can be enabled by setting bit **EIE** in the CAN Control register.

Error interrupts can only be routed to interrupt line DCAN0INT, which has to be enabled by setting **IE0** in the CAN Control register.

**Figure 23-9. CAN Interrupt Topology 1**



**Figure 23-10. CAN Interrupt Topology 2**



### 23.3.11 Local Power-Down Mode

The DCAN supports a local power-down mode, which can be controlled within the DCAN control registers.

#### 23.3.11.1 Entering Local Power-Down Mode

The local power-down mode is requested by setting the [PDR](#) bit in CAN Control register.

The DCAN then finishes all transmit requests of the message objects. When all requests are done, DCAN waits until a bus idle state is recognized. Then it will automatically set the [Init](#) bit in CAN control register to prevent any further CAN transfers, and it will also set the [PDA](#) bit in CAN error and status register. With setting the PDA bits, the DCAN module indicates that the local power-down mode has been entered.

During local power-down mode, the internal clocks of the DCAN module are turned off, but there is a wakeup logic (see [Section 23.3.11.2](#)) that can be active, if enabled. Also, the actual contents of the control registers can be read back.

---

**NOTE:** In local low-power mode, the application should not clear the Init bit while PDR is set. If there are any messages in the message RAM which are configured as transmit messages and the application resets the init bit, these messages may get sent.

---

#### 23.3.11.2 Wakeup From Local Power Down

There are two ways to wake up the DCAN from local power-down mode:

- The application could wake up the DCAN module manually by clearing the [PDR](#) bit and then clearing the [Init](#) bit in CAN Control register.
- Alternatively, a CAN bus activity detection circuit can be activated by setting the wakeup on bus activity bit ([WUBA](#)) in the CAN control register. If this circuit is active, on occurrence of a dominant CAN bus level, the DCAN will automatically start the wakeup sequence. It will clear the PDR bit in the CAN Control register and also clear the PDA bit in the error and status register. The [WakeUpPnd](#) bit in CAN error and status register will be set. If status interrupts are enabled, also an interrupt will be generated. Finally the [Init](#) bit in CAN control register will be cleared.

After the Init bit has been cleared, the module waits until it detects 11 consecutive recessive bits on the CAN\_RX pin and then goes bus-active again.

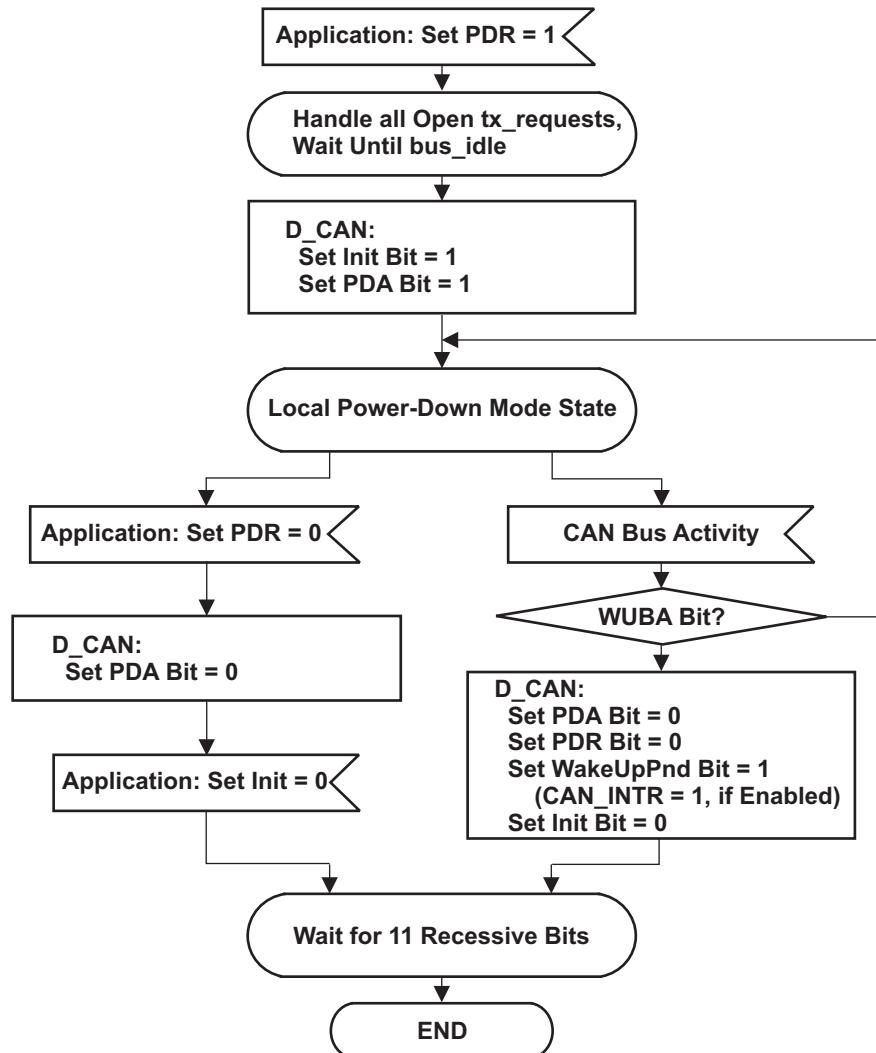
---

**NOTE:** The CAN transceiver circuit has to stay active in order to detect any CAN bus activity while the DCAN is in local power down mode. The first CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power-down and automatic wake-up mode, is lost.

---

[Figure 23-11](#) shows a flow diagram about entering and leaving local power-down mode.

**Figure 23-11. Local Power-Down Mode Flow Diagram**



### 23.3.12 Parity Check Mechanism

The DCAN provides a parity check mechanism to ensure data integrity of message RAM data. For each word (32 bits) in message RAM, one parity bit will be calculated. The formation of the different words is according to the message RAM representation in RDA mode, see [Section 23.3.18.4](#).

Parity information is stored in the message RAM on write accesses and will be checked against the stored parity bit from message RAM on read accesses.

The Parity check functionality can be enabled or disabled by [PMD](#) bit field in the CAN control register.

In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the DCAN. The parity bits could be read in debug/suspend mode (see [Section 23.3.18.3](#)) or in RDA mode (see [Section 23.3.18.4](#)). However, direct write access to the parity bits is only possible in these two modes with parity check disabled.

A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: The parity bit will be set, if the number of 1 bits in the data is odd.

---

**NOTE:** The parity scheme is tied to even parity at the device level.

---

#### 23.3.12.1 Behavior on Parity Error

On any read access to message RAM (e.g., during start of a CAN frame transmission), the parity of the message object will be checked. If a parity error is detected, the [PER](#) bit in the error and status register will be set. If error interrupts are enabled, an interrupt would also be generated. In order to avoid the transmission of invalid data over the CAN bus, the D bit of the message object will be reset.

The message object data can be read by the host CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, e.g., by immediately checking the parity error code register (DCAN PERR) on parity error interrupt.

---

**NOTE:** During RAM initialization, no parity check will be done.

---

#### 23.3.12.2 Parity Testing

Testing the parity mechanism can be done by enabling the bit RamDirectAccess ([RDA](#)) and manually writing the parity bits directly to the dedicated RAM locations. With this, data and parity bits could be checked when reading directly from RAM.

---

**NOTE:** If parity check was disabled, the application has to ensure correct parity bit handling in order to prevent parity errors later on when parity check is enabled.

---

### 23.3.13 Debug/Suspend Mode

The module supports the usage of an external debug unit by providing functions like pausing DCAN activities and making message RAM content accessible via OCP interface.

Before entering debug/suspend mode, the circuit will either wait until a started transmission or reception will be finished and bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit **IDS** in the CAN control register.

Afterwards, the DCAN enters debug/suspend mode, indicated by **InitDbg** flag in the CAN control register.

During debug/suspend mode, all DCAN registers can be accessed. Reading reserved bits will return '0'. Writing to reserved bits will have no effect.

Also, the message RAM will be memory mapped. This allows the external debug unit to read the message RAM. For the memory organization, see [Section 23.3.18.3](#).

---

**NOTE:** During debug/suspend mode, the message RAM cannot be accessed via the IFx register sets.

---

---

**NOTE:** Writing to control registers in debug/suspend mode may influence the CAN state machine and further message handling.

---

For debug support, the auto clear functionality of the following DCAN registers is disabled:

- Error and status register (clear of status flags by read)
- IF1/IF2 command registers (clear of DMAActive flag by read/write)

### 23.3.14 Configuration of Message Objects

The whole message RAM should be configured before the end of the initialization, however it is also possible to change the configuration of message objects during CAN communication.

The CAN software driver library should offer subroutines that:

- Transfer a complete message structure into a message object. (Configuration)
- Transfer the data bytes of a message into a message object and set TxRqst and NewDat. (Start a new transmission)
- Get the data bytes of a message from a message object and clear NewDat (and IntPnd). (Read received data)
- Get the complete message from a message object and clear NewDat (and IntPnd). (Read a received message, including identifier, from a message object with UMask = '1')

Parameters of the subroutines are the Message Number and a pointer to a complete message structure or to the data bytes of a message structure.

Two examples of assigning the IFx interface register sets to these subroutines are shown here:

In the first method, the tasks of the application program that may access the module are divided into two groups. Each group is restricted to the use of one of the interface register sets. The tasks of one group may interrupt tasks of the other group, but not of the same group.

In the second method, which may be a special case of the first method, there are only two tasks in the application program that access the module. A Read\_Message task that uses IF2 or IF3 to get received messages from the message RAM and a Write\_Message task that uses IF1 to write messages to be transmitted (or to be configured) into the message RAM. Both tasks may interrupt each other.

### 23.3.14.1 Configuration of a Transmit Object for Data Frames

[Table 23-4](#) shows how a transmit object can be initialized.

**Table 23-4. Initialization of a Transmit Object**

| MsgVal | Arb   | Data  | Mask  | EoB | Dir | NewDat | MsgLst | RxIE | TxIE  | IntPnd | RmtEn | TxRqst |
|--------|-------|-------|-------|-----|-----|--------|--------|------|-------|--------|-------|--------|
| 1      | appl. | appl. | appl. | 1   | 1   | 0      | 0      | 0    | appl. | 0      | appl. | 0      |

The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.

The data registers (DLC[3:0] and Data0-7) are given by the application. TxRqst and RmtEn should not be set before the data is valid.

If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.

If the RmtEn bit is set, a matching received remote frame will cause the TxRqst bit to be set; the remote frame will autonomously be answered by a data frame.

The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask='1') to allow groups of remote frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details, see [Section 23.3.15.8](#). Identifier masking must be disabled (UMask = '0') if no remote frames are allowed to set the TxRqst bit (RmtEn = '0').

### 23.3.14.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure transmit objects for the transmission of remote frames. Setting TxRqst for a receive object causes the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

### 23.3.14.3 Configuration of a Single Receive Object for Data Frames

[Table 23-5](#) shows how a receive object for data frames can be initialized.

**Table 23-5. Initialization of a single Receive Object for Data Frames**

| MsgVal | Arb   | Data  | Mask  | EoB | Dir | NewDat | MsgLst | RxIE  | TxIE | IntPnd | RmtEn | TxRqst |
|--------|-------|-------|-------|-----|-----|--------|--------|-------|------|--------|-------|--------|
| 1      | appl. | appl. | appl. | 1   | 0   | 0      | 0      | appl. | 0    | 0      | 0     | 0      |

The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a data frame with an 11-bit Identifier is received, ID[17:0] is set to '0'.

The data length code (DLC[3:0]) is given by the application. When the message handler stores a data frame in the message object, it will store the received data length code and eight data bytes. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.

The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = '1') to allow groups of data frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the mask bits are set to "don't care," the corresponding bits of the arbitration register will be overwritten by the bits of the stored data frame.

If the RxIE bit is set, the IntPnd bit will be set when a received data frame is accepted and stored in the message object.

If the TxRqst bit is set, the transmission of a remote frame with the same identifier as actually stored in the arbitration bits will be triggered. The content of the arbitration bits may change if the mask bits are used (UMask = '1') for acceptance filtering.

### 23.3.14.4 Configuration of a Single Receive Object for Remote Frames

Table 23-6 shows how a receive object for remote frames can be initialized.

**Table 23-6. Initialization of a Single Receive Object for Remote Frames**

| MsgVal | Arb   | Data  | Mask  | EoB | Dir | NewDat | MsgLst | RxIE  | TxIE | IntPnd | RmtEn | TxRqst |
|--------|-------|-------|-------|-----|-----|--------|--------|-------|------|--------|-------|--------|
| 1      | appl. | appl. | appl. | 1   | 1   | 0      | 0      | appl. | 0    | 0      | 0     | 0      |

A receive object for remote frames may be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object will not trigger the transmission of a data frame. Receive objects for remote frames may be expanded to a FIFO buffer (see [Section 23.3.14.5](#)).

UMask must be set to '1.' The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to "must-match" or to "don't care," to allow groups of remote frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details, see [Section 23.3.15.8](#).

The arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received remote frames. If some bits of the mask bits are set to "don't care," the corresponding bits of the arbitration bits will be overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a remote frame with an 11-bit Identifier is received, ID[17:0] will be set to '0.'

The data length code (DLC[3:0]) may be given by the application. When the message handler stores a remote frame in the message object, it will store the received data length code. The data bytes of the message object will remain unchanged.

If the RxIE bit is set, the IntPnd bit will be set when a received remote frame is accepted and stored in the message object.

### 23.3.14.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of receive objects belonging to a FIFO buffer is the same as the configuration of a single receive object.

To concatenate multiple message objects to a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO Buffer is set to one, configuring it as the end of the block.

### 23.3.15 Message Handling

When initialization is finished, the DCAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application has to update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching remote frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten.

Messages may be read interrupt-driven or after polling of NewDat.

#### 23.3.15.1 Message Handler Overview

The message handler state machine controls the data transfer between the Rx/Tx shift register of the CAN core and the message RAM. It performs the following tasks:

- Data transfer from message RAM to CAN core (messages to be transmitted)
- Data transfer from CAN core to the message RAM (received messages)
- Data transfer from CAN core to the acceptance filtering unit
- Scanning of message RAM for a matching message object (acceptance filtering)
- Scanning the same message object after being changed by IF1/IF2 registers when priority is the same or higher as the message the object found by last scanning
- Handling of TxRqst flags
- Handling of interrupt flags

The message handler registers contains status flags of all message objects grouped into the following topics:

- Transmission Request Flags
- New Data Flags
- Interrupt Pending Flags
- Message Valid Registers

Instead of collecting above listed status information of each message object via IFx registers separately, these message handler registers provides a fast and easy way to get an overview, for example, about all pending transmission requests.

All message handler registers are read-only.

#### 23.3.15.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while the last implemented message object has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object so messages with the highest priority, for example, can be placed in the message objects with the lowest numbers.

The acceptance filtering for received data frames or remote frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher message number. The last message object may be configured to accept any data frame or remote frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

#### 23.3.15.3 Transmission of Messages in Event-Driven CAN Communication

If the shift register of the CAN core is ready for loading and if there is no data transfer between the IFx registers and message RAM, the D bits in the Message Valid register (DCAN MSGVAL12 to DCAN MSGVAL78) and the TxRqst bits in the transmission request register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the message handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the DCAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If automatic retransmission mode is disabled by setting the **DAR** bit in the CAN control register, the behavior of bits TxRqst and NewDat in the Message Control register of the interface register set is as follows:

- When a transmission starts, the TxRqst bit of the respective interface register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received remote frames do not require a receive object. They will automatically trigger the transmission of a data frame, if in the matching transmit object the RmtEn bit is set.

#### 23.3.15.4 Updating a Transmit Object

The CPU may update the data bytes of a transmit object any time via the IF1/IF2 interface registers, neither D nor TxRqst have to be reset before the update.

Even if only part of the data bytes is to be updated, all four bytes in the corresponding IF1/IF2 Data A register (DCAN IF1DATA/DCAN IF2DATA) or IF1/IF2 Data B register (DCAN IF1DATB/DCAN IF2DATB) have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 data register or the message object is transferred to the IF1/IF2 data register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the IF1/IF2 Command register (DCAN IF1CMD/DCAN IF2CMD) and then the number of the message object is written to bits [7:0] of the command register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details, see [Section 23.3.15.3](#).

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

#### 23.3.15.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the transmit objects may be managed dynamically. The CPU can write the whole message (arbitration, control, and data) into the interface register. The bits [23:16] of the command register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither D nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however, it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the command register should be set to 0x87.

---

**NOTE:** After the update of the transmit object, the interface register set will contain a copy of the actual contents of the object, including the part that had not been updated.

---

### 23.3.15.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message are completely shifted into the shift register of the CAN core, the message handler starts the scan of the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of message object 1 are loaded into the acceptance filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

### 23.3.15.7 Reception of Data Frames

The message handler stores the message from the CAN core shift register into the respective message object in the message RAM. Not only the data bytes, but all arbitration bits and the data length code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the interrupt register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

### 23.3.15.8 Reception of Remote Frames

When a remote frame is received, three different configurations of the matching message object have to be considered:

- Dir = '1' (direction = transmit), RmtEn = '1', UMask = '1' or '0'  
The TxRqst bit of this message object is set at the reception of a matching remote frame. The rest of the message object remains unchanged.
- Dir = '1' (direction = transmit), RmtEn = '0', UMask = '0'  
The remote frame is ignored, this message object remains unchanged.
- Dir = '1' (direction = transmit), RmtEn = '0', UMask = '1'

The remote frame is treated similar to a received data frame. At the reception of a matching Remote Message Frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged.

### 23.3.15.9 Reading Received Messages

The CPU may read a received message any time via the IFx interface register. The data consistency is guaranteed by the message handler state machine. Typically the CPU will write first 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the command register. That combination will transfer the entire received message from the message RAM into the interface register set. Additionally, the bits NewDat and IntPnd are cleared in the message RAM (not in the interface register set). The values of these bits in the message control register always reflect the status before resetting the bits. If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message has been received since the last time when this message object was read. MsgLst will not be automatically reset.

### 23.3.15.10 Requesting New Data for a Receive Object

By means of a remote frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a remote frame with the identifier of the receive object. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TxRqst bit is automatically reset. Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the command register.

### 23.3.15.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO buffers. Each FIFO buffer configured to store received messages with a particular (group of) identifier(s). Arbitration and mask registers of the FIFO buffer's message objects are identical. The end of buffer (EoB) bits of all but the last of the FIFO buffer's message objects are '0'; in the last one the EoB bit is '1.'

Received messages with identifiers matching to a FIFO buffer are stored into a message object of this FIFO buffer, starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NewDat bit of this message object is set. By setting NewDat while EoB is '0', the message object is locked for further write accesses by the message handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. If none of the preceding message objects is released by writing NewDat to '0,' all further messages for this FIFO buffer will be written into the last message object of the FIFO buffer (EoB = '1') and therefore overwrite previous messages in this message object.

### 23.3.15.12 Reading From a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO buffer before the application program is required (in order to avoid the loss of data) to empty the buffer.

A FIFO buffer of length N will store  $-1$  plus the last received message since last time it was cleared.

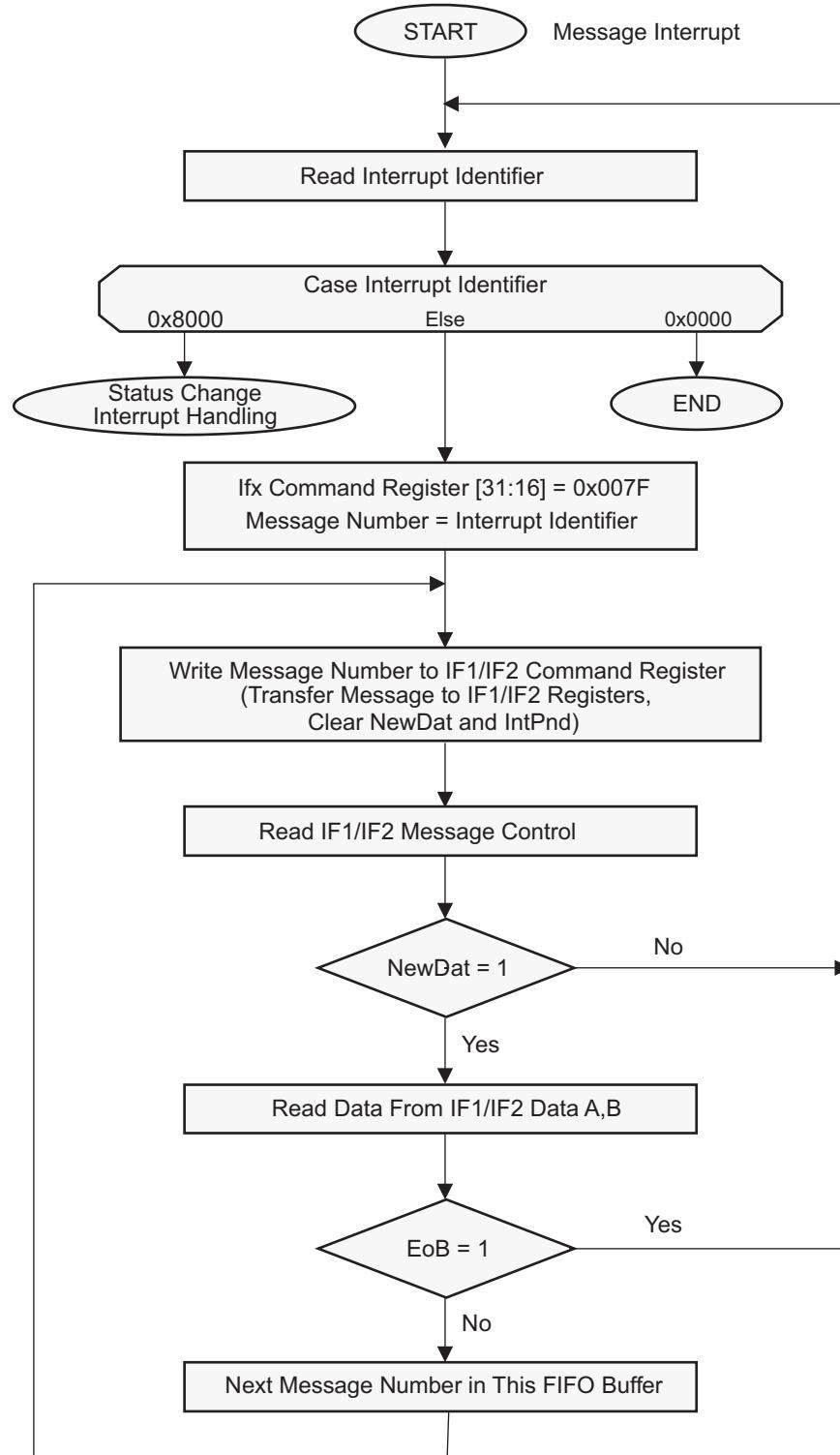
A FIFO buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in [Figure 23-12](#).

---

**NOTE:** All message objects of a FIFO buffer need to be read and cleared before the next batch of messages can be stored. Otherwise, true FIFO functionality can not be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

---

Reading from a FIFO buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

**Figure 23-12. CPU Handling of a FIFO Buffer (Interrupt Driven)**


### 23.3.16 CAN Bit Timing

The DCAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The bit timing parameters can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods ( $f_{osc}$ ) may be different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range ( $\delta f$ ), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

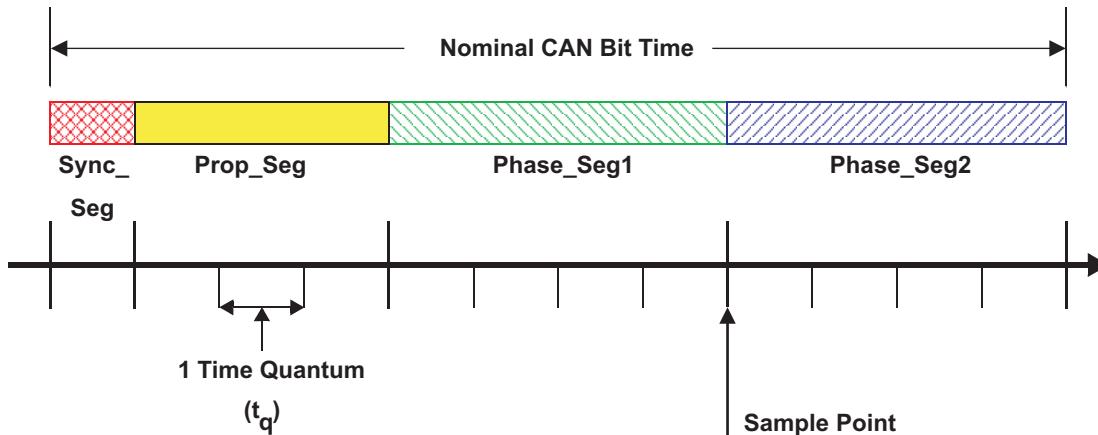
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

#### 23.3.16.1 Bit Time and Bit Rate

According to the CAN specification, the bit time is divided into four segments (see [Figure 23-13](#)):

- Synchronization segment (Sync\_Seg)
- Propagation time segment (Prop\_Seg)
- Phase buffer segment 1 (Phase\_Seg1)
- Phase buffer segment 2 (Phase\_Seg2)

**Figure 23-13. Bit Timing**



Each segment consists of a specific number of time quanta. The length of one time quantum ( $t_q$ ), which is the basic time unit of the bit time, is given by the CAN\_CLK and the baud rate prescalers (BRPE and BRP). With these two baud rate prescalers combined, divider values from 1 to 1024 can be programmed:

$$t_q = \text{Baud Rate Prescaler} / \text{CAN\_CLK}$$

Apart from the fixed length of the synchronization segment, these numbers are programmable. [Table 23-7](#) describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

**Table 23-7. Parameters of the CAN Bit Time**

| Parameter                        | Range           | Remark   |
|----------------------------------|-----------------|--|
| Sync_Seg                         | $1 t_q$ (fixed) | Synchronization of bus input to CAN_CLK            |
| Prop_Seg                         | [1 ... 8] $t_q$ | Compensates for the physical delay times           |
| Phase_Seg1                       | [1 ... 8] $t_q$ | May be lengthened temporarily by synchronization   |
| Phase_Seg2                       | [1 ... 8] $t_q$ | May be shortened temporarily by synchronization    |
| Synchronization Jump Width (SJW) | [1 ... 4] $t_q$ | May not be longer than either Phase Buffer Segment |

**NOTE:** For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

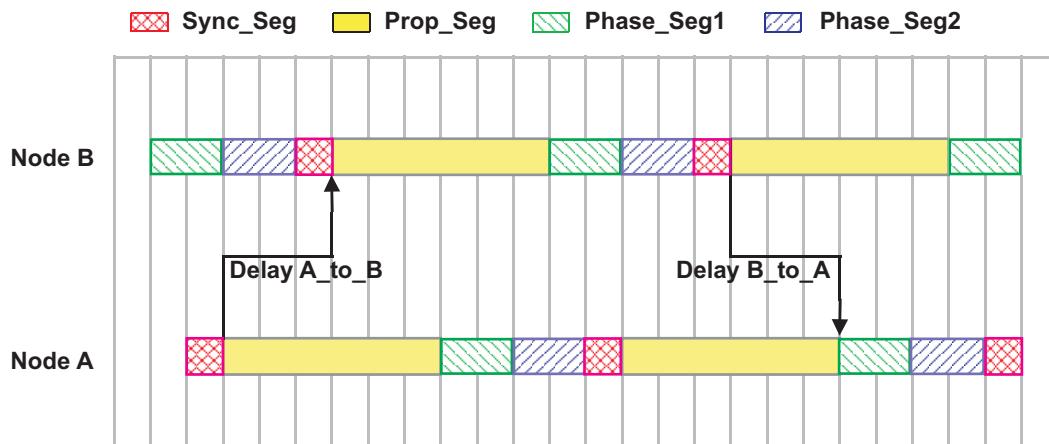
### 23.3.16.1.1 Synchronization Segment

The synchronization segment (Sync\_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync\_Seg, its distance to the Sync\_Seg is called the phase error of this edge.

### 23.3.16.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's nondestructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in [Figure 23-14](#) shows the phase shift and propagation times between two CAN nodes.

**Figure 23-14. The Propagation Time Segment**

$$\text{Delay A\_to\_B} \geq \text{node output delay(A)} + \text{bus line delay(A}\rightarrow\text{B)} + \text{node input delay(B)}$$

$$\text{Prop\_Seg} \geq \text{Delay A\_to\_B} + \text{Delay B\_to\_A}$$

$$\text{Prop\_Seg} \geq 2 \cdot [\max(\text{node output delay} + \text{bus line delay} + \text{node input delay})]$$

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its start of frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A\_to\_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B\_to\_A).

Due to oscillator tolerances, the actual position of node A's sample point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase\_Seg1. This condition defines the length of Prop\_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase\_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the bit timing configuration (Prop\_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The DCAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of  $1 t_q$ , requiring a longer Prop\_Seg.

### **23.3.16.1.3 Phase Buffer Segments and Synchronization**

The phase buffer segments (Phase\_Seg1 and Phase\_Seg2) and the synchronization jump width (SJW) are used to compensate for the oscillator tolerance.

The phase buffer segments surround the sample point and may be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the phase buffer segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync\_Seg; otherwise, its distance to the Sync\_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync\_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame, only resynchronization is possible.

- Hard Synchronization

After a hard synchronization, the bit time is restarted with the end of Sync\_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization, to lie within the synchronization segment of the restarted bit time.

- Bit Resynchronizations

Resynchronization leads to a shortening or lengthening of the Bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes resynchronization is positive, Phase\_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase\_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge which causes Resynchronization is negative, Phase\_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase\_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

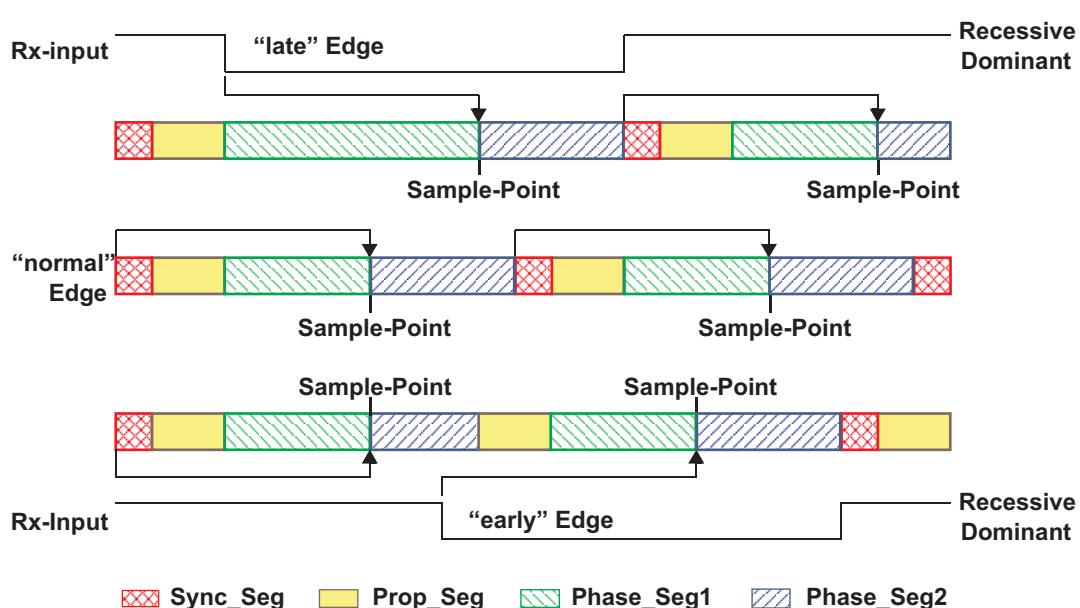
If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop\_Seg + Phase\_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize "hard" on the edge transmitted by the "leading" transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The leading transmitter does not necessarily win the arbitration; therefore, the receivers have to synchronize themselves to different transmitters that subsequently take the lead and that are differently synchronized to the previously leading transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that takes the lead in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

**Figure 23-15. Synchronization on Late and Early Edges**



In the first example, an edge from recessive to dominant occurs at the end of Prop\_Seg. The edge is "late" since it occurs after the Sync\_Seg. Reacting to the late edge, Phase\_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync\_Seg to the sample point if no edge had occurred. The phase error of this late edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync\_Seg.

In the second example, an edge from recessive to dominant occurs during Phase\_Seg2. The edge is "early" since it occurs before a Sync\_Seg. Reacting to the early edge, Phase\_Seg2 is shortened and Sync\_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync\_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this early edge's phase error is less than SJW, so it is fully compensated.

The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

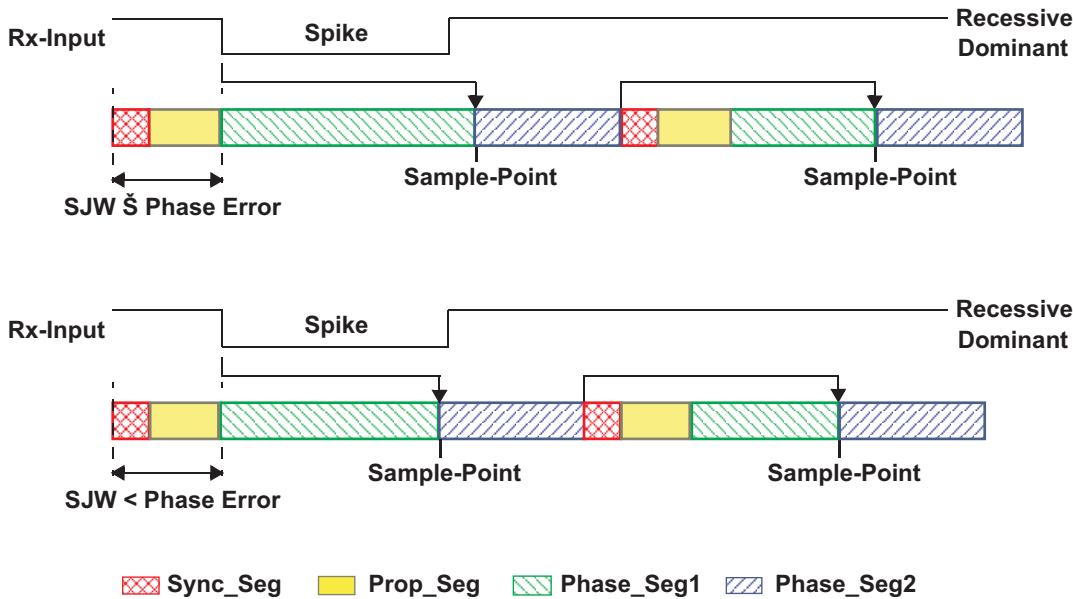
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync\_Seg when synchronizing on an early edge because it cannot subsequently redefine that time quantum of Phase\_Seg2 where the edge occurs to be the Sync\_Seg.

[Figure 23-16](#) shows how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop\_Seg and has the length of (Prop\_Seg + Phase\_Seg1).

In the first example, the synchronization jump width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

**Figure 23-16. Filtering of Short Dominant Spikes**



#### 23.3.16.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range  $df$  for an oscillator's frequency  $f_{osc}$  around the nominal frequency  $f_{nom}$  with:

$$(1 - df) \cdot f_{nom} \leq f_{osc} \leq (1 + df) \cdot f_{nom}$$

depends on the proportions of Phase\_Seg1, Phase\_Seg2, SJW, and the bit time. The maximum tolerance  $df$  is defined by two conditions (both shall be met):

$$\begin{aligned} \text{I: } df &\leq \frac{\min(T\text{Seg1}, T\text{Seg2})}{2x(13x(\text{bit\_time} - T\text{Seg2}))} \\ \text{II: } df &\leq \frac{\text{SJW}}{20x\text{bit\_time}} \end{aligned}$$

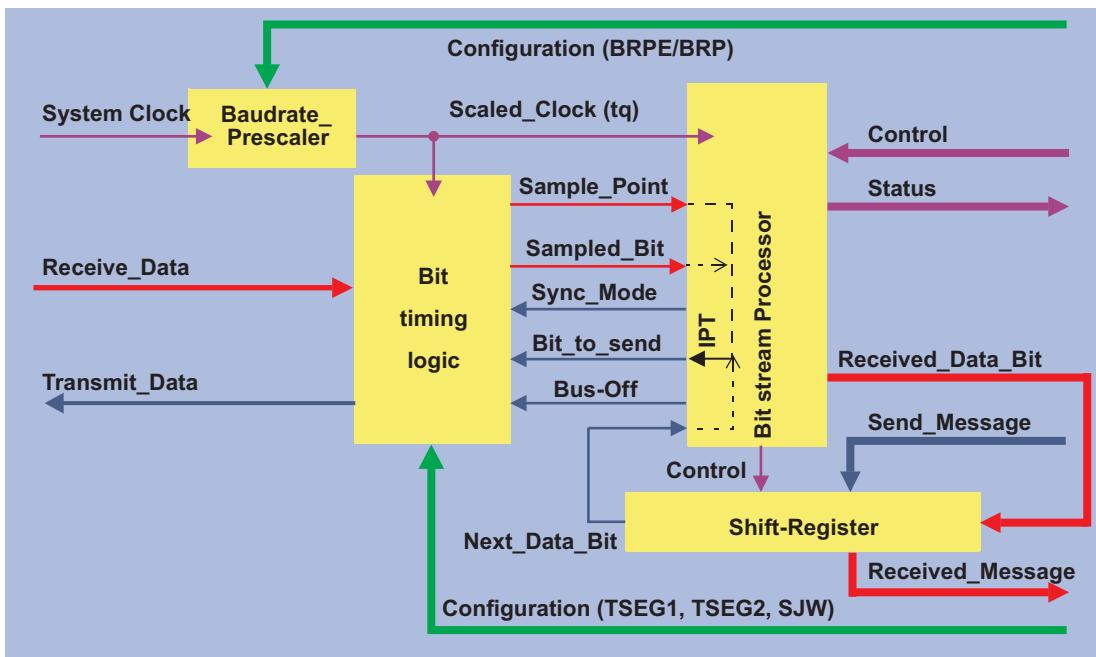
It has to be considered that SJW may not be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that may be used for the phase buffer segments.

The combination Prop\_Seg = 1 and Phase\_Seg1 = Phase\_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a propagation time segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8  $\mu$ s) with a bus length of 40 m.

### 23.3.16.2 DCAN Bit Timing Registers

In the DCAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of four bits (BRPE) is provided. The sum of Prop\_Seg and Phase\_Seg1 (as TSEG1) is combined with Phase\_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 23-17).

**Figure 23-17. Structure of the CAN Core's CAN Protocol Controller**



In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1...n], values in the range of [0...-1] are programmed. That way, e.g., SJW (functional range of [1...4]) is represented by only two bits.

Therefore, the length of the bit time is (programmed values)  $[TSEG1 + TSEG2 + 3] t_q$  or (functional values)  $[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] t_q$ .

The data in the bit timing register (BTR) is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the bit stream processor (BSP) state machine, is evaluated once each bit time, at the sample point.

The shift register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP. The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (e.g., data bit, CRC bit, stuff bit, error flag, or idle) is called the information processing time (IPT), which is 0  $t_q$  for the DCAN.

Generally, the IPT is CAN controller-specific, but may not be longer than  $2 t_q$ . The IPC length is the lower limit of the programmed length of Phase\_Seg2. In case of a synchronization, Phase\_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

### **23.3.16.2.1 Calculation of the Bit Timing Parameters**

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time ( $1 / \text{Bit rate}$ ) must be an integer multiple of the CAN clock period.

---

**NOTE:** 8 MHz is the minimum CAN clock frequency required to operate the DCAN at a bit rate of 1 MBit/s.

---

The bit time may consist of 8 to 25 time quanta. The length of the time quantum  $t_q$  is defined by the baud rate prescaler with  $t_q = (\text{Baud Rate Prescaler}) / \text{CAN\_CLK}$ . Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop\_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop\_Seg is converted into time quanta (rounded up to the nearest integer multiple of  $t_q$ ).

The Sync\_Seg is  $1 t_q$  long (fixed), leaving  $(\text{bit time} - \text{Prop\_Seg} - 1) t_q$  for the two Phase Buffer Segments. If the number of remaining  $t_q$  is even, the Phase Buffer Segments have the same length, Phase\_Seg2 = Phase\_Seg1, else Phase\_Seg2 = Phase\_Seg1 + 1.

The minimum nominal length of Phase\_Seg2 has to be regarded as well. Phase\_Seg2 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of  $[0...2] t_q$ .

The length of the synchronization jump width is set to its maximum value, which is the minimum of four (4) and Phase\_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in [Section 23.3.16.2.3](#).

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the bit timing register:

Tseg2 = Phase\_Seg2-1

Tseg1 = Phase\_Seg1+ Prop\_Seg-1

SJW = SynchronizationJumpWidth-1

BRP = Prescaler-1

### **23.3.16.2.2 Example for Bit Timing at High Baud Rate**

In this example, the frequency of CAN\_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

|                           |         |   |  |
|---------------------------|---------|---|--|
| $t_q$                     | 100 ns  | = | $t_{CAN\_CLK}$   |
| delay of bus driver       | 60 ns   | = |  |
| delay of receiver circuit | 40 ns   | = |  |
| delay of bus line (40 m)  | 220 ns  | = |  |
| $t_{Prop}$                | 700 ns  | = | $INT(2^{*}delays + 1) = 7 \cdot t_q$                     |
| $t_{SJW}$                 | 100 ns  | = | $1 \cdot t_q$  |
| $t_{TSeg1}$               | 800 ns  | = | $t_{Prop} + t_{SJW}$                                     |
| $t_{TSeg2}$               | 100 ns  | = | Information Processing Time + $1 \cdot t_q$              |
| $t_{Sync-Seg}$            | 100 ns  | = | $1 \cdot t_q$  |
| bit time                  | 1000 ns | = | $t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$                   |
| tolerance for CAN_CLK     | 0.43 %  | = | $\frac{(min(TSeg1, TSeg2))}{2x(13x(bit\_time - TSeg2))}$ |
|                           |         | = | $\frac{0.1\mu s}{2x(13x(1\mu s - 0.1\mu s))}$            |

In this example, the concatenated bit time parameters are (1-1)<sub>3</sub>&(8-1)<sub>4</sub>&(1-1)<sub>2</sub>&(1-1)<sub>6</sub>, so the bit timing register is programmed to = 00000700.

### 23.3.16.2.3 Example for Bit Timing at Low Baud Rate

In this example, the frequency of CAN\_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

|                           |           |   |  |
|---------------------------|-----------|---|--|
| $t_q$                     | 1 $\mu s$ | = | $t_{CAN\_CLK}$   |
| Delay of bus driver       | 200 ns    | = |  |
| Delay of receiver circuit | 80 ns     | = |  |
| Delay of bus line (40 m)  | 220 ns    | = |  |
| $t_{Prop}$                | 1 $\mu s$ | = | $1 \cdot t_q$  |
| $t_{SJW}$                 | 4 $\mu s$ | = | $4 \cdot t_q$  |
| $t_{TSeg1}$               | 5 $\mu s$ | = | $t_{Prop} + t_{SJW}$                                     |
| $t_{TSeg2}$               | 3 $\mu s$ | = | Information Processing Time + $3 \cdot t_q$              |
| $t_{Sync-Seg}$            | 1 $\mu s$ | = | $1 \cdot t_q$  |
| Bit time                  | 9 $\mu s$ | = | $t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$                   |
| Tolerance for CAN_CLK     |           | = | $\frac{(min(TSeg1, TSeg2))}{2x(13x(bit\_time - TSeg2))}$ |
|                           |           | = | $\frac{0.1\mu s}{2x(13x(1\mu s - 0.1\mu s))}$            |

In this example, the concatenated bit time parameters are (3-1)<sub>3</sub>&(5-1)<sub>4</sub>&(4-1)<sub>2</sub>&(2-1)<sub>6</sub>, so the bit timing register is programmed to = 0x000024C1.

### 23.3.17 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the message RAM. There are two interface registers sets for read/write access, IF1 and IF2 and one interface register set for read access only, IF3.

Due to the structure of the message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the message RAM is accessed. Therefore the data transfer from the IF1/IF2 registers to the message RAM requires the message handler to perform a read-modify-write cycle: First those parts of the message object that are not to be changed are read from the message RAM into the interface register set, and after the update the whole content of the interface register set is written into the message object.

After the partial write of a message object, those parts of the interface register set that are not selected in the command register, will be set to the actual contents of the selected message object.

After the partial read of a message object, those parts of the interface register set that are not selected in the command register, will be left unchanged.

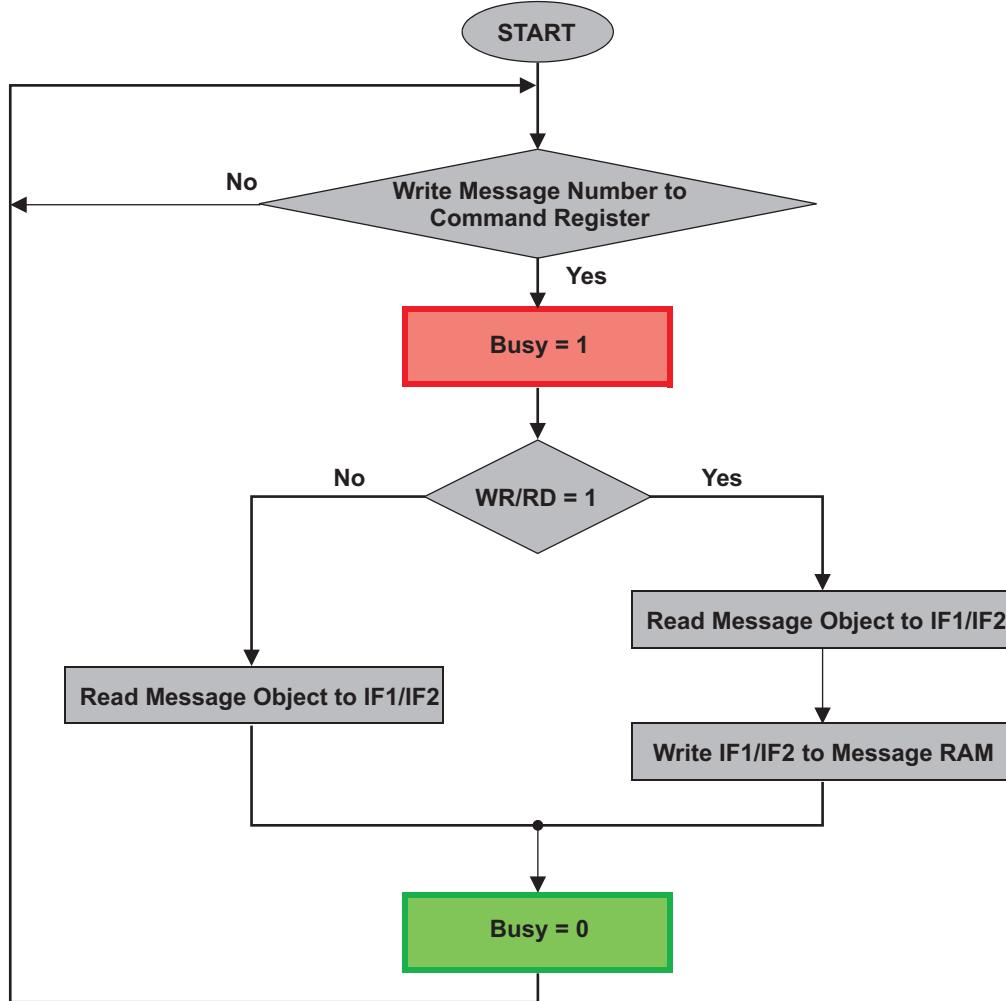
By buffering the data to be transferred, the interface register sets avoid conflicts between concurrent CPU accesses to the message RAM and CAN message reception and transmission. A complete message object (see [Section 23.3.18.1](#)) or parts of the message object may be transferred between the message RAM and the IF1/IF2 register set (see [Section 23.4.1.19](#)) in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

#### 23.3.17.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 register sets control the data transfer to and from the message object. The command register addresses the desired message object in the message RAM and specifies whether a complete message object or only parts should be transferred. The data transfer is initiated by writing the message number to the bits [7:0] of the command register.

When the CPU initiates a data transfer between the IF1/IF2 registers and message RAM, the message handler sets the busy bit in the respective command register to '1'. After the transfer has completed, the busy bit is set back to '0' (see [Figure 23-18](#)).

**Figure 23-18. Data Transfer Between IF1/IF2 Registers and Message RAM**



### 23.3.17.2 IF3 Register Set

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from message RAM by CPU. The intention of this feature of IF3 is to provide an interface for the DMA to read packets efficiently. The automatic update functionality can be programmed for each message object (Section 23.4.1.28).

All valid message objects in message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (if no previous DMA transfers are ongoing), controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If DCAN internal IF3 update is complete, a DMA request is generated. The DMA request stays active until first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in CAN control register. Please refer to the device datasheet to find out if this DMA source is available.

---

**NOTE:** The IF3 register set can not be used for transferring data into message objects.

---

### 23.3.18 Message RAM

The DCAN message RAM contains message objects and parity bits for the message objects. There are up to 64 message objects in the message RAM.

During normal operation, accesses to the message RAM are performed via the interface register sets, and the CPU cannot directly access the message RAM.

The interface register sets IF1 and IF2 provide indirect read/write access from the CPU to the message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third interface register set IF3 can be configured to automatically receive control and user data from the message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from message RAM to IF3 register set.

The message handler avoids potential conflicts between concurrent accesses to message RAM and CAN frame reception/transmission.

There are two modes where the message RAM can be directly accessed by the CPU:

- Debug/Suspend mode (see [Section 23.3.18.3](#))
- RAM Direct Access (RDA) mode (see [Section 23.3.18.4](#))

### 23.3.18.1 Structure of Message Objects

[Table 23-8](#) shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

**Table 23-8. Structure of a Message Object**

| Message Object |           |      |      |          |        |        |        |        |        |        |        |        |
|----------------|-----------|------|------|----------|--------|--------|--------|--------|--------|--------|--------|--------|
| UMask          | Msk[28:0] | MXtd | MDir | EoB      | unused | NewDat | MsgLst | RxE    | TxE    | IntPnd | RmtEn  | TxRqst |
| MsgVal         | ID[28:0]  | Xtd  | Dir  | DLC[3:0] | Data 0 | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 |

**Table 23-9. Field Descriptions**

| Name      | Value     | Description   |
|-----------|-----------|---|
| MsgVal    | 0         | Message valid   |
|           | 1         | The message object is ignored by the message handler.   |
|           |           | The message object is to be used by the message handler.  |
|           |           | Note: This bit may be kept at level '1' even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the data length code are changed. It should be reset if the Messages Object is no longer required.                                |
| UMask     | 0         | Use acceptance mask   |
|           | 1         | Mask bits (Msk[28:0], MXtd and MDir) are ignored and not used for acceptance filtering.   |
|           |           | Mask bits are used for acceptance filtering.  |
|           |           | Note: If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one.   |
| ID[28:0]  | ID[28:0]  | Message identifier  |
|           | ID[28:18] | 29-bit ("extended") identifier bits   |
|           |           | 11-bit ("standard") identifier bits   |
| Msk[28:0] | 0         | Identifier mask   |
|           | 1         | The corresponding bit in the message identifier is not used for acceptance filtering (don't care).  |
|           |           | The corresponding bit in the message identifier is used for acceptance filtering.   |
| Xtd       | 0         | Mask extended identifier  |
|           | 1         | The extended identifier bit (IDE) has no effect on the acceptance filtering.  |
|           |           | The extended identifier bit (IDE) is used for acceptance filtering.   |
|           |           | Note: When 11-bit ("standard") Identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. |
| Dir       | 0         | Message direction   |
|           | 1         | Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message is stored in this message object.                                  |
|           |           | Direction = transmit: On TxRqst, a data frame is transmitted. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = one).   |
| MDir      | 0         | Mask message direction  |
|           | 1         | The message direction bit (Dir) has no effect on the acceptance filtering.  |
|           |           | The message direction bit (Dir) is used for acceptance filtering.   |
| EOB       | 0         | End of block  |
|           | 1         | The message object is part of a FIFO Buffer block and is not the last message object of this FIFO Buffer block.   |
|           |           | The message object is a single message object or the last message object in a FIFO Buffer Block.  |
|           |           | Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one.   |

**Table 23-9. Field Descriptions (continued)**

| Name   | Value | Description   |
|--|-------|---|
| NewDat   | 0     | New data<br>No new data has been written into the data bytes of this message object by the message handler since the last time when this flag was cleared by the CPU.   |
|  | 1     | The message handler or the CPU has written new data into the data bytes of this message object.   |
| MsgLst   | 0     | Message lost (only valid for message objects with direction = receive)<br>No message was lost since the last time when this bit was reset by the CPU.   |
|  | 1     | The message handler stored a new message into this message object when NewDat was still set, so the previous message has been overwritten.  |
| RxIE   |       | Receive interrupt enable<br>IntPnd will not be triggered after the successful reception of a frame.   |
|  | 1     | IntPnd will be triggered after the successful reception of a frame.   |
| TxIE   |       | Transmit interrupt enable<br>IntPnd will not be triggered after the successful transmission of a frame.   |
|  | 1     | IntPnd will be triggered after the successful transmission of a frame.  |
| IntPnd   | 0     | Interrupt pending<br>This message object is not the source of an interrupt.   |
|  | 1     | This message object is the source of an interrupt. The interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.   |
| RmtEn  |       | Remote enable<br>At the reception of a remote frame, TxRqst is not changed.   |
|  | 1     | At the reception of a remote frame, TxRqst is set.  |
| TxRqst   |       | Transmit request<br>This message object is not waiting for a transmission.  |
|  | 1     | The transmission of this message object is requested and is not yet done.   |
| DLC[3:0]   |       | Data length code<br>Data frame has 0-8 data bits.<br>Data frame has 8 data bytes.<br>Note: The data length code of a message object must be defined to the same value as in the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.  |
| Data 0<br>Data 1<br>Data 2<br>Data 3<br>Data 4<br>Data 5<br>Data 6<br>Data 7 |       | 1st data byte of a CAN data frame<br>2nd data byte of a CAN data frame<br>3rd data byte of a CAN data frame<br>4th data byte of a CAN data frame<br>5th data byte of a CAN data frame<br>6th data byte of a CAN data frame<br>7th data byte of a CAN data frame<br>8th data byte of a CAN data frame<br>Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN core during a reception, byte Data 7 is the last. When the message handler stores a data frame, it will write all the eight data bytes into a message object. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by undefined values. |

### 23.3.18.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:  
 Message RAM base address + (message object number) \* 0x20

This means that message object 1 starts at offset 0x0020; message object 2 starts at offset 0x0040, etc.

'0' is not a valid message object number.

The base address for DCAN0 RAM is 0x481C\_D000 and DCAN1 RAM is 0x481D\_1000.

Message object number 1 has the highest priority.

**Table 23-10. Message RAM addressing in Debug/Suspend and RDA Mode**

| Message Object Number              | Offset From Base Address | Word Number | Debug/Suspend Mode, see <a href="#">Section 23.3.18.3</a> | RDA mode, see <a href="#">Section 23.3.18.4</a> |
|------------------------------------|--------------------------|-------------|---|---|
| 1                                  | 0x0020                   | 1           | Parity  | Data Bytes 4-7                                  |
|                                    | 0x0024                   | 2           | MXtd,MDir,Mask  | Data Bytes 0-3                                  |
|                                    | 0x0028                   | 3           | Xtd,Dir,ID  | ID[27:0],DLC                                    |
|                                    | 0x002C                   | 4           | Ctrl  | Mask,Xtd,Dir,ID[28]                             |
|                                    | 0x0030                   | 5           | Data Bytes 3-0  | Parity,Ctrl,MXtd,MDir                           |
|                                    | 0x0034                   | 6           | Data Bytes 7-4  | ...   |
| ...                                | ...                      | ...         | ...   | ...   |
| 31                                 | 0x03E0                   | 1           | Parity  | Data Bytes 4-7                                  |
|                                    | 0x03E4                   | 2           | MXtd,MDir,Mask  | Data Bytes 0-3                                  |
|                                    | 0x03E8                   | 3           | Xtd,Dir,ID  | ID[27:0],DLC                                    |
|                                    | 0x03EC                   | 4           | Ctrl  | Mask,Xtd,Dir,ID[28]                             |
|                                    | 0x03F0                   | 5           | Data Bytes 3-0  | Parity,Ctrl,MXtd,MDir                           |
|                                    | 0x03F4                   | 6           | Data Bytes 7-4  | ...   |
| ...                                | ...                      | ...         | ...   | ...   |
| 63                                 | 0x07E0                   | 1           | Parity  | Data Bytes 4-7                                  |
|                                    | 0x07E4                   | 2           | MXtd,MDir,Mask  | Data Bytes 0-3                                  |
|                                    | 0x07E8                   | 3           | Xtd,Dir,ID  | ID[27:0],DLC                                    |
|                                    | 0x07EC                   | 4           | Ctrl  | Mask,Xtd,Dir,ID[28]                             |
|                                    | 0x07F0                   | 5           | Data Bytes 3-0  | Parity,Ctrl,MXtd,MDir                           |
|                                    | 0x07F4                   | 6           | Data Bytes 7-4  | ...   |
| last implemented (32(DCAN3) or 64) | 0x0000                   | 1           | Parity  | Data Bytes 4-7                                  |
|                                    | 0x0004                   | 2           | MXtd,MDir,Mask  | Data Bytes 0-3                                  |
|                                    | 0x0008                   | 3           | Xtd,Dir,ID  | ID[27:0],DLC                                    |
|                                    | 0x000C                   | 4           | Ctrl  | Mask,Xtd,Dir,ID[28]                             |
|                                    | 0x0010                   | 5           | Data Bytes 3-0  | Parity,Ctrl,MXtd,MDir                           |
|                                    | 0x0014                   | 6           | Data Bytes 7-4  | ...   |

### 23.3.18.3 Message RAM Representation in Debug/Suspend Mode

In debug/suspend mode, the message RAM will be memory mapped. This allows the external debug unit to access the message RAM.

**NOTE:** During debug/suspend mode, the message RAM cannot be accessed via the IFx register sets.

**Table 23-11. Message RAM Representation in Debug/Suspend Mode**

| Bit #          | 31/<br>15                     | 30/<br>14  | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9  | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5    | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0 |  |  |  |  |  |  |  |
|----------------|-------------------------------|------------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|-------------|----------|----------|----------|----------|----------|--|--|--|--|--|--|--|
|                | Reserved                      |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x00 | Reserved                      |            |           |           |           |           |           |          |          |          | Parity[4:0] |          |          |          |          |          |  |  |  |  |  |  |  |
|                | Mxt<br>d MDir Rsvd Msk[28:16] |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x04 | Msk[15:0]                     |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
|                | Rsv<br>d Xtd Dir ID[28:16]    |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x08 | ID[15:0]                      |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
|                | Reserved                      |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x0C | Rsv<br>d                      | Msg<br>Lst | Rsvd      | UMas<br>k | TxIE      | RxTE      | RmtE<br>n | Rsv<br>d | EOB      | Reserved |             |          | DLC[3:0] |          |          |          |  |  |  |  |  |  |  |
|                | Data 3                        |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x10 | Data 1                        |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
|                | Data 7                        |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |
| MsgAddr + 0x14 | Data 5                        |            |           |           |           |           |           |          |          |          |             |          |          |          |          |          |  |  |  |  |  |  |  |

### 23.3.18.4 Message RAM Representation in Direct Access Mode

When the **RDA** bit in the test register is set while the DCAN module is in test mode (test bit in the CAN control register is set), the CPU has direct access to the message RAM. Due to the 32-bit bus structure, the RAM is split into word lines to support this feature. The CPU has access to one word line at a time only.

In RAM direct access mode, the RAM is represented by a continuous memory space within the address frame of the DCAN module, starting at the message RAM base address.

Note: During direct access mode, the message RAM cannot be accessed via the IFx register sets.

Any read or write to the RAM addresses for RamDirectAccess during normal operation mode (TestMode bit or RDA bit not set) will be ignored.

**Table 23-12. Message RAM Representation in RAM Direct Access Mode**

| Bit #          | 31/<br>15  | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0 |  |  |  |  |  |  |
|----------------|------------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|--|--|--|--|--|--|
|                | Data 4     |           |           |           |           |           |          |          |          |          |          |          |          |          |          |          |  |  |  |  |  |  |
| MsgAddr + 0x00 | Data 6     |           |           |           |           |           |          |          |          |          | Data 7   |          |          |          |          |          |  |  |  |  |  |  |
|                | Data 0     |           |           |           |           |           |          |          |          |          |          |          |          |          |          |          |  |  |  |  |  |  |
| MsgAddr + 0x04 | Data 2     |           |           |           |           |           |          |          |          |          |          |          |          |          |          |          |  |  |  |  |  |  |
|                | ID[27:12]  |           |           |           |           |           |          |          |          |          |          |          |          |          |          |          |  |  |  |  |  |  |
| MsgAddr + 0x08 | ID[11:0]   |           |           |           |           |           |          |          |          |          | DLC[3:0] |          |          |          |          |          |  |  |  |  |  |  |
|                | Msk[28:13] |           |           |           |           |           |          |          |          |          |          |          |          |          |          |          |  |  |  |  |  |  |

**Table 23-12. Message RAM Representation in RAM Direct Access Mode (continued)**

| Bit #          | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8   | 23/<br>7  | 22/<br>6  | 21/<br>5 | 20/<br>4  | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0      |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|------------|-----------|-----------|----------|-----------|----------|----------|----------|---------------|
| MsgAddr + 0x0C | Msk[12:0] |           |           |           |           |           |          |            |           |           | Xtd      | Dir       | ID[28]   |          |          |               |
| MsgAddr + 0x10 | Reserved  |           |           |           |           |           |          |            |           |           |          |           |          |          |          | Parity<br>[4] |
|                | MsgLst    |           |           | Unused    |           |           |          | Msg<br>Lst | UMa<br>sk | TxE<br>IE | RxT<br>E | Rmt<br>En | EOB      | MX<br>td | MDir     |               |

---

**NOTE:** Writes to unused bits have no effect.

---

### 23.3.19 GIO Support

The CAN\_RX and CAN\_TX pins of the DCAN module can be used as general purpose IO pins, if CAN functionality is not needed. This function is controlled by the CAN TX IO control register (DCAN TIOC) (see [Section 23.4.1.29](#)) and the CAN RX IO control register (DCAN RIOC) (see [Section 23.4.1.30](#)).

## 23.4 DCAN Registers

### 23.4.1 DCAN Control Registers

After hardware reset, the registers of the DCAN hold the values shown in the register descriptions.

Additionally, the Bus-Off state is reset and the CAN\_TX pin is set to recessive (HIGH). The Init bit in the CAN control register is set to enable the software initialization. The DCAN will not influence the CAN bus until the CPU resets Init to '0'.

**Table 23-13. DCAN Control Register Summary Table**

| Offset         | Acronym                         | Description                     | Section                           |
|----------------|---------------------------------|---------------------------------|-----------------------------------|
| 0x00           | DCAN CTL                        | CAN Control Register            | <a href="#">Section 23.4.1.1</a>  |
| 0x04           | DCAN ES                         | Error and Status Register       | <a href="#">Section 23.4.1.2</a>  |
| 0x08           | DCAN ERRC                       | Error Counter Register          | <a href="#">Section 23.4.1.3</a>  |
| 0x0C           | DCAN BTR                        | Bit Timing Register             | <a href="#">Section 23.4.1.4</a>  |
| 0x10           | DCAN INT                        | Interrupt Register              | <a href="#">Section 23.4.1.5</a>  |
| 0x14           | DCAN TEST                       | Test Register                   | <a href="#">Section 23.4.1.6</a>  |
| 0x1C           | DCAN PERR                       | Parity Error Code Register      | <a href="#">Section 23.4.1.7</a>  |
| 0x80           | DCAN ABOTR                      | Auto-Bus-On Time Register       | <a href="#">Section 23.4.1.8</a>  |
| 0x84           | DCAN TXRQ X                     | Transmission Request X Register | <a href="#">Section 23.4.1.9</a>  |
| 0x88 to 0x94   | DCAN TXRQ12 to DCAN TXRQ78      | Transmission Request Registers  | <a href="#">Section 23.4.1.10</a> |
| 0x98           | DCAN NWDAT X                    | New Data X Register             | <a href="#">Section 23.4.1.11</a> |
| 0x9C to 0xA8   | DCAN NWDAT12 to DCAN NWDAT78    | New Data Registers              | <a href="#">Section 23.4.1.12</a> |
| 0xAC           | DCAN INTPND X                   | Interrupt Pending X Register    | <a href="#">Section 23.4.1.13</a> |
| 0xB0 to 0xBC   | DCAN INTPND12 to DCAN INTPND78) | Interrupt Pending Registers     | <a href="#">Section 23.4.1.14</a> |
| 0xC0           | DCAN MSGVAL X                   | Message Valid X Register        | <a href="#">Section 23.4.1.15</a> |
| 0xC4 to 0xD0   | DCAN MSGVAL12 to DCAN MSGVAL78  | Message Valid Registers         | <a href="#">Section 23.4.1.16</a> |
| 0xD8 to 0xE4   | DCAN INTMUX12 to DCAN INTMUX78  | Interrupt Multiplexer Registers | <a href="#">Section 23.4.1.17</a> |
| 0x100          | DCAN IF1CMD                     | IF1 Command Registers           | <a href="#">Section 23.4.1.18</a> |
| 0x120          | DCAN IF2CMD                     | IF2 Command Registers           | <a href="#">Section 23.4.1.18</a> |
| 0x104          | DCAN IF1MSK                     | IF1 Mask Register               | <a href="#">Section 23.4.1.19</a> |
| 0x124          | DCAN IF2MSK                     | IF2 Mask Register               | <a href="#">Section 23.4.1.19</a> |
| 0x108          | DCAN IF1ARB                     | IF1 Arbitration Register        | <a href="#">Section 23.4.1.20</a> |
| 0x128          | DCAN IF2ARB                     | IF2 Arbitration Register        | <a href="#">Section 23.4.1.20</a> |
| 0x10C          | DCAN IF1MCTL                    | IF1 Message Control Register    | <a href="#">Section 23.4.1.21</a> |
| 0x12C          | DCAN IF2MCTL                    | IF2 Message Control Register    | <a href="#">Section 23.4.1.21</a> |
| 0x110          | DCAN IF1DATA                    | IF1 Data A Register             | <a href="#">Section 23.4.1.22</a> |
| 0x114          | DCAN IF1DATB                    | IF1 Data B Register             | <a href="#">Section 23.4.1.22</a> |
| 0x130          | DCAN IF2DATA                    | IF2 Data A Register             | <a href="#">Section 23.4.1.22</a> |
| 0x134          | DCAN IF2DATB                    | IF2 Data B Register             | <a href="#">Section 23.4.1.22</a> |
| 0x140          | DCAN IF3OBS                     | IF3 Observation Register        | <a href="#">Section 23.4.1.23</a> |
| 0x144          | DCAN IF3MSK                     | IF3 Mask Register               | <a href="#">Section 23.4.1.24</a> |
| 0x148          | DCAN IF3ARB                     | IF3 Arbitration Register        | <a href="#">Section 23.4.1.25</a> |
| 0x14C          | DCAN IF3MCTL                    | IF3 Message Control Register    | <a href="#">Section 23.4.1.26</a> |
| 0x150          | DCAN IF3DATA                    | IF3 Data A Register             | <a href="#">Section 23.4.1.27</a> |
| 0x154          | DCAN IF3DATB                    | IF3 Data B Register             | <a href="#">Section 23.4.1.27</a> |
| 0x160 to 0x16C | DCAN IF3UPD12 to IF3UPD78       | IF3 Update Enable Registers     | <a href="#">Section 23.4.1.28</a> |

**Table 23-13. DCAN Control Register Summary Table (continued)**

| Offset | Acronym   | Description                | Section                           |
|--------|-----------|----------------------------|-----------------------------------|
| 0x1E0  | DCAN TIOC | CAN TX IO Control Register | <a href="#">Section 23.4.1.29</a> |
| 0x1E4  | DCAN RIOC | CAN RX IO Control Register | <a href="#">Section 23.4.1.30</a> |

**23.4.1.1 CAN Control Register (DCAN CTL)**

The CAN control register (DCAN CTL) is shown in [Figure 23-19](#) and described in [Table 23-14](#).

**Figure 23-19. CAN Control Register (DCAN CTL)**

| 31     | 26        | 25      | 24    | 23       | 21    | 20    | 19    | 18        | 17      | 16    |       |       |
|--------|-----------|---------|-------|----------|-------|-------|-------|-----------|---------|-------|-------|-------|
|        | Reserved  | WUBA    | PDR   | Reserved | DE3   | DE2   | DE1   | IE1       | InitDbg |       |       |       |
|        | R-0       | R/W-0   | R/W-0 | R-0      | R/W-0 | R/W-0 | R/W-0 | R/W-0     | R/W-0   | R-0   |       |       |
| 15     | 14        | 13      | 10    | 9        | 8     | 7     | 6     | 5         | 4       | 3     |       |       |
| SWR    | Reserv ed | PMD     | ABO   | IDS      | Test  | CCE   | DAR   | Reserv ed | EIE     | SIE   | IE0   | Init  |
| R/WP-0 | R-0       | R/W-0x5 | R/W-0 | R/W-0    | R/W-0 | R/W-0 | R/W-0 | R-0       | R/W-0   | R/W-0 | R/W-0 | R/W-1 |

LEGEND: R/W = Read/Write; R = Read only; WP = Write protected by init bit; -n = value after reset

**Table 23-14. CAN Control Register (DCAN CTL) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-26 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 26    | WUBA     | 0     | Automatic wake up on bus activity when in local power-down mode  |
|       |          | 1     | No detection of a dominant CAN bus level while in local power-down mode.   |
|       |          | 1     | Detection of a dominant CAN bus level while in local power-down mode is enabled. On occurrence of a dominant CAN bus level, the wake up sequence is started (Additional information can be found in <i>Local Power-Down Mode</i> ).<br><br>Note: The CAN message, which initiates the bus activity, cannot be received. This means that the first message received in power down and automatic wake-up mode, will be lost. |
| 24    | PDR      | 0     | Request for local low power-down mode  |
|       |          | 0     | No application request for local low power-down mode. If the application has cleared this bit while DCAN in local power-down mode, also the Init bit has to be cleared.  |
|       |          | 1     | Local power-down mode has been requested by application. The DCAN will acknowledge the local power-down mode by setting bit PDA in the error and status register. The local clocks will be turned off by DCAN internal logic (Additional information can be found in <i>Local Power-Down Mode</i> ).   |
| 23-21 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 20    | DE3      | 0     | Enable DMA request line for IF3  |
|       |          | 0     | Disabled   |
|       |          | 1     | Enabled  |
|       |          |       | Note: A pending DMA request for IF3 remains active until first access to one of the IF3 registers.   |
| 19    | DE2      | 0     | Enable DMA request line for IF2  |
|       |          | 0     | Disabled   |
|       |          | 1     | Enabled  |
|       |          |       | Note: A pending DMA request for IF2 remains active until first access to one of the IF2 registers.   |
| 18    | DE1      | 0     | Enable DMA request line for IF1  |
|       |          | 0     | Disabled   |
|       |          | 1     | Enabled  |
|       |          |       | Note: A pending DMA request for IF1 remains active until first access to one of the IF1 registers.   |

**Table 23-14. CAN Control Register (DCAN CTL) Field Descriptions (continued)**

| Bit   | Field    | Value         | Description   |
|-------|----------|---------------|---|
| 17    | IE1      | 0<br>1        | Interrupt line 1 enable<br>Disabled - Module interrupt DCAN1INT is always low.<br>Enabled - interrupts will assert line DCAN1INT to one; line remains active until pending interrupts are processed.  |
| 16    | InitDbg  | 0<br>1        | Internal init state while debug access<br>Not in debug mode, or debug mode requested but not entered.<br>Debug mode requested and internally entered; the DCAN is ready for debug accesses.   |
| 15    | SWR      | 0<br>1        | SW reset enable<br>Normal Operation<br>Module is forced to reset state. This bit will automatically get cleared after execution of SW reset after one OCP clock cycle.<br>Note: To execute software reset, the following procedure is necessary:<br>1. Set Init bit to shut down CAN communication.<br>2. Set SWR bit additionally to Init bit. |
| 14    | Reserved | 0             | This bit is always read as 0. Writes have no effect.  |
| 13-10 | PMD      | 0x5<br>Others | Parity on/off<br>Parity function disabled<br>Parity function enabled  |
| 9     | ABO      | 0<br>1        | Auto-Bus-On enable<br>The Auto-Bus-On feature is disabled<br>The Auto-Bus-On feature is enabled   |
| 8     | IDS      | 0<br>1        | Interruption debug support enable<br>When Debug/Suspend mode is requested, DCAN will wait for a started transmission or reception to be completed before entering Debug/Suspend mode<br>When Debug/Suspend mode is requested, DCAN will interrupt any transmission or reception, and enter Debug/Suspend mode immediately.                      |
| 7     | Test     | 0<br>1        | Test mode enable<br>Normal Operation<br>Test Mode   |
| 6     | CCE      | 0<br>1        | Configuration change enable<br>The CPU has no write access to the configuration registers.<br>The CPU has write access to the configuration registers (when Init bit is set).   |
| 5     | DAR      | 0<br>1        | Disable automatic retransmission<br>Automatic retransmission of not successful messages enabled.<br>Automatic retransmission disabled.  |
| 4     | Reserved | 0             | This bit is always read as 0. Writes have no effect.  |
| 3     | EIE      | 0<br>1        | Error interrupt enable<br>Disabled - PER, BOff and EWarn bits can not generate an interrupt.<br>Enabled - PER, BOff and EWarn bits can generate an interrupt at DCAN0INT line and affect the interrupt register.  |
| 2     | SIE      | 0<br>1        | Status change interrupt enable<br>Disabled - WakeUpPnd, RxOk, TxOk and LEC bits can not generate an interrupt.<br>Enabled - WakeUpPnd, RxOk, TxOk and LEC can generate an interrupt at DCAN0INT line and affect the interrupt register.   |
| 1     | IE0      | 0<br>1        | Interrupt line 0 enable<br>Disabled - Module interrupt DCAN0INT is always low.<br>Enabled - interrupts will assert line DCAN0INT to one; line remains active until pending interrupts are processed.  |

**Table 23-14. CAN Control Register (DCAN CTL) Field Descriptions (continued)**

| Bit | Field | Value | Description                        |
|-----|-------|-------|------------------------------------|
| 0   | Init  | 0     | Initialization<br>Normal operation |
|     |       | 1     | Initialization mode is entered     |

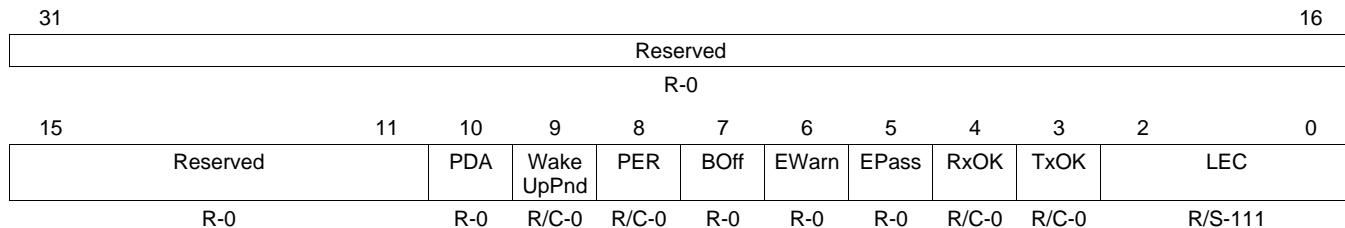
**NOTE:** The Bus-Off recovery sequence (refer to CAN specification) cannot be shortened by setting or resetting Init bit. If the module goes Bus-Off, it will automatically set the Init bit and stop all bus activities.

When the Init bit is cleared by the application again, the module will then wait for 129 occurrences of Bus Idle (129 \* 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error counters will be reset.

After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 error code is written to the error and status register, enabling the CPU to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

#### 23.4.1.2 Error and Status Register (DCAN ES)

The error and status register (DCAN ES) is shown in [Figure 23-20](#) and described in [Table 23-15](#).

**Figure 23-20. Error and Status Register (DCAN ES)**

LEGEND: R = Read only; S = Set by Read; C = Clear by Read; -n = value after reset

**Table 23-15. Error and Status Register (DCAN ES) Field Descriptions**

| Bit   | Field      | Value | Description  |
|-------|------------|-------|--|
| 31-11 | Reserved   | 0     | These bits are always read as 0. Writes have no effect.  |
| 10    | PDA        | 0     | Local power-down mode acknowledge  |
|       |            | 1     | DCAN is not in local power-down mode.  |
|       |            | 0     | Application request for setting DCAN to local power-down mode was successful. DCAN is in local power-down mode.  |
| 9     | WakeUp Pnd | 0     | Wake up pending. This bit can be used by the CPU to identify the DCAN as the source to wake up the system.   |
|       |            | 1     | No Wake Up is requested by DCAN.   |
|       |            | 0     | DCAN has initiated a wake up of the system due to dominant CAN bus while module power down. This bit will be reset if error and status register is read. |
| 8     | PER        | 0     | Parity error detected  |
|       |            | 1     | No parity error has been detected since last read access.  |
|       |            | 0     | The parity check mechanism has detected a parity error in the Message RAM.   |
|       |            | 1     | This bit will be reset if error and status register is read.   |
| 7     | BOff       | 0     | Bus-Off state  |
|       |            | 1     | The CAN module is not bus-off state.   |
|       |            | 0     | The CAN module is in bus-off state.  |

**Table 23-15. Error and Status Register (DCAN ES) Field Descriptions (continued)**

| Bit | Field | Value | Description  |
|-----|-------|-------|--|
| 6   | EWarn | 0     | Warning state<br>Both error counters are below the error warning limit of 96.  |
|     |       | 1     | At least one of the error counters has reached the error warning limit of 96.  |
|     |       |       |  |
| 5   | EPass | 0     | Error passive state<br>On CAN Bus error, the DCAN could send active error frames.  |
|     |       | 1     | The CAN core is in the error passive state as defined in the CAN Specification.  |
|     |       |       |  |
| 4   | RxOk  | 0     | Received a message successfully<br>No message has been successfully received since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.   |
|     |       | 1     | A message has been successfully received since the last time when this bit was reset by a read access of the CPU (independent of the result of acceptance filtering).<br>This bit will be reset if error and status register is read.  |
|     |       |       |  |
| 3   | TxOk  | 0     | Transmitted a message successfully<br>No message has been successfully transmitted since the last time when this bit was read by the CPU. This bit is never reset by DCAN internal events.   |
|     |       | 1     | A message has been successfully transmitted (error free and acknowledged by at least one other node) since the last time when this bit was reset by a read access of the CPU.<br>This bit will be reset if error and status register is read.  |
|     |       |       |  |
| 2-0 | LEC   | 0     | Last error code. The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error.   |
|     |       | 1     | No error   |
|     |       | 2     | Stuff error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.  |
|     |       | 3     | Form error: A fixed format part of a received frame has the wrong format.  |
|     |       | 4     | Ack error: The message this CAN core transmitted was not acknowledged by another node.   |
|     |       | 5     | Bit1 error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.  |
|     |       | 6     | Bit0 error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). |
|     |       | 7     | CRC error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).   |

Interrupts are generated by bits PER, BOff and EWarn (if EIE bit in CAN control register is set) and by bits WakeUpPnd, RxOk, TxOk, and LEC (if SIE bit in CAN control register is set).

A change of bit EPass will not generate an interrupt.

---

**NOTE:** Reading the error and status register clears the WakeUpPnd, PER, RxOk and TxOk bits and set the LEC to value '7.' Additionally, the status interrupt value (0x8000) in the interrupt register will be replaced by the next lower priority interrupt value.

---



---

**NOTE:** For debug support, the auto clear functionality of error and status register (clear of status flags by read) is disabled when in debug/suspend mode.

---

### 23.4.1.3 Error Counter Register (DCAN ERRC)

The error counter register (DCAN ERRC) is shown in [Figure 23-21](#) and described in [Table 23-16](#).

**Figure 23-21. Error Counter Register (DCAN ERRC)**

|            |          |          |  |          |    |
|------------|----------|----------|--|----------|----|
| 31         | Reserved |          |  |          | 16 |
|            | R-0      |          |  |          |    |
| 15      14 |          | 8      7 |  |          | 0  |
| RP         | REC[6:0] |          |  | TEC[7:0] |    |
| R-0        | R-0      |          |  | R-0      |    |

LEGEND: R = Read only; -n = value after reset

**Table 23-16. Error Counter Register (DCAN ERRC) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 15    | RP       | 0     | Receive error passive  |
|       |          |       | The receive error counter is below the error passive level.  |
|       |          |       | The receive error counter has reached the error passive level as defined in the CAN specification. |
| 14-8  | REC[6:0] |       | Receive error counter. Actual state of the receive error counter (values from 0 to 255).           |
| 7-0   | TEC[7:0] |       | Transmit error counter. Actual state of the transmit error counter (values from 0 to 255).         |

### 23.4.1.4 Bit Timing Register (DCAN BTR)

The bit timing register (DCAN BTR) is shown in [Figure 23-22](#) and described in [Table 23-17](#).

**Figure 23-22. Bit Timing Register (DCAN BTR)**

|           |          |    |          |   |        |    |          |    |
|-----------|----------|----|----------|---|--------|----|----------|----|
| 31        | Reserved |    |          |   |        | 20 | 19       | 16 |
|           | R-0      |    |          |   |        |    | R-0      |    |
| 15        | 14       | 12 | 11       | 8 | 7      | 6  | 5        | 0  |
| Reserv ed | TSeg2    |    | TSeg1    |   | SJW    |    | BRP      |    |
| R-0       | R/WP-0x2 |    | R/WP-0x3 |   | R/WP-0 |    | R/WP-0x1 |    |

LEGEND: R = Read only; WP = Write protected by CCE bit; -n = value after reset

**Table 23-17. Bit Timing Register (DCAN BTR) Field Descriptions**

| Bit   | Field    | Value     | Description  |
|-------|----------|-----------|--|
| 31-20 | Reserved | 0         | These bits are always read as 0. Writes have no effect.  |
| 19-16 | BRPE     | 0x00-0x0F | Baud rate prescaler extension.<br><br>Valid programmed values are 0 to 15.<br><br>By programming BRPE the baud rate prescaler can be extended to values up to 1024.  |
| 15    | Reserved | 0         | These bits are always read as 0. Writes have no effect.  |
| 14-12 | TSeg2    | 0x0-0x7   | Time segment after the sample point<br><br>Valid programmed values are 0 to 7.<br><br>The actual TSeg2 value which is interpreted for the bit timing will be the programmed TSeg2 value + 1.   |
| 11-8  | TSeg1    | 0x01-0x0F | Time segment before the sample point<br><br>Valid programmed values are 1 to 15.<br><br>The actual TSeg1 value interpreted for the bit timing will be the programmed TSeg1 value + 1.  |
| 7-6   | SJW      | 0x0-0x3   | Synchronization Jump Width<br><br>Valid programmed values are 0 to 3.<br><br>The actual SJW value interpreted for the synchronization will be the programmed SJW value + 1.  |
| 5-0   | BRP      | 0x00-0x3F | Baud rate prescaler<br><br>Value by which the CAN_CLK frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta.<br><br>Valid programmed values are 0 to 63.<br><br>The actual BRP value interpreted for the bit timing will be the programmed BRP value + 1. |

**NOTE:** This register is only writable if CCE and Init bits in the CAN control register are set.

**NOTE:** The CAN bit time may be programmed in the range of 8 to 25 time quanta.

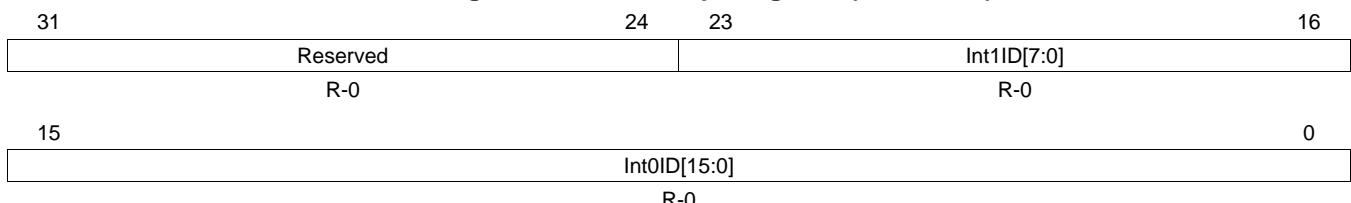
**NOTE:** The CAN time quantum may be programmed in the range of 1 to 1024 CAN\_CLK periods.

With a CAN\_CLK of 8 MHz and BRPE = 0x00, the reset value of 0x00002301 configures the DCAN for a bit rate of 500kBit/s.

### 23.4.1.5 Interrupt Register (DCAN INT)

The interrupt register (DCAN INT) is shown in [Figure 23-23](#) and described in [Table 23-18](#).

**Figure 23-23. Interrupt Register (DCAN INT)**



LEGEND: R = Read only; -n = value after reset

**Table 23-18. Interrupt Register (DCAN INT) Field Descriptions**

| Bit   | Field         | Value   | Description  |
|-------|---------------|---|--|
| 31-24 | Reserved      | 0   | These bits are always read as 0. Writes have no effect.  |
| 23-16 | Int1ID[23:16] | 0x00<br>0x01-0x80<br>0x81-0xFF  | Interrupt 1 Identifier (indicates the message object with the highest pending interrupt)<br>No interrupt is pending<br>Number of message object which caused the interrupt.<br>Unused<br>If several interrupts are pending, the CAN interrupt register will point to the pending interrupt with the highest priority. The DCAN1INT interrupt line remains active until Int1ID reaches value 0 (the cause of the interrupt is reset) or until IE1 is cleared.<br>A message interrupt is cleared by clearing the message object's IntPnd bit.<br>Among the message interrupts, the message object's interrupt priority decreases with increasing message number.               |
| 15-0  | Int0ID[15:0]  | 0x0000<br>0x0001-<br>0x0080<br>0x0081-<br>0x7FFF<br>0x8000<br>0x8001-<br>0xFFFF | Interrupt Identifier (the number here indicates the source of the interrupt)<br>No interrupt is pending<br>Number of message object which caused the interrupt.<br>Unused<br>Error and status register value is not 0x07.<br>Unused<br>If several interrupts are pending, the CAN interrupt register will point to the pending interrupt with the highest priority. The DCAN0INT interrupt line remains active until Int0ID reaches value 0 (the cause of the interrupt is reset) or until IE0 is cleared.<br>The Status interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. |

### 23.4.1.6 Test Register (DCAN TEST)

The test register (DCAN TEST) is shown in [Figure 23-24](#) and described in [Table 23-19](#).

**Figure 23-24. Test Register (DCAN TEST)**

|     |          |        |        |     |         |        |        |          |   |   |    |
|-----|----------|--------|--------|-----|---------|--------|--------|----------|---|---|----|
| 31  | Reserved |        |        |     |         |        |        |          |   |   | 16 |
| R-0 |          |        |        |     |         |        |        |          |   |   |    |
| 15  | Reserved | RDA    | EXL    | Rx  | Tx[1:0] | LBack  | Silent | Reserved | 2 | 1 | 0  |
|     | R-0      | R/WP-0 | R/WP-0 | R-U | R/WP-0  | R/WP-0 | R/WP-0 | R-0      |   |   |    |

LEGEND: R = Read only; WP = Write protected by test bit; -n = value after reset; U = Undefined

**Table 23-19. Test Register (DCAN TEST) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-10 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 9     | RDA      | 0     | RAM direct access enable                                 |
|       |          | 1     | Normal operation   |
|       |          | 1     | Direct access to the RAM is enabled while in test mode   |
| 8     | EXL      | 0     | External loopback mode                                   |
|       |          | 1     | Disabled   |
|       |          | 1     | Enabled  |
| 7     | Rx       | 0     | Receive pin. Monitors the actual value of the CAN_RX pin |
|       |          | 1     | The CAN bus is dominant                                  |
|       |          | 1     | The CAN bus is recessive                                 |
| 6-5   | Tx[1:0]  | 00    | Control of CAN_TX pin                                    |
|       |          | 01    | Normal operation, CAN_TX is controlled by the CAN core.  |
|       |          | 10    | Sample point can be monitored at CAN_TX pin.             |
|       |          | 11    | CAN_TX pin drives a dominant value.                      |
|       |          | 11    | CAN_TX pin drives a recessive value.                     |
| 4     | LBack    | 0     | Loopback mode  |
|       |          | 1     | Disabled   |
|       |          | 1     | Enabled  |
| 3     | Silent   | 0     | Silent mode  |
|       |          | 1     | Disabled   |
|       |          | 1     | Enabled  |
| 2-0   | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |

For all test modes, the test bit in CAN control register needs to be set to one. If test bit is set, the RDA, EXL, Tx1, Tx0, LBack and Silent bits are writable. Bit Rx monitors the state of pin CAN\_RX and therefore is only readable. All test register functions are disabled when test bit is cleared.

---

**NOTE:** The test register is only writable if test bit in CAN control register is set.

---



---

**NOTE:** Setting Tx[1:0] other than '00' will disturb message transfer.

---



---

**NOTE:** When the internal loop-back mode is active (bit LBack is set), bit EXL will be ignored.

---

#### 23.4.1.7 Parity Error Code Register (DCAN PERR)

The parity error code register (DCAN\_PERR) is shown in Figure 23-25 and described in Table 23-20.

**Figure 23-25. Parity Error Code Register (DCAN PERR)**

LEGEND: R = Read only; -n = value after reset; U = Undefined

**Table 23-20. Parity Error Code Register (DCAN PERR) Field Descriptions**

| Bit   | Field          | Value     | Description   |
|-------|----------------|-----------|---|
| 31-11 | Reserved       | 0         | These bits are always read as 0. Writes have no effect.   |
| 10-8  | Word Number    | 0x01-0x05 | Word number where parity error has been detected<br>RDA word number (1 to 5) of the message object (according to the message RAM representation in RDA mode). |
| 7-0   | Message Number | 0x01-0x80 | Message object number where parity error has been detected  |

If a parity error is detected, the PER flag will be set in the error and status register. This bit is not reset by the parity check mechanism; it must be reset by reading the error and status register.

In addition to the PER flag, the parity error code register will indicate the memory area where the parity error has been detected (message number and word number).

If more than one word with a parity error was detected, the highest word number with a parity error will be displayed.

After a parity error has been detected, the register will hold the last error code until power is removed.

### 23.4.1.8 Auto-Bus-On Time Register (DCAN ABOTR)

The Auto-Bus-On time register (DCAN ABOTR) is shown in [Figure 23-26](#) and described in [Table 23-21](#).

**Figure 23-26. Auto-Bus-On Time Register (DCAN ABOTR)**

| 31              | 16 15 | 0              |
|-----------------|-------|----------------|
| ABO Time[31:16] |       | ABO Time[15:0] |
| R/W-0           |       | R/W-0          |

LEGEND: R = Read only; W = Write only; -n = value after reset

**Table 23-21. Auto-Bus-On Time Register (DCAN ABOTR) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-0 | ABO Time |       | <p>Number of OCP clock cycles before a Bus-Off recovery sequence is started by clearing the Init bit.</p> <p>This function has to be enabled by setting bit ABO in CAN control register.</p> <p>The Auto-Bus-On timer is realized by a 32-bit counter that starts to count down to zero when the module goes Bus-Off.</p> <p>The counter will be reloaded with the preload value of the ABO time register after this phase.</p> |

---

**NOTE:** On write access to the CAN control register while Auto-Bus-On timer is running, the Auto-Bus-On procedure will be aborted.

---



---

**NOTE:** During Debug/Suspend mode, running Auto-Bus-On timer will be paused.

---

### 23.4.1.9 Transmission Request X Register (DCAN TXRQ X)

The transmission request X register (DCAN TXRQ X) is shown in [Figure 23-27](#).

**Figure 23-27. Transmission Request X Register (DCAN TXRQ X)**

| 31         |            |            |            |            |            |            |            |     |     |     |     |     |     |     |     | 16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|------------|------------|------------|------------|------------|------------|------------|------------|-----|-----|-----|-----|-----|-----|-----|-----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| Reserved   |            |            |            |            |            |            |            |     |     |     |     |     |     |     |     |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R-0        |            |            |            |            |            |            |            |     |     |     |     |     |     |     |     |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 15         | 14         | 13         | 12         | 11         | 10         | 9          | 8          | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TxRqstReg8 | TxRqstReg7 | TxRqstReg6 | TxRqstReg5 | TxRqstReg4 | TxRqstReg3 | TxRqstReg2 | TxRqstReg1 |     |     |     |     |     |     |     |     |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R-0        | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

LEGEND: R = Read only; -n = value after reset

**Example 1.** Bit 0 of the transmission request X register represents byte 0 of the transmission request 1 register. If one or more bits in this byte are set, bit 0 of the transmission request X register will be set.

### 23.4.1.10 Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78)

These registers hold the TxRqst bits of the implemented message objects. By reading out these bits, the CPU can check for pending transmission requests. The TxRqst bit in a specific message object can be set/reset by the CPU via the IF1/IF2 message interface registers, or by the message handler after reception of a remote frame or after a successful transmission.

The transmission request registers are shown in [Table 23-22](#) and described in [Table 23-23](#).

**Table 23-22. Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78)**

| Offset/Register    | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7        | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0 |
|--------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|-----------------|----------|----------|----------|----------|----------|----------|----------|
| 0x88<br>CAN TXRQ12 |           |           |           |           |           |           |          |          | TxRqst[32:17]   |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | TxRqst[16:1]    |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
| 0x8C<br>CAN TXRQ34 |           |           |           |           |           |           |          |          | TxRqst[64:49]   |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | TxRqst[48:33]   |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
| 0x90<br>CAN TXRQ56 |           |           |           |           |           |           |          |          | TxRqst[96:81]   |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | TxRqst[80:65]   |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
| 0x94<br>CAN TXRQ78 |           |           |           |           |           |           |          |          | TxRqst[128:113] |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | TxRqst[112:97]  |          |          |          |          |          |          |          |
|                    |           |           |           |           |           |           |          |          | R-0             |          |          |          |          |          |          |          |

LEGEND: R = Read only; -n = value after reset

**Table 23-23. Transmission Request Registers (DCAN TXRQ12 to DCAN TXRQ78) Field Descriptions**

| Bit  | Field        | Value | Description  |
|------|--------------|-------|--|
| 31-0 | TxRqs[128:1] | 0     | Transmission request bits (for all message objects)<br>No transmission has been requested for this message object. |
|      |              | 1     | The transmission of this message object is requested and is not yet done.  |

### 23.4.1.11 New Data X Register (DCAN NWDAT X)

With the new data X register, the CPU can detect if one or more bits in the different new data registers are set. Each register bit represents a group of eight message objects. If at least one of the NewDat bits of these message objects are set, the corresponding bit in the new data X register will be set.

The new data X register (DCAN NWDAT X) is shown in [Figure 23-28](#).

**Figure 23-28. Transmission Request X Register (DCAN TXRQ X)**

| 31         | Reserved   |            |            |            |            |            |            |     |     |     |     |     |     |     |     | 16  |
|------------|------------|------------|------------|------------|------------|------------|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| R-0        |            |            |            |            |            |            |            |     |     |     |     |     |     |     |     |     |
| 15         | 14         | 13         | 12         | 11         | 10         | 9          | 8          | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |     |
| NewDatReg8 | NewDatReg7 | NewDatReg6 | NewDatReg5 | NewDatReg4 | NewDatReg3 | NewDatReg2 | NewDatReg1 | R-0 |

LEGEND: R = Read only; -n = value after reset

**Example 1.** Bit 0 of the new data X register represents byte 0 of the new data 1 register. If one or more bits in this byte are set, bit 0 of the new data X register will be set.

### 23.4.1.12 New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)

These registers hold the NewDat bits of the implemented message objects. By reading out these bits, the CPU can check for new data in the message objects. The NewDat bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after reception of a data frame or after a successful transmission.

The new data registers (DCAN NWDAT12 to DCAN NWDAT78) are shown in [Table 23-24](#) and described in [Table 23-25](#).

**Table 23-24. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)**

| Offset/Register     | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0        |
|---------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|
| 0x9C<br>CAN NWDAT12 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[32:17]   |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[16:1]    |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
| 0xA0<br>CAN NWDAT34 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[64:49]   |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[48:33]   |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
| 0xA4<br>CAN NWDAT56 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[96:81]   |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[80:65]   |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
| 0xA8<br>CAN NWDAT78 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[128:113] |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | NewDat[112:97]  |
|                     |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |

LEGEND: R = Read only; -n = value after reset

**Table 23-25. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) Field Descriptions**

| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 31-0 | NewDat[128:1] | 0     | New Data Bits (for all message objects)<br>No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU. |
|      |               | 1     | The message handler or the CPU has written new data into the data portion of this message object.  |

### 23.4.1.13 Interrupt Pending X Register (DCAN INTPND X)

With the interrupt pending X register, the CPU can detect if one or more bits in the different interrupt pending registers are set. Each bit of this register represents a group of eight message objects. If at least one of the IntPnd bits of these message objects are set, the corresponding bit in the interrupt pending X register will be set.

The interrupt pending X register (DCAN INTPND X) is shown in [Figure 23-29](#).

**Figure 23-29. Interrupt Pending X Register (DCAN INTPND X)**

|            |            |            |            |            |            |            |            |  |  |  |  |  |  |  |  |     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|------------|------------|------------|------------|------------|------------|------------|------------|--|--|--|--|--|--|--|--|-----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 31         |            |            |            |            |            |            |            |  |  |  |  |  |  |  |  | 16  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Reserved   |            |            |            |            |            |            |            |  |  |  |  |  |  |  |  | R-0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|            |            |            |            |            |            |            |            |  |  |  |  |  |  |  |  | 0   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| IntPndReg8 | IntPndReg7 | IntPndReg6 | IntPndReg5 | IntPndReg4 | IntPndReg3 | IntPndReg2 | IntPndReg1 |  |  |  |  |  |  |  |  | 1   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R-0        |  |  |  |  |  |  |  |  | 0   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

LEGEND: R = Read only; -n = value after reset

**Example 2.** Bit 0 of the interrupt pending X register represents byte 0 of the interrupt pending 1 register. If one or more bits in this byte are set, bit 0 of the interrupt pending X register will be set.

### 23.4.1.14 Interrupt Pending Registers (DCAN INTPND12 to DCAN INTPND78)

These registers hold the IntPnd bits of the implemented message objects. By reading out these bits, the CPU can check for pending interrupts in the message objects. The IntPnd bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

The interrupt pending registers (DCAN INTPND12 to DCAN INTPND78) are shown in [Table 23-26](#) and described in [Table 23-27](#).

**Table 23-26. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78)**

| Offset/Register      | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0        |
|----------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|
| 0xB0<br>CAN INTPND12 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                 |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[32:17]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[16:1]    |
| 0xB4<br>CAN INTPND34 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[64:49]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[48:33]   |
| 0xB8<br>CAN INTPND56 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[96:81]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[80:65]   |
| 0xBC<br>CAN INTPND78 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[128:113] |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntPnd[112:97]  |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |

LEGEND: R = Read only; -n = value after reset

**Table 23-27. New Data Registers (DCAN NWDAT12 to DCAN NWDAT78) Field Descriptions**

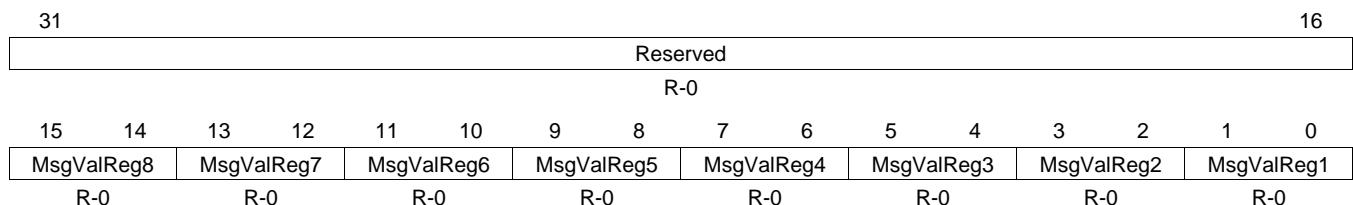
| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 31-0 | IntPnd[128:1] | 0     | Interrupt Pending Bits (for all message objects)<br>This message object is not the source of an interrupt. |

### 23.4.1.15 Message Valid X Register (DCAN MSGVAL X)

With the message valid X register, the CPU can detect if one or more bits in the different message valid registers are set. Each bit of this register represents a group of eight message objects. If at least one of the MsgVal bits of these message objects are set, the corresponding bit in the message valid X register will be set.

The message valid X register (DCAN MSGVAL X) is shown in [Figure 23-30](#).

**Figure 23-30. Message Valid X Register (DCAN MSGVAL X)**



LEGEND: R = Read only; -n = value after reset

**Example 3.** Bit 0 of the message valid X register represents byte 0 of the message valid 1 register. If one or more bits in this byte are set, bit 0 of the message valid X register will be set.

### 23.4.1.16 Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78)

These registers hold the MsgVal bits of the implemented message objects. By reading out these bits, the CPU can check which message objects are valid. The MsgVal bit of a specific message object can be set/reset by the CPU via the IF1/IF2 interface register sets, or by the message handler after a reception or a successful transmission.

The message valid registers (DCAN MSGVAL12 to MSGVAL78) is shown in [Table 23-28](#) and described in [Table 23-29](#).

**Table 23-28. Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78)**

| Offset/Register      | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0        |
|----------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|
| 0xC4<br>CAN MSGVAL12 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                 |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[32:17]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[16:1]    |
| 0xC8<br>CAN MSGVAL34 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[64:49]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[48:33]   |
| 0xCC<br>CAN MSGVAL56 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[96:81]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[80:65]   |
| 0xD0<br>CAN MSGVAL78 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[128:113] |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | MsgVal[112:97]  |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |

LEGEND: R = Read only; -n = value after reset

**Table 23-29. Message Valid Registers (DCAN MSGVAL12 to DCAN MSGVAL78) Field Descriptions**

| Bit  | Field         | Value | Description  |
|------|---------------|-------|--|
| 31-0 | MsgVal[128:1] | 0     | Message valid bits (for all message objects)<br>This message object is ignored by the message handler. |

### 23.4.1.17 Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)

The IntMux flag determine for each message object, which of the two interrupt lines (DCAN0INT or DCAN1INT) will be asserted when the IntPnd of this message object is set. Both interrupt lines can be globally enabled or disabled by setting or clearing IE0 and IE1 bits in CAN control register.

The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 interface register sets, or by message handler after reception or successful transmission of a frame. This will also affect the Int0ID resp Int1ID flags in the interrupt register.

The interrupt multiplexer registers are shown in [Table 23-30](#) and described in [Table 23-31](#).

**Table 23-30. Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78)**

| Offset/Register      | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0        |
|----------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------------|
| 0xD8<br>CAN INTMUX12 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                 |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[32:17]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[16:1]    |
| 0xDC<br>CAN INTMUX34 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[64:49]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[48:33]   |
| 0xE0<br>CAN INTMUX56 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[96:81]   |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[80:65]   |
| 0xE4<br>CAN INTMUX78 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[128:113] |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IntMux[112:97]  |
|                      |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R-0             |

LEGEND: R = Read only; -n = value after reset

**Table 23-31. Interrupt Multiplexer Registers (DCAN INTMUX12 to DCAN INTMUX78) Field Descriptions**

| Bit  | Field         | Value | Description   |
|------|---------------|-------|---|
| 31-0 | IntMux[128:1] | 0     | Multiplexes IntPnd value to either DCAN0INT or DCAN1INT interrupt lines (for all message objects)<br>DCAN0INT line is active if corresponding IntPnd flag is one. |
|      |               | 1     | DCAN1INT line is active if corresponding IntPnd flag is one.  |

### 23.4.1.18 IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD)

The IF1/IF2 Command Register (DCAN IF1CMD, DCAN IF2CMD) configure and initiate the transfer between the IF1/IF2 register sets and the message RAM. It is configurable which portions of the message object should be transferred.

A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 command register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress.

After 4 to 14 OCP clock cycles, the transfer between the interface register and the message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer concurs with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 command registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

The IF1/IF2 command registers are shown in [Figure 23-31](#) and [Figure 23-32](#) and described in [Table 23-32](#).

**NOTE:** While Busy bit is one, IF1/IF2 register sets are write protected.

**NOTE:** For debug support, the auto clear functionality of the IF1/IF2 command registers (clear of DMAactive flag by r/w) is disabled during Debug/Suspend mode.

**NOTE:** If an invalid Message Number is written to bits [7:0] of the IF1/IF2 command register, the message handler may access an implemented (valid) message object instead.

**Figure 23-31. IF1 Command Registers (DCAN IF1CMD)**

| 31                 | 24         | 23       | 22     | 21       | 20         | 19               | 18             | 17     | 16     |
|--------------------|------------|----------|--------|----------|------------|------------------|----------------|--------|--------|
| Reserved           | WR/R D     | Mask     | Arb    | Contro l | Clr IntPnd | TxRqst / New-Dat | Data A         | Data B |        |
| R-0                | R/WP-0     | R/WP-0   | R/WP-0 | R/WP-0   | R/WP-0     | R/WP-0           | R/WP-0         | R/WP-0 | R/WP-0 |
| 15      14      13 | 8          | 7        |        |          |            |                  |                |        | 0      |
| Busy               | DMA active | Reserved |        |          |            |                  | Message Number |        |        |
| R-0                | R/WP/C-0   | R-0      |        |          |            |                  | R/WP-0x1       |        |        |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); C = Clear by IF1 access; -n = value after reset

**Figure 23-32. IF2 Command Registers (DCAN IF2CMD)**

| 31                 | 24         | 23       | 22     | 21       | 20         | 19               | 18             | 17     | 16     |
|--------------------|------------|----------|--------|----------|------------|------------------|----------------|--------|--------|
| Reserved           | WR/R D     | Mask     | Arb    | Contro l | Clr IntPnd | TxRqst / New-Dat | Data A         | Data B |        |
| R-0                | R/WP-0     | R/WP-0   | R/WP-0 | R/WP-0   | R/WP-0     | R/WP-0           | R/WP-0         | R/WP-0 | R/WP-0 |
| 15      14      13 | 8          | 7        |        |          |            |                  |                |        | 0      |
| Busy               | DMA active | Reserved |        |          |            |                  | Message Number |        |        |
| R-0                | R/WP/C-0   | R-0      |        |          |            |                  | R/WP-0x1       |        |        |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); C = Clear by IF2 access; -n = value after reset

**Table 23-32. IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD) Field Descriptions**

| Bit   | Field         | Value | Description  |
|-------|---------------|-------|--|
| 31-24 | Reserved      | 0     | These bits are always read as 0. Writes have no effect.  |
| 23    | WR/RD         | 0     | Write/Read<br>Direction = Read: Transfer direction is from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 register set.  |
|       |               | 1     | Direction = Write: Transfer direction is from the IF1/IF2 register set to the message object addressed by Message Number (Bits [7:0]).   |
|       |               |       |  |
| 22    | Mask          | 0     | Access mask bits<br>Mask bits will not be changed  |
|       |               | 1     | Direction = Read: The mask bits (identifier mask + MDir + MXtd) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the IF1/IF2 register set.<br>Direction = Write: The mask bits (identifier mask + MDir + MXtd) will be transferred from the IF1/IF2 register set to the message object addressed by Message Number (Bits [7:0]).  |
|       |               |       |  |
| 21    | Arb           | 0     | Access arbitration bits<br>Arbitration bits will not be changed  |
|       |               | 1     | Direction = Read: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 register set.<br>Direction = Write: The Arbitration bits (Identifier + Dir + Xtd + MsgVal) will be transferred from the IF1/IF2 register set to the message object addressed by Message Number (Bits [7:0]).  |
|       |               |       |  |
| 20    | Control       | 0     | Access control bits<br>Control bits will not be changed  |
|       |               | 1     | Direction = Read: The message control bits will be transferred from the message object addressed by message number (Bits [7:0]) to the IF1/IF2 registerset.<br>Direction = Write: The message control bits will be transferred from the IF1/IF2 registerset to the message object addressed by message number (Bits [7:0]).<br>If the TxRqst/NewDat bit in this register(Bit [18]) is set, the TxRqst/ NewDat bit in the IF1/IF2 message control register will be ignored.   |
|       |               |       |  |
| 19    | ClrlntPnd     | 0     | Clear interrupt pending bit<br>IntPnd bit will not be changed  |
|       |               | 1     | Direction = Read: Clears IntPnd bit in the message object.<br>Direction = Write: This bit is ignored. Copying of IntPnd flag from IF1/IF2 Registers to message RAM can only be controlled by the control flag (Bit [20]).  |
|       |               |       |  |
| 18    | TxRqst/NewDat | 0     | Access transmission request bit<br>Direction = Read: NewDat bit will not be changed. Direction = Write: TxRqst/NewDat bit will be handled according to the control bit.  |
|       |               | 1     | Direction = Read: Clears NewDat bit in the message object.<br>Direction = Write: Sets TxRqst/NewDat in message object.<br>Note: If a CAN transmission is requested by setting TxRqst/NewDat in this register, the TxRqst/NewDat bits in the message object will be set to one independent of the values in IF1/IF2 message control Register.<br>Note: A read access to a message object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IF1/IF2 message control registeralways reflect the status before resetting them. |
|       |               |       |  |
| 17    | Data A        | 0     | Access Data Bytes 0-3<br>Data Bytes 0-3 will not be changed.   |
|       |               | 1     | Direction = Read: The data bytes 0-3 will be transferred from the message object addressed by the Message Number (Bits [7:0]) to the corresponding IF1/IF2 registerset.<br>Direction = Write: The data bytes 0-3 will be transferred from the IF1/IF2 register set to the message object addressed by the Message Number (Bits [7:0]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred.  |
|       |               |       |  |

**Table 23-32. IF1/IF2 Command Registers (DCAN IF1CMD, DCAN IF2CMD) Field Descriptions (continued)**

| Bit  | Field          | Value     | Description  |
|------|----------------|-----------|--|
| 16   | Data B         | 0         | Access Data Bytes 4-7<br>Data Bytes 4-7 will not be changed.   |
|      |                | 1         | Direction = Read: The data bytes 4-7 will be transferred from the message object addressed by Message Number (Bits [7:0]) to the corresponding IF1/IF2 register set.<br>Direction = Write: The data bytes 4-7 will be transferred from the IF1/IF2 register set to the message object addressed by message number (Bits [7:0]).<br>Note: The duration of the message transfer is independent of the number of bytes to be transferred. |
| 15   | Busy           | 0         | Busy flag<br>No transfer between IF1/IF2 register set and message RAM is in progress.  |
|      |                | 1         | Transfer between IF1/IF2 register set and message RAM is in progress.<br>This bit is set to one after the message number has been written to bits [7:0]. IF1/IF2 register set will be write protected. The bit is cleared after read/write action has been finished.   |
| 14   | DMAactive      | 0         | Activation of DMA feature for subsequent internal IF1/IF2 update<br>DMA request line is independent of IF1/IF2 activities.   |
|      |                | 1         | DMA is requested after completed transfer between IF1/IF2 register set and message RAM.<br>The DMA request remains active until the first read or write to one of the IF1/IF2 registers; an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.<br>Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.                                   |
| 13-8 | Reserved       | 0         | These bits are always read as 0. Writes have no effect.  |
| 7-0  | Message Number | 0x00      | Number of message object in message RAM which is used for data transfer  |
|      |                | 0x01-0x80 | Invalid message number   |
|      |                | 0x81-0xFF | Valid message numbers  |
|      |                |           | Invalid message numbers  |

### 23.4.1.19 IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK)

The bits of the IF1/IF2 mask registers mirror the mask bits of a message object. The function of the relevant message objects bits is described in *Structure of Message Objects*.

The IF1/IF2 mask registers (DCAN IF1MSK, DCAN IF2MSK) are shown in [Figure 23-33](#) and [Figure 23-34](#) and described in [Table 23-33](#).

---

**NOTE:** While Busy bit of IF1/IF2 command register is one, IF1/IF2 register set is write protected.

---

**Figure 23-33. IF1 Mask Register (DCAN IF1MSK)**

| 31     | 30     | 29       | 28        | 16          |
|--------|--------|----------|-----------|-------------|
| MXtd   | MDir   | Reserved |           | Msk[28:16]  |
| R/WP-1 | R/WP-1 | R-1      |           | R/WP-0x1FFF |
| 15     |        |          |           | 0           |
|        |        |          | Msk[15:0] |             |
|        |        |          |           | R/WP-0xFFFF |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); -n = value after reset

**Figure 23-34. IF2 Mask Register (DCAN IF2MSK)**

| 31     | 30     | 29       | 28        | 16          |
|--------|--------|----------|-----------|-------------|
| MXtd   | MDir   | Reserved |           | Msk[28:16]  |
| R/WP-1 | R/WP-1 | R-1      |           | R/WP-0x1FFF |
| 15     |        |          |           | 0           |
|        |        |          | Msk[15:0] |             |
|        |        |          |           | R/WP-0xFFFF |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); -n = value after reset

**Table 23-33. IF1/IF2 Mask Registers (DCAN IF1MSK, DCAN IF2MSK) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 31   | MXtd      | 0     | Mask Extended Identifier<br>The extended identifier bit (IDE) has no effect on the acceptance filtering.   |
|      |           | 1     | The extended identifier bit (IDE) is used for acceptance filtering.<br>When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. |
| 30   | MDir      | 0     | Mask Message Direction<br>The message direction bit (Dir) has no effect on the acceptance filtering.   |
|      |           | 1     | The message direction bit (Dir) is used for acceptance filtering.  |
| 29   | Reserved  | 0     | These bits are always read as 0. Writes have no effect.  |
| 28-0 | Msk[28:0] | 0     | Identifier Mask<br>The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).  |
|      |           | 1     | The corresponding bit in the identifier of the message object is used for acceptance filtering.  |

### 23.4.1.20 IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB)

The bits of the IF1/IF2 arbitration registers mirror the arbitration bits of a message object. The function of the relevant message objects bits is described in *Structure of Message Objects*.

The IF1/IF2 arbitration registers (DCAN IF1ARB, DCAN IF2ARB) are shown in [Figure 23-35](#) and [Figure 23-36](#) and described in [Table 23-34](#).

---

**NOTE:** While Busy bit of IF1/IF2 command register is one, IF1/IF2 register set is write protected.

---

**Figure 23-35. IF1 Arbitration Register (DCAN IF1ARB)**

| 31         | 30         | 29         | 28       | 16        |
|------------|------------|------------|----------|-----------|
| MsgVa<br>I | Xtd        | Dir        |          | ID[28:16] |
| R/WP-<br>0 | R/WP-<br>0 | R/WP-<br>0 |          | R/WP-0    |
| 15         |            |            |          | 0         |
|            |            |            | ID[15:0] |           |
|            |            |            | R/WP-0   |           |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); -n = value after reset

**Figure 23-36. IF2 Arbitration Register (DCAN IF2ARB)**

| 31         | 30         | 29         | 28       | 16        |
|------------|------------|------------|----------|-----------|
| MsgVa<br>I | Xtd        | Dir        |          | ID[28:16] |
| R/WP-<br>0 | R/WP-<br>0 | R/WP-<br>0 |          | R/WP-0    |
| 15         |            |            |          | 0         |
|            |            |            | ID[15:0] |           |
|            |            |            | R/WP-0   |           |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); -n = value after reset

**Table 23-34. IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) Field Descriptions**

| Bit | Field  | Value  | Description  |
|-----|--------|--------|--|
| 31  | MsgVal | 0<br>1 | Message valid<br><br>The message object is ignored by the message handler.<br><br>The message object is to be used by the message handler.<br><br>The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN control register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required. |
| 30  | Xtd    | 0<br>1 | Extended identifier<br><br>The 11-bit ("standard") Identifier is used for this message object.<br><br>The 29-bit ("extended") Identifier is used for this message object.  |
| 29  | Dir    | 0<br>1 | Message direction<br><br>Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, this message is stored in this message object.<br><br>Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).   |

**Table 23-34. IF1/IF2 Arbitration Registers (DCAN IF1ARB, DCAN IF2ARB) Field Descriptions (continued)**

| Bit  | Field     | Value                 | Description  |
|------|-----------|-----------------------|--|
| 28-0 | Msk[28:0] | ID[28:0]<br>ID[28:18] | Message identifier<br>29-bit identifier (extended frame)<br>11-bit identifier (standard frame) |

The Arbitration bits ID[28:0], Xtd, and Dir are used to define the identifier and type of outgoing messages and (together with the mask bits Msk[28:0], MXtd, and MDir) for acceptance filtering of incoming messages.

A received message is stored into the valid message object with matching identifier and Direction = receive (data frame) or Direction = transmit (remote frame).

Extended frames can be stored only in message objects with Xtd = one, standard frames in message objects with Xtd = zero.

If a received message (data frame or remote frame) matches more than one valid message objects, it is stored into the one with the lowest message number.

### 23.4.1.21 IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL)

The bits of the IF1/IF2 message control registers mirror the message control bits of a message object. The function of the relevant message objects bits is described in *Structure of Message Objects*.

The IF1/IF2 message control registers (DCAN IF1MCTL, DCAN IF2MCTL) are shown in [Figure 23-37](#) and [Figure 23-38](#) and described in [Table 23-35](#).

---

**NOTE:** While Busy bit of IF1/IF2 command register is one, IF1/IF2 register set is write protected.

---

**Figure 23-37. IF1 Message Control Register (DCAN IF1MCTL)**

| 31      | Reserved |         |         |         |         |         |         |         |          |         |          |         | 16      |         |
|---------|----------|---------|---------|---------|---------|---------|---------|---------|----------|---------|----------|---------|---------|---------|
| R/WP- 0 |          |         |         |         |         |         |         |         |          |         |          |         |         |         |
| 15      | 14       | 13      | 12      | 11      | 10      | 9       | 8       | 7       | 6        | 4       | 3        | 2       | 1       | 0       |
| New Dat | Msg Lst  | Int Pnd | UMask   | TxIE    | RxE     | Rmt En  | Tx Rqst | EoB     | Reserved |         | DLC[3:0] |         |         |         |
| R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0 | R/WP- 0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

**Figure 23-38. IF2 Message Control Register (DCAN IF2MCTL)**

| 31      | Reserved |         |         |         |         |         |         |         |          |         |          |         | 16      |         |
|---------|----------|---------|---------|---------|---------|---------|---------|---------|----------|---------|----------|---------|---------|---------|
| R/WP- 0 |          |         |         |         |         |         |         |         |          |         |          |         |         |         |
| 15      | 14       | 13      | 12      | 11      | 10      | 9       | 8       | 7       | 6        | 4       | 3        | 2       | 1       | 0       |
| New Dat | Msg Lst  | Int Pnd | UMask   | TxIE    | RxE     | Rmt En  | Tx Rqst | EoB     | Reserved |         | DLC[3:0] |         |         |         |
| R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0  | R/WP- 0 | R/WP- 0 | R/WP- 0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

**Table 23-35. IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 15    | NewDat   | 0     | New data   |
|       |          | 0     | No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.  |
|       |          | 1     | The message handler or the CPU has written new data into the data portion of this message object.  |
| 14    | MsgLst   | 0     | Message lost (only valid for message objects with direction = receive)   |
|       |          | 0     | No message lost since the last time when this bit was reset by the CPU.  |
|       |          | 1     | The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.   |
| 13    | IntPnd   | 0     | Interrupt pending  |
|       |          | 0     | This message object is not the source of an interrupt.   |
|       |          | 1     | This message object is the source of an interrupt. The Interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.                            |
| 12    | UMask    | 0     | Use acceptance mask  |
|       |          | 0     | Mask ignored   |
|       |          | 1     | Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering<br>If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. |

**Table 23-35. IF1/IF2 Message Control Registers (DCAN IF1MCTL, DCAN IF2MCTL) Field Descriptions (continued)**

| Bit | Field    | Value       | Description  |
|-----|----------|-------------|--|
| 11  | TxIE     | 0<br>1      | Transmit interrupt enable<br><br>IntPnd will not be triggered after the successful transmission of a frame.<br><br>IntPnd will be triggered after the successful transmission of a frame.  |
| 10  | RxIE     | 0<br>1      | Receive interrupt enable<br><br>IntPnd will not be triggered after the successful reception of a frame.<br><br>IntPnd will be triggered after the successful reception of a frame.   |
| 9   | RmtEn    | 0<br>1      | Remote enable<br><br>At the reception of a remote frame, TxRqst is not changed.<br><br>At the reception of a remote frame, TxRqst is set.  |
| 8   | TxRqst   | 0<br>1      | Transmit request<br><br>This message object is not waiting for a transmission.<br><br>The transmission of this message object is requested and is not yet done.  |
| 7   | EoB      | 0<br>1      | Data frame has 0-8 data bits.<br><br>Data frame has 8 data bytes.<br><br>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.<br><br>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one. |
| 6-4 | Reserved | 0           | These bits are always read as 0. Writes have no effect.  |
| 3-0 |          | 0-8<br>9-15 | Data length code<br><br>Data frame has 0-8 data bits.<br><br>Data frame has 8 data bytes.<br><br>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.  |

### 23.4.1.22 IF1/IF2 Data A and Data B Registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB)

The data bytes of CAN messages are stored in the IF1/IF2 registers in the following order:

1. In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received.
2. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

The IF1/IF2 data A and data B registers (DCAN IF1DATA/DATB, DCAN IF2DATA/DATB) registers are shown in [Figure 23-39](#), [Figure 23-40](#), [Figure 23-41](#) and [Figure 23-42](#).

**Figure 23-39. IF1 Data A Register (DCAN IF1DATA)**

| 31     | 24 | 23 | 16     |
|--------|----|----|--------|
| Data 3 |    |    | Data 2 |
| R/WP-0 |    |    | R/WP-0 |
| 15     | 8  | 7  | 0      |
| Data 1 |    |    | Data 0 |
| R/WP-0 |    |    | R/WP-0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

**Figure 23-40. IF1 Data B Register (DCAN IF1DATA)**

| 31     | 24 | 23 | 16     |
|--------|----|----|--------|
| Data 7 |    |    | Data 6 |
| R/WP-0 |    |    | R/WP-0 |
| 15     | 8  | 7  | 0      |
| Data 5 |    |    | Data 4 |
| R/WP-0 |    |    | R/WP-0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

**Figure 23-41. IF2 Data A Register (DCAN IF2DATA)**

| 31     | 24 | 23 | 16     |
|--------|----|----|--------|
| Data 7 |    |    | Data 6 |
| R/WP-0 |    |    | R/WP-0 |
| 15     | 8  | 7  | 0      |
| Data 5 |    |    | Data 4 |
| R/WP-0 |    |    | R/WP-0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

**Figure 23-42. IF2 Data B Register (DCAN IF2DATA)**

| 31     | 24 | 23 | 16     |
|--------|----|----|--------|
| Data 7 |    |    | Data 6 |
| R/WP-0 |    |    | R/WP-0 |
| 15     | 8  | 7  | 0      |
| Data 5 |    |    | Data 4 |
| R/WP-0 |    |    | R/WP-0 |

LEGEND: R = Read only; WP = Protected Write (protected by Busy bit); -n = value after reset

### 23.4.1.23 IF3 Observation Register (DCAN IF3OBS)

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from message RAM by CPU (Additional information can be found in *Structure of Message Objects*).

The observation flags (Bits [4:0]) in the IF3 observation register are used to determine, which data sections of the IF3 interface register set have to be read in order to complete a DMA read cycle. After all marked data sections are read, the DCAN is enabled to update the IF3 interface register set with new data.

Any access order of single bytes or half-words is supported. When using byte or half-word accesses, a data section is marked as completed, if all bytes are read.

---

**NOTE:** If IF3 Update Enable is used and no Observation flag is set, the corresponding message objects will be copied to IF3 without activating the DMA request line and without waiting for DMA read accesses.

---

A write access to this register aborts a pending DMA cycle by resetting the DMA line and enables updating of IF3 interface register set with new data. To avoid data inconsistency, the DMA controller should be disabled before reconfiguring IF3 observation register.

The status of the current read-cycle can be observed via status flags (Bits [12:8]).

If an interrupt line is available for IF3, an interrupt will be generated by IF3Upd flag. See the device-specific data sheet for the availability of this interrupt source.

With this interrupt, the observation status bits and the IF3Upd bit could be used by the application to realize the notification about new IF3 content in polling or interrupt mode.

The IF3 observation register (DCAN IF3OBS) is shown in [Figure 23-43](#) and described in [Table 23-36](#).

**Figure 23-43. IF3 Observation Register (DCAN IF3OBS)**

| 31      | Reserved |         |         |        |        |        |          |          |       |       |       |       |       |       |       | 16 |
|---------|----------|---------|---------|--------|--------|--------|----------|----------|-------|-------|-------|-------|-------|-------|-------|----|
| R-0     |          |         |         |        |        |        |          |          |       |       |       |       |       |       |       |    |
| 15      | 14       | 13      | 12      | 11     | 10     | 9      | 8        | 7        | 5     | 4     | 3     | 2     | 1     | 0     |       |    |
| IF3 Upd | Reserved | IF3 SDB | IF3 SDA | IF3 SC | IF3 SA | IF3 SM | Reserved | Reserved | DataB | DataA | Ctrl  | Arb   | Mask  |       |       |    |
| R-0     | R-0      | R-0     | R-0     | R-0    | R-0    | R-0    | R-0      | R-0      | R/W-0 |    |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 23-36. IF3 Observation Register (DCAN IF3OBS) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | Reserved | 0     | These bits are always read as 0. Writes have no effect.              |
| 15    | IF3 Upd  | 0     | IF3 Update Data  |
|       |          | 1     | No new data has been loaded since last IF3 read.                     |
|       |          | 1     | New data has been loaded since last IF3 read.                        |
| 14-13 | Reserved | 0     | These bits are always read as 0. Writes have no effect.              |
| 12    | IF3 SDB  | 0     | IF3 Status of Data B read access                                     |
|       |          | 1     | All Data B bytes are already read out, or are not marked to be read. |
|       |          | 1     | Data B section has still data to be read out.                        |
| 11    | IF3 SDA  | 0     | IF3 Status of Data A read access                                     |
|       |          | 0     | All Data A bytes are already read out, or are not marked to be read. |
|       |          | 1     | Data A section has still data to be read out.                        |

**Table 23-36. IF3 Observation Register (DCAN IF3OBS) Field Descriptions (continued)**

| Bit | Field    | Value | Description  |
|-----|----------|-------|--|
| 10  | IF3 SC   | 0     | IF3 Status of control bits read access<br>All control section bytes are already read out, or are not marked to be read.      |
|     |          | 1     | Control section has still data to be read out.   |
| 9   | IF3 SA   | 0     | IF3 Status of Arbitration data read access<br>All Arbitration data bytes are already read out, or are not marked to be read. |
|     |          | 1     | Arbitration section has still data to be read out.   |
| 8   | IF3 SM   | 0     | IF3 Status of Mask data read access<br>All mask data bytes are already read out, or are not marked to be read.               |
|     |          | 1     | Mask section has still data to be read out.  |
| 7-5 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 4   | DataB    | 0     | Data B read observation<br>Data B section has not to be read.  |
|     |          | 1     | Data B section has to be read to enable next IF3 update.   |
| 3   | DataA    | 0     | Data A read observation<br>Data A section has not to be read.  |
|     |          | 1     | Data A section has to be read to enable next IF3 update.   |
| 2   | Ctrl     | 0     | Ctrl read observation<br>Ctrl section has not to be read.  |
|     |          | 1     | Ctrl section has to be read to enable next IF3 update.   |
| 1   | Arb      | 0     | Arbitration data read observation<br>Arbitration data has not to be read.  |
|     |          | 1     | Arbitration data has to be read to enable next IF3 update.   |
| 0   | Mask     | 0     | Mask data read observation<br>Mask data has not to be read.  |
|     |          | 1     | Mask data has to be read to enable next IF3 update.  |

### 23.4.1.24 IF3 Mask Register (DCAN IF3MSK)

The IF3 mask register (DCAN IF3MSK) is shown in [Figure 23-44](#) and described in [Table 23-37](#).

**Figure 23-44. IF3 Mask Register (DCAN IF3MSK)**

| 31   | 30   | 29        | 28        | 16          |
|------|------|-----------|-----------|-------------|
| MXtd | MDir | Reserv ed |           | Msk[28:16]  |
| R-1  | R-1  | R-1       |           | R/WP-0xFFFF |
| 15   |      |           |           | 0           |
|      |      |           | Msk[15:0] |             |
|      |      |           |           | R/WP-0xFFFF |

LEGEND: R = Read only; -n = value after reset

**Table 23-37. IF3 Mask Register (DCAN IF3MSK) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 31   | MXtd      | 0     | Mask Extended Identifier<br>The extended identifier bit (IDE) has no effect on the acceptance filtering.   |
|      |           | 1     | The extended identifier bit (IDE) is used for acceptance filtering.<br>When 11-bit ("standard") identifiers are used for a message object, the identifiers of received data frames are written into bits ID[28:18]. For acceptance filtering, only these bits together with mask bits Msk[28:18] are considered. |
| 30   | MDir      | 0     | Mask Message Direction<br>The message direction bit (Dir) has no effect on the acceptance filtering.   |
|      |           | 1     | The message direction bit (Dir) is used for acceptance filtering.  |
| 29   | Reserved  | 0     | These bits are always read as 0. Writes have no effect.  |
| 28-0 | Msk[28:0] | 0     | Identifier Mask<br>The corresponding bit in the identifier of the message object is not used for acceptance filtering (don't care).  |
|      |           | 1     | The corresponding bit in the identifier of the message object is used for acceptance filtering.  |

### 23.4.1.25 IF3 Arbitration Register (DCAN IF3ARB)

The IF3 arbitration register (DCAN IF3ARB) is shown in [Figure 23-45](#) and described in [Table 23-38](#).

**Figure 23-45. IF3 Arbitration Register (DCAN IF3ARB)**

| 31     | 30  | 29  | 28       | 16        |
|--------|-----|-----|----------|-----------|
| MsgVal | Xtd | Dir |          | ID[28:16] |
| R-0    | R-0 | R-0 |          | R-0       |
| 15     |     |     |          | 0         |
|        |     |     | ID[15:0] |           |
|        |     |     | R-0      |           |

LEGEND: R = Read only; WP = Write protected (protected by busy bit); -n = value after reset

**Table 23-38. IF3 Arbitration Register (DCAN IF3ARB) Field Descriptions**

| Bit  | Field     | Value                 | Description  |
|------|-----------|-----------------------|--|
| 31   | MsgVal    | 0                     | Message Valid<br>The message object is ignored by the message handler.   |
|      |           | 1                     | The message object is to be used by the message handler.<br>The CPU should reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN control register. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required. |
| 30   | Xtd       | 0                     | Extended Identifier<br>The 11-bit ("standard") Identifier is used for this message object.   |
|      |           | 1                     | The 29-bit ("extended") Identifier is used for this message object.  |
| 29   | Dir       | 0                     | Message Direction<br>Direction = receive: On TxRqst, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, this message is stored in this message object.   |
|      |           | 1                     | Direction = transmit: On TxRqst, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set (if RmtEn = 1).   |
| 28-0 | Msk[28:0] | ID[28:0]<br>ID[28:18] | Message Identifier<br>29-bit Identifier ("extended frame")<br>11-bit Identifier ("standard frame")   |

### 23.4.1.26 IF3 Message Control Register (DCAN IF3MCTL)

The IF3 message control register (DCAN IF3MCTL) is shown in [Figure 23-46](#) and described in [Table 23-39](#).

**Figure 23-46. IF3 Message Control Register (DCAN IF3MCTL)**

|         |          |         |       |     |     |        |         |     |          |   |          |   |    |   |
|---------|----------|---------|-------|-----|-----|--------|---------|-----|----------|---|----------|---|----|---|
| 31      | Reserved |         |       |     |     |        |         |     |          |   |          |   | 16 |   |
| R-0     |          |         |       |     |     |        |         |     |          |   |          |   |    |   |
| 15      | 14       | 13      | 12    | 11  | 10  | 9      | 8       | 7   | 6        | 4 | 3        | 2 | 1  | 0 |
| New Dat | Msg Lst  | Int Pnd | UMask | TxE | RxE | Rmt En | Tx Rqst | EoB | Reserved |   | DLC[3:0] |   |    |   |

LEGEND: R = Read only; -n = value after reset

**Table 23-39. IF3 Message Control Register (DCAN IF3MCTL) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 15    | NewDat   | 0     | New Data   |
|       |          | 1     | No new data has been written into the data portion of this message object by the message handler since the last time when this flag was cleared by the CPU.  |
|       |          | 1     | The message handler or the CPU has written new data into the data portion of this message object.  |
| 14    | MsgLst   | 0     | Message Lost (only valid for message objects with direction = receive)   |
|       |          | 1     | No message lost since the last time when this bit was reset by the CPU.  |
|       |          | 1     | The message handler stored a new message into this object when NewDat was still set, so the previous message has been overwritten.   |
| 13    | IntPnd   | 0     | Interrupt Pending  |
|       |          | 1     | This message object is not the source of an interrupt.   |
|       |          | 1     | This message object is the source of an interrupt. The Interrupt Identifier in the interrupt register will point to this message object if there is no other interrupt source with higher priority.                            |
| 12    | UMask    | 0     | Use Acceptance Mask  |
|       |          | 1     | Mask ignored   |
|       |          | 1     | Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering<br>If the UMask bit is set to one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. |
| 11    | TxE      | 0     | Transmit Interrupt enable  |
|       |          | 1     | IntPnd will not be triggered after the successful transmission of a frame.   |
|       |          | 1     | IntPnd will be triggered after the successful transmission of a frame.   |
| 10    | RxE      | 0     | Receive Interrupt enable   |
|       |          | 1     | IntPnd will not be triggered after the successful reception of a frame.  |
|       |          | 1     | IntPnd will be triggered after the successful reception of a frame.  |
| 9     | RmtEn    | 0     | Remote enable  |
|       |          | 1     | At the reception of a remote frame, TxRqst is not changed.   |
|       |          | 1     | At the reception of a remote frame, TxRqst is set.   |
| 8     | TxRqst   | 0     | Transmit Request   |
|       |          | 1     | This message object is not waiting for a transmission.   |
|       |          | 1     | The transmission of this message object is requested and is not yet done.  |

**Table 23-39. IF3 Message Control Register (DCAN IF3MCTL) Field Descriptions (continued)**

| Bit | Field    | Value       | Description  |
|-----|----------|-------------|--|
| 7   | EoB      | 0<br>1      | Data frame has 0-8 data bits.<br>Data frame has 8 data bytes.<br><br>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.<br><br>Note: This bit is used to concatenate multiple message objects to build a FIFO Buffer. For single message objects (not belonging to a FIFO Buffer), this bit must always be set to one. |
| 6-4 | Reserved | 0           | These bits are always read as 0. Writes have no effect.  |
| 3-0 |          | 0-8<br>9-15 | Data Length Code<br>Data frame has 0-8 data bits.<br>Data frame has 8 data bytes.<br><br>Note: The data length code of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it will write the DLC to the value given by the received message.  |

### 23.4.1.27 IF3 Data A and Data B Registers (DCAN IF3DATA/DATB)

The data bytes of CAN messages are stored in the IF3 registers in the following order.

In a CAN data frame, Data 0 is the first, and Data 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

The IF3 data A and data B registers (DCAN IF3DATA/DATB) are shown in [Figure 23-47](#) and [Figure 23-48](#).

**Figure 23-47. IF3 Data A Register (DCAN IF3DATA)**

| 31 | 24     | 23 | 16     |
|----|--------|----|--------|
|    | Data 3 |    | Data 2 |
|    | R-0    |    | R-0    |
| 15 | 8      | 7  | 0      |
|    | Data 1 |    | Data 0 |
|    | R-0    |    | R-0    |

LEGEND: R = Read only; -n = value after reset

**Figure 23-48. IF3 Data A Register (DCAN IF3DATB)**

| 31 | 24     | 23 | 16     |
|----|--------|----|--------|
|    | Data 7 |    | Data 6 |
|    | R-0    |    | R-0    |
| 15 | 8      | 7  | 0      |
|    | Data 5 |    | Data 4 |
|    | R-0    |    | R-0    |

LEGEND: R = Read only; -n = value after reset

### 23.4.1.28 Update Enable Registers (DCAN IF3UPD12 to IF3UPD78)

The automatic update functionality of the IF3 register set can be configured for each message object. A message object is enabled for automatic IF3 update, if the dedicated IF3UpdEn flag is set. This means that an active NewDat flag of this message object (e.g due to reception of a CAN frame) will trigger an automatic copy of the whole message object to IF3 register set.

The update enable registers (DCAN IF3UPD12 to IF3UPD78) are shown in [Table 23-40](#) and described in [Table 23-41](#).

---

**NOTE:** IF3 Update enable should not be set for transmit objects.

---

**Table 23-40. Update Enable Registers (DCAN IF3UPD12 to IF3UPD78))**

| Offset/Register       | 31/<br>15 | 30/<br>14 | 29/<br>13 | 29/<br>12 | 27/<br>11 | 26/<br>10 | 25/<br>9 | 24/<br>8 | 23/<br>7 | 22/<br>6 | 21/<br>5 | 20/<br>4 | 19/<br>3 | 18/<br>2 | 17/<br>1 | 16/<br>0          |
|-----------------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-------------------|
| 0x160<br>CAN IF3UPD12 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[32:17]   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R/W-0             |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[16:1]    |
| 0x164<br>CAN IF3UPD34 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[64:49]   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R/W-0             |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[48:33]   |
| 0x168<br>CAN IF3UPD56 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[96:81]   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R/W-0             |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[80:65]   |
| 0x16C<br>CAN IF3UPD78 |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          |                   |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[128:113] |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R/W-0             |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | IF3UpdEn[112:97]  |
|                       |           |           |           |           |           |           |          |          |          |          |          |          |          |          |          | R/W-0             |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 23-41. Update Enable Registers (DCAN IF3UPD12 to IF3UPD78) Field Descriptions**

| Bit  | Field           | Value | Description  |
|------|-----------------|-------|--|
| 31-0 | IF3UpdEn[128:1] | 0     | IF3 Update Enabled (for all message objects)<br>Automatic IF3 update is disabled for this message object.  |
|      |                 | 1     | Automatic IF3 update is enabled for this message object. A message object is scheduled to be copied to IF3 register set, if NewDat flag of the message object is active. |

### 23.4.1.29 CAN TX I/O Control Register (DCAN TIOC)

The CAN\_TX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed. The CAN TX I/O control register (DCAN TIOC) is shown in [Figure 23-49](#) and described in [Table 23-42](#).

---

**NOTE:** The values of the IO control registers are only writable if Init bit of the CAN control register is set.

---



---

**NOTE:** The OD, Func, Dir and Out bits of the CAN TX IO control register are forced to certain values when Init bit of CAN control register is reset (see bit descriptions).

---

**Figure 23-49. CAN TX I/O Control Register (DCAN TIOC)**

| 31 | Reserved | 19     | 18     | 17     | 16  |
|----|----------|--------|--------|--------|-----|
|    | R-0      | PU     | PD     | OD     |     |
|    |          | R/W-D  | R/W-D  | R/WP-0 |     |
| 15 | Reserved | 3      | 2      | 1      | 0   |
|    | R-0      | Func   | Dir    | Out    | In  |
|    |          | R/WP-0 | R/WP-0 | R/WP-0 | R-U |

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Init bit), U = Undefined; D = Device dependent; -n = value after reset

**Table 23-42. CAN TX I/O Control Register (DCAN TIOC) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 31-19 | Reserved | 0      | These bits are always read as 0. Writes have no effect.  |
| 18    | PU       | 0<br>1 | CAN_TX pull up/pull down select. This bit is only active when CAN_TX is configured to be an input.<br>CAN_TX pull down is selected, when pull logic is active (PD = 0).<br>CAN_TX pull up is selected, when pull logic is active(PD = 0).  |
| 17    | PD       | 0<br>1 | CAN_TX pull disable. This bit is only active when CAN_TX is configured to be an input.<br>CAN_TX pull is active<br>CAN_TX pull is disabled   |
| 16    | OD       | 0<br>1 | CAN_TX open drain enable. This bit is only active when CAN_TX is configured to be in GIO mode (TIOC.Func=0).<br>The CAN_TX pin is configured in push/pull mode.<br>The CAN_TX pin is configured in open drain mode.<br>Forced to '0' if Init bit of CAN control register is reset. |
| 15-4  | Reserved | 0      | These bits are always read as 0. Writes have no effect.  |
| 3     | Func     | 0<br>1 | CAN_TX function. This bit changes the function of the CAN_TX pin<br>CAN_TX pin is in GIO mode.<br>CAN_TX pin is in functional mode (as an output to transmit CAN data).<br>Forced to '1' if Init bit of CAN control register is reset.   |
| 2     | Dir      | 0<br>1 | CAN_TX data direction. This bit controls the direction of the CAN_TX pin when it is configured to be in GIO mode only (TIOC.Func=0)<br>The CAN_TX pin is an input.<br>The CAN_TX pin is an output<br>Forced to '1' if Init bit of CAN control register is reset.                   |

**Table 23-42. CAN TX I/O Control Register (DCAN TIOC) Field Descriptions (continued)**

| Bit | Field | Value  | Description  |
|-----|-------|--------|--|
| 1   | Out   | 0<br>1 | CAN_TX data out write. This bit is only active when CAN_TX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_TX pin.<br><br>0 The CAN_TX pin is driven to logic low<br>1 The CAN_TX pin is driven to logic high<br><br>Forced to Tx output of the CAN core, if Init bit of CAN control register is reset. |
| 0   | In    | 0<br>1 | CAN_TX data in<br><br>0 The CAN_TX pin is at logic low<br>1 The CAN_TX pin is at logic high<br><br>Note: When CAN_TX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (e.g. while reset of the DCAN module).   |

### 23.4.1.30 CAN RX IO Control Register (DCAN RIOC)

The CAN\_RX pin of the DCAN module can be used as general purpose IO pin if CAN function is not needed. The CAN RX I/O control register (DCAN RIOC) is shown in [Figure 23-50](#) and described in [Table 23-43](#).

**NOTE:** The values of the IO control registers are writable only if Init bit of CAN control register is set.

**NOTE:** The OD, Func and Dir bits of the CAN RX IO control register are forced to certain values when the Init bit of CAN control register is reset (see bit descriptions).

**Figure 23-50. CAN RX IO control register (DCAN RIOC)**

| 31 | Reserved | 19     | 18     | 17     | 16     |
|----|----------|--------|--------|--------|--------|
|    | R-0      |        | PU     | PD     | OD     |
|    |          |        | R/W-D  | R/W-D  | R/WP-0 |
| 15 | Reserved | 3      | 2      | 1      | 0      |
|    | R-0      | Func   | Dir    | Out    | In     |
|    |          | R/WP-0 | R/WP-0 | R/WP-0 | R-U    |

LEGEND: R/W = Read/Write; R = Read only; WP = Protected Write (protected by Init bit), U = Undefined; D = Device dependent; -n = value after reset

**Table 23-43. CAN RX IO Control Register (DCAN RIOC) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-19 | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 18    | PU       | 0     | CAN_RX pull up/pull down select. This bit is only active when CAN_RX is configured to be an input.<br>CAN_RX pull down is selected, when pull logic is active (PD = 0).<br>CAN_T=RX pull up is selected, when pull logic is active(PD = 0).  |
| 17    | PD       | 0     | CAN_RX pull disable. This bit is only active when CAN_TX is configured to be an input.<br>CAN_RX pull is active<br>CAN_RX pull is disabled   |
| 16    | OD       | 0     | CAN_RX open drain enable. This bit is only active when CAN_RX is configured to be in GIO mode (TIOC.Func=0).<br>The CAN_RX pin is configured in push/pull mode.<br>The CAN_RX pin is configured in open drain mode.<br>Forced to '0' if Init bit of CAN control register is reset. |
| 15-4  | Reserved | 0     | These bits are always read as 0. Writes have no effect.  |
| 3     | Func     | 0     | CAN_RX function. This bit changes the function of the CAN_RX pin<br>CAN_RX pin is in GIO mode.<br>CAN_RX pin is in functional mode (as an output to transmit CAN data).<br>Forced to '1' if Init bit of CAN control register is reset.   |
| 2     | Dir      | 0     | CAN_RX data direction. This bit controls the direction of the CAN_RX pin when it is configured to be in GIO mode only (TIOC.Func=0)<br>The CAN_RX pin is an input.<br>The CAN_RX pin is an output<br>Forced to '1' if Init bit of CAN control register is reset.                   |

**Table 23-43. CAN RX IO Control Register (DCAN RIOC) Field Descriptions (continued)**

| Bit | Field | Value  | Description  |
|-----|-------|--------|--|
| 1   | Out   | 0<br>1 | CAN_RX data out write. This bit is only active when CAN_RX pin is configured to be in GIO mode (TIOC.Func = 0) and configured to be an output pin (TIOC.Dir = 1). The value of this bit indicates the value to be output to the CAN_RX pin.<br><br>0 The CAN_RX pin is driven to logic low<br>1 The CAN_RX pin is driven to logic high<br><br>Forced to Tx output of the CAN core, if Init bit of CAN control register is reset. |
| 0   | In    | 0<br>1 | CAN_RX data in<br><br>0 The CAN_RX pin is at logic low<br>1 The CAN_RX pin is at logic high<br><br>Note: When CAN_RX pin is connected to a CAN transceiver, an external pullup resistor has to be used to ensure that the CAN bus will not be disturbed (e.g. while reset of the DCAN module).   |

## ***Multichannel Serial Port Interface (McSPI)***

This chapter describes the McSPI of the device.

| Topic                             | Page |
|-----------------------------------|------|
| 24.1 Introduction .....           | 3947 |
| 24.2 Integration .....            | 3947 |
| 24.3 Functional Description ..... | 3950 |
| 24.4 McSPI Registers .....        | 3985 |

## 24.1 Introduction

This document is intended to provide programmers with a functional presentation of the Master/Slave Multichannel Serial Port Interface (McSPI) module. It also provides a register description and a module configuration example.

McSPI is a general-purpose receive/transmit master/slave controller that can interface with up to four slave external devices or one single external master. It allows a duplex, synchronous, serial communication between a CPU and SPI compliant external devices (Slaves and Masters).

### 24.1.1 McSPI Features

The general features of the SPI controller are:

- Buffered receive/transmit data register per channel (1 word deep)
- Multiple SPI word access with one channel using a FIFO
- Two DMA requests per channel, one interrupt line
- Single interrupt line, for multiple interrupt source events
- Serial link interface supports:
  - Full duplex / Half duplex
  - Multi-channel master or single channel slave operations
  - Programmable 1-32 bit transmit/receive shift operations.
  - Wide selection of SPI word lengths continuous from 4 to 32 bits
- Up to four SPI channels
- SPI word Transmit / Receive slot assignment based on round robin arbitration
- SPI configuration per channel (clock definition, enable polarity and word width)
- Clock generation supports:
  - Programmable master clock generation (operating from fixed 48-MHz functional clock input)
  - Selectable clock phase and clock polarity per chip select.

### 24.1.2 Unsupported McSPI Features

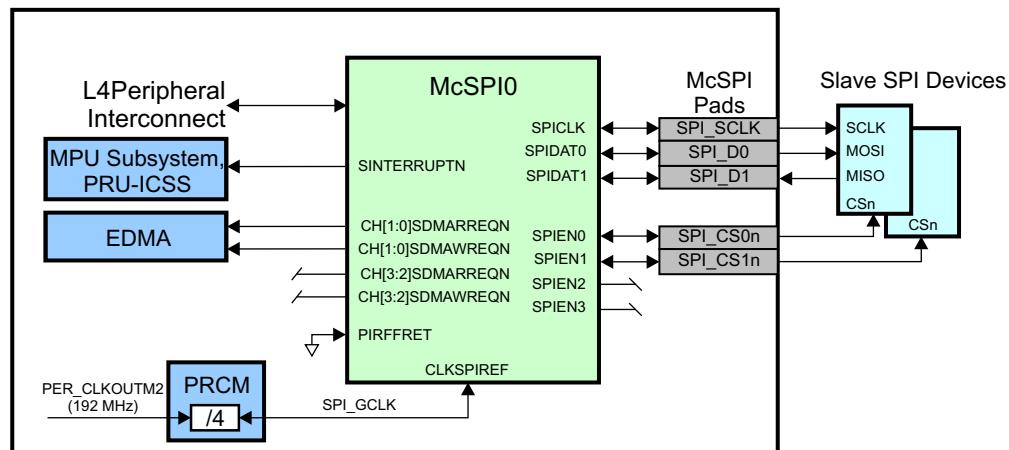
This device supports only two chip selects per module. Module wakeup during slave mode operation is not supported, as noted in *McSPI Clock and Reset Management*.

**Table 24-1. Unsupported McSPI Features**

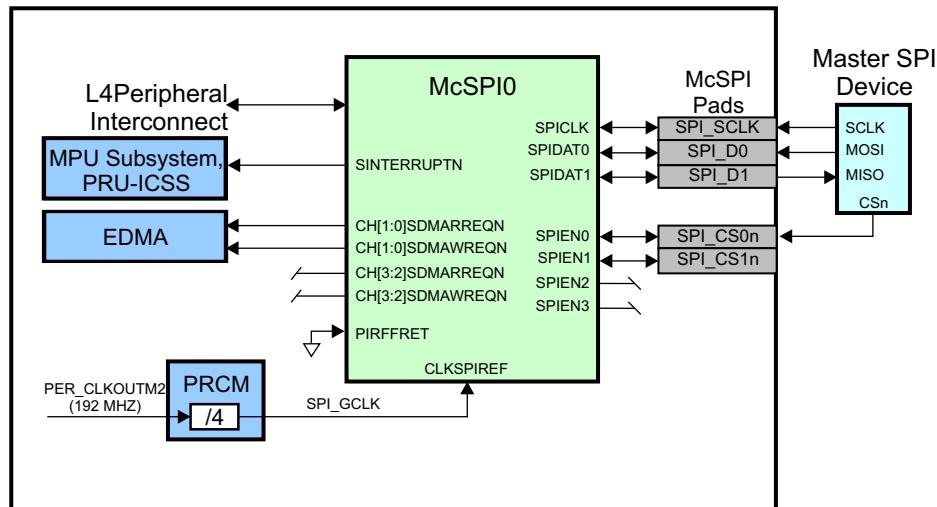
| Feature                     | Reason  |
|-----------------------------|---|
| Chip selects 2 and 3        | Not pinned out                                |
| Slave mode wakeup           | SWAKEUP not connected                         |
| Retention during power down | Module not synthesized with retention enabled |

## 24.2 Integration

This device includes two instantiations of McSPI: SPI0 and SPI1. The McSPI module is a general-purpose receive/transmit master/slave controller that can interface with either up to four slave external devices or one single external master. [Figure 24-1](#) shows the example of a system with multiple external slave SPI compatible devices and [Figure 24-2](#) shows the example of a system with an external master.



**Figure 24-1. SPI Master Application**



**Figure 24-2. SPI Slave Application**

### 24.2.1 McSPI Connectivity Attributes

The general connectivity attributes for the McSPI module are shown in [Table 24-2](#).

**Table 24-2. McSPI Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L4LS_GCLK (Interface/OCP)<br>PD_PER_SPI_GCLK (Func)   |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | 1 interrupt to MPU subsystem and PRU-ICSS (McSPI0INT)<br>1 interrupt to MPU subsystem only (McSPI1INT)   |
| DMA Requests        | 4 DMA requests per instance to EDMA <ul style="list-style-type: none"> <li>• 1 RX request for CS0 (SPIREVT0)</li> <li>• 1 TX request for CS0 (SPIXEVT0)</li> <li>• 1 RX request for CS1 (SPIREVT1)</li> <li>• 1 TX request for CS1 (SPIXEVT1)</li> </ul> |
| Physical Address    | L4 Peripheral slave port   |

### 24.2.2 McSPI Clock and Reset Management

The SPI module clocks can be woken up in two manners: by the SPI module itself using the SWAKEUP signal (refer to the module functional spec for detailed conditions), or directly from an external SPI master device by detecting an active low level on its chip select input pin (CS0n) using a GPIO attached to that device pin. Neither of these methods is supported on the device.

**Table 24-3. McSPI Clock Signals**

| Clock Signal                  | Max Freq | Reference / Source | Comments                      |
|-------------------------------|----------|--------------------|-------------------------------|
| CLK<br>Interface clock        | 100 MHz  | CORE_CLKOUTM4 / 2  | pd_per_l4ls_gclk<br>From PRCM |
| CLKSPIREF<br>Functional clock | 48 MHz   | PER_CLKOUTM2 / 4   | pd_per_spi_gclk<br>From PRCM  |

### 24.2.3 McSPI Pin List

The McSPI interface pins are summarized in [Table 24-4](#).

**Table 24-4. McSPI Pin List**

| Pin       | Type | Description   |
|-----------|------|---|
| SPIx_SCLK | I/O  | SPI serial clock (output when master, input when slave)             |
| SPIx_D0   | I/O  | Can be configured as either input or output (MOSI or MISO)          |
| SPIx_D1   | I/O  | Can be configured as either input or output (MOSI or MISO)          |
| SPIx_CS0  | I/O  | SPI chip select 0 output when master, input when slave (active low) |
| SPIx_CS1  | O    | SPI chip select 1 output when master, input when slave (active low) |

## 24.3 Functional Description

### 24.3.1 SPI Transmission

This section describes the transmissions supported by McSPI. The SPI protocol is a synchronous protocol that allows a master device to initiate serial communication with a slave device. Data is exchanged between these devices. A slave select line (SPIEN) can be used to allow selection of an individual slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. Connected to multiple external devices, McSPI exchanges data with a single SPI device at a time through two main modes:

- Two data pins interface mode. (See [Section 24.3.1.1](#))
- Single data pin interface mode (recommended for half-duplex transmission). (See [Section 24.3.1.2](#))

The flexibility of McSPI allows exchanging data with several formats through programmable parameters described in [Section 24.3.1.3](#).

### 24.3.1.1 Two Data Pins Interface Mode

The two data pins interface mode, allows a full duplex SPI transmission where data is transmitted (shifted out serially) and received (shifted in serially) simultaneously on separate data lines SPIDAT [0] and SPIDAT [1]. Data leaving the master exits on transmit serial data line also known as MOSI: MasterOutSlaveIn. Data leaving the slave exits on the receive data line also known as MISO: MasterInSlaveOut.

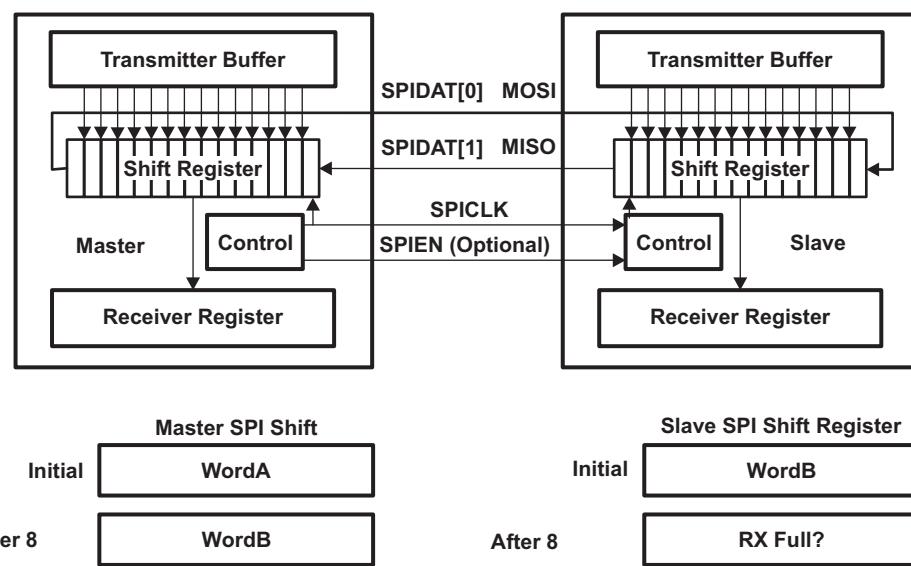
McSPI has a unified SPI port control: SPIDAT [1:0] can be independently configured as receive or transmit lines. The user has the responsibility to program which data line to use and in which direction (receive or transmit), according to the external slave/master connection.

The serial clock (SPICLK) synchronizes shifting and sampling of the information on the two serial data lines (SPIDAT [1:0]). Each time a bit is transferred out from the Master, one bit is transferred in from Slave.

[Figure 24-3](#) shows an example of a full duplex system with a Master device on the left and a Slave device on the right. After 8 cycles of the serial clock SPICLK, the WordA has been transferred from the master to the slave. At the same time, the 8-bit WordB has been transferred from the slave to the master.

When referring to the master device, the control block transmits the clock SPICLK and the enable signal SPIEN (optional, see [Section 24.4.1.7, McSPI\\_MODULECTRL](#)).

**Figure 24-3. SPI Full-Duplex Transmission**



### 24.3.1.2 Single Data Pin Interface Mode

In single data pin interface mode, under software control, a single data line is used to alternatively transmit and receive data (Half duplex transmission).

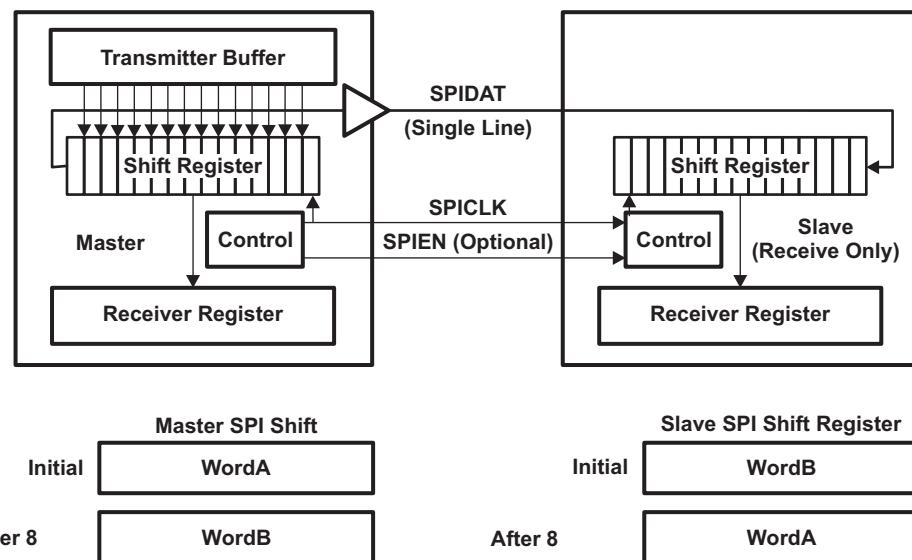
McSPI has a unified SPI port control: SPIDAT [1:0] can be independently configured as receive or transmit lines. The user has the responsibility to program which data line to use and in which direction (receive or transmit), according to the external slave/master connection.

As for a full duplex transmission, the serial clock (SPICLK) synchronizes shifting and sampling of the information on the single serial data line.

#### 24.3.1.2.1 Example With a Receive-Only Slave

Figure 24-4 shows a half duplex system with a Master device on the left and a receive-only Slave device on the right. Each time a bit is transferred out from the Master, one bit is transferred in the Slave. After 8 cycles of the serial clock SPICLK, the 8-bit WordA has been transferred from the master to the slave.

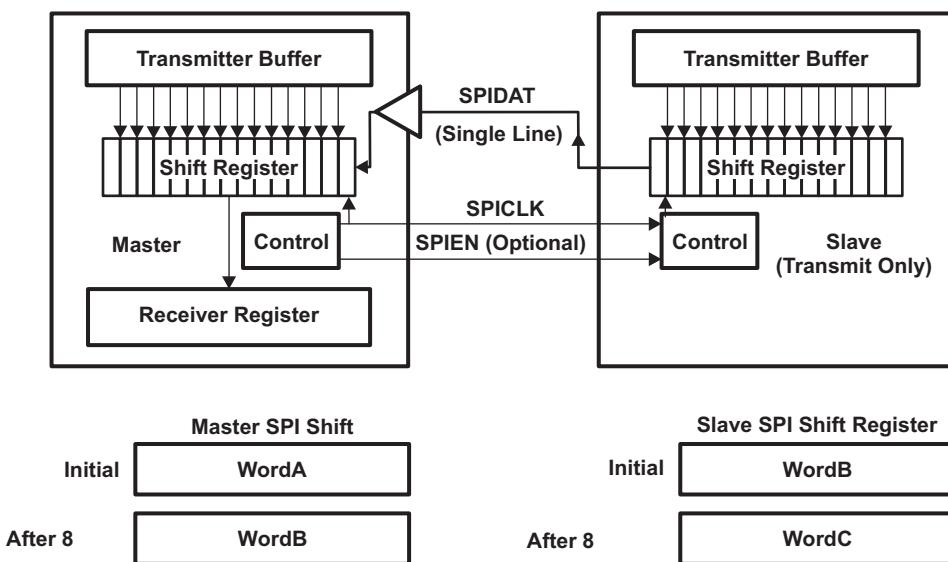
**Figure 24-4. SPI Half-Duplex Transmission (Receive-only Slave)**



#### 24.3.1.2.2 Example With a Transmit-Only Slave

Figure 24-5 shows a half duplex system with a Master device on the left and a transmit-only Slave device on the right. Each time a bit is transferred out from the Slave, one bit is transferred in the Master. After 8 cycles of the serial clock SPICLK, the 8-bit WordA has been transferred from the slave to the master.

**Figure 24-5. SPI Half-Duplex Transmission (Transmit-Only Slave)**



### 24.3.1.3 Transfer Formats

This section describes the transfer formats supported by McSPI.

The flexibility of McSPI allows setting the parameters of the SPI transfer:

- SPI word length
- SPI enable generation programmable
- SPI enable assertion
- SPI enable polarity
- SPI clock frequency
- SPI clock phase
- SPI clock polarity

The consistency between SPI word length, clock phase and clock polarity of the master SPI device and the communicating slave device remains under software responsibility.

#### 24.3.1.3.1 Programmable Word Length

McSPI supports any SPI word from 4 to 32 bits long.

The SPI word length can be changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

#### 24.3.1.3.2 Programmable SPI Enable Generation

McSPI is able to generate or not the SPI enable, if management of chip select is de-asserted a point to point connection is mandatory. Only a single master or slave device can be connected to the SPI bus.

#### 24.3.1.3.3 Programmable SPI Enable (SPIEN)

The polarity of the SPIEN signals is programmable. SPIEN signals can be active high or low.

The assertion of the SPIEN signals is programmable: SPIEN signals can be manually asserted or can be automatically asserted.

Two consecutive words for two different slave devices may go along with active SPIEN signals with different polarity.

#### 24.3.1.3.4 Programmable SPI Clock (SPICLK)

The phase and the polarity of the SPI serial clock are programmable when McSPI is a SPI master device or a SPI slave device. The baud rate of the SPI serial clock is programmable when McSPI is a SPI master.

When McSPI is operating as a slave, the serial clock SPICLK is an input from the master.

#### 24.3.1.3.5 Bit Rate

In Master Mode, an internal reference clock CLKSPIREF is used as an input of a programmable divider to generate bit rate of the serial clock SPICLK. Granularity of this clock divider can be changed.

#### 24.3.1.3.6 Polarity and Phase

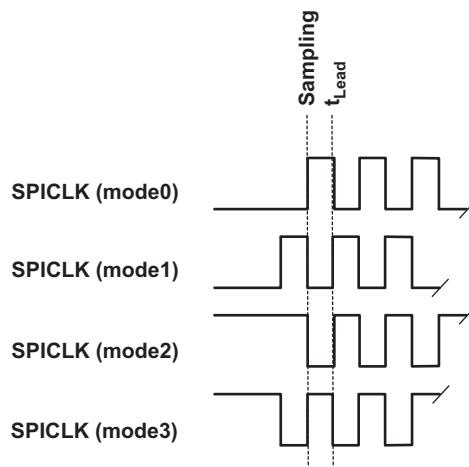
McSPI supports four sub-modes of the SPI format transfer that depend on the polarity (POL) and the phase (PHA) of the SPI serial clock (SPICLK). [Table 24-5](#) and [Figure 24-6](#) show a summary of the four sub-modes. Software selects one of four combinations of serial clock phase and polarity.

Two consecutive SPI words for two different slave devices may go along with active SPICLK signal with different phase and polarity.

**Table 24-5. Phase and Polarity Combinations**

| Polarity (POL) | Phase (PHA) | SPI Mode | Comments  |
|----------------|-------------|----------|---|
| 0              | 0           | mode0    | SPICLK active high and sampling occurs on the rising edge.  |
| 0              | 1           | mode1    | SPICLK active high and sampling occurs on the falling edge. |
| 1              | 0           | mode2    | SPICLK active low and sampling occurs on the falling edge.  |
| 1              | 1           | mode3    | SPICLK active low and sampling occurs on the rising edge.   |

**Figure 24-6. Phase and Polarity Combinations**



#### 24.3.1.3.7 Transfer Format With PHA = 0

This section describes the concept of a SPI transmission with the SPI mode0 and the SPI mode2.

In the transfer format with PHA = 0, SPIEN is activated a half cycle of SPICLK ahead of the first SPICLK edge.

In both master and slave modes, McSPI drives the data lines at the time of SPIEN is asserted.

Each data frame is transmitted starting with the MSB. At the extremity of both SPI data lines, the first bit of SPI word is valid a half-cycle of SPICLK after the SPIEN assertion.

Therefore, the first edge of the SPICLK line is used by the master to sample the first data bit sent by the slave. On the same edge, the first data bit sent by the master is sampled by the slave.

On the next SPICLK edge, the received data bit is shifted into the shift register, and a new data bit is transmitted on the serial data line.

This process continues for a total of pulses on the SPICLK line defined by the SPI word length programmed in the master device, with data being latched on odd numbered edges and shifted on even numbered edges.

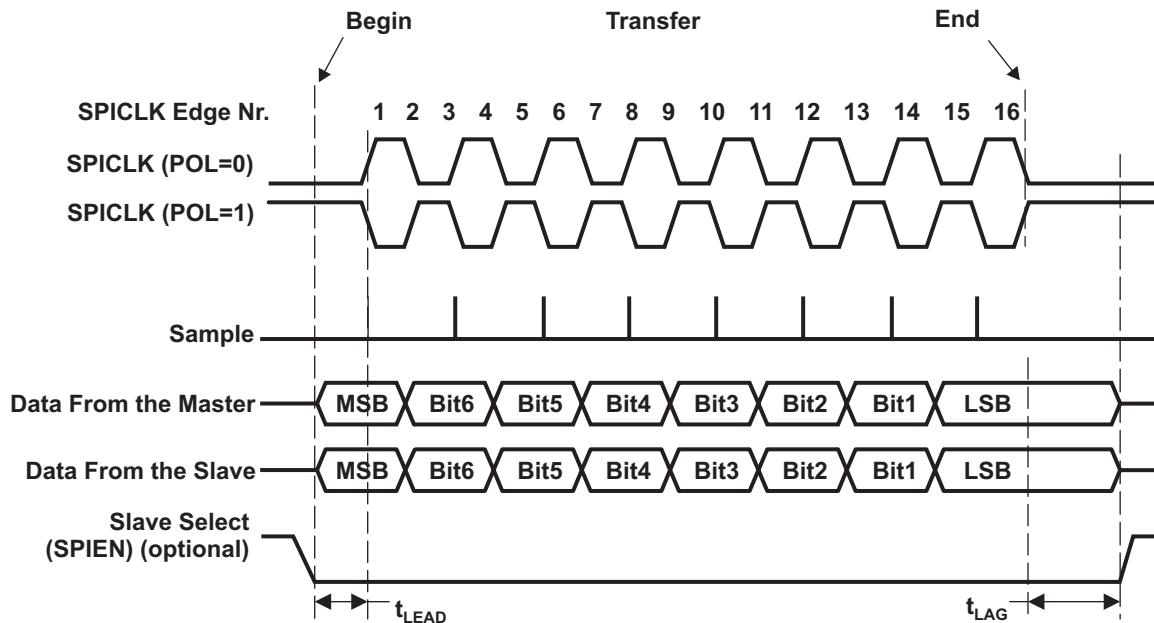
**Figure 24-7** is a timing diagram of a SPI transfer for the SPI mode0 and the SPI mode2, when McSPI is master or slave, with the frequency of SPICLK equals to the frequency of CLKSPIREF. It should not be used as a replacement for SPI timing information and requirements detailed in the data manual.

When McSPI is in slave mode, if the SPIEN line is not de-asserted between successive transmissions then the content of the Transmitter register is not transmitted, instead the last received SPI word is transmitted.

In master mode, the SPIEN line must be negated and reasserted between each successive SPI word. This is because the slave select pin freezes the data in its shift register and does not allow it to be altered if PHA bit equals 0.

In 3-pin mode without using the SPIEN signal, the controller provides the same waveform but with SPIEN forced to low state. In slave mode SPIEN is useless

**Figure 24-7. Full Duplex Single Transfer Format with PHA = 0**



#### 24.3.1.3.8 Transfer Format With PHA = 1

This section describes SPI full duplex transmission with the SPI mode1 and the SPI mode3.

In the transfer format with PHA = 1, SPIEN is activated a delay ( $t_{\text{Lead}}$ ) ahead of the first SPICLK edge.

In both master and slave modes, McSPI drives the data lines on the first SPICLK edge.

Each data frame is transmitted starting with the MSB. At the extremity of both SPI data lines, the first bit of SPI word is valid on the next SPICLK edge, a half-cycle later of SPICLK. It is the sampling edge for both the master and slave.

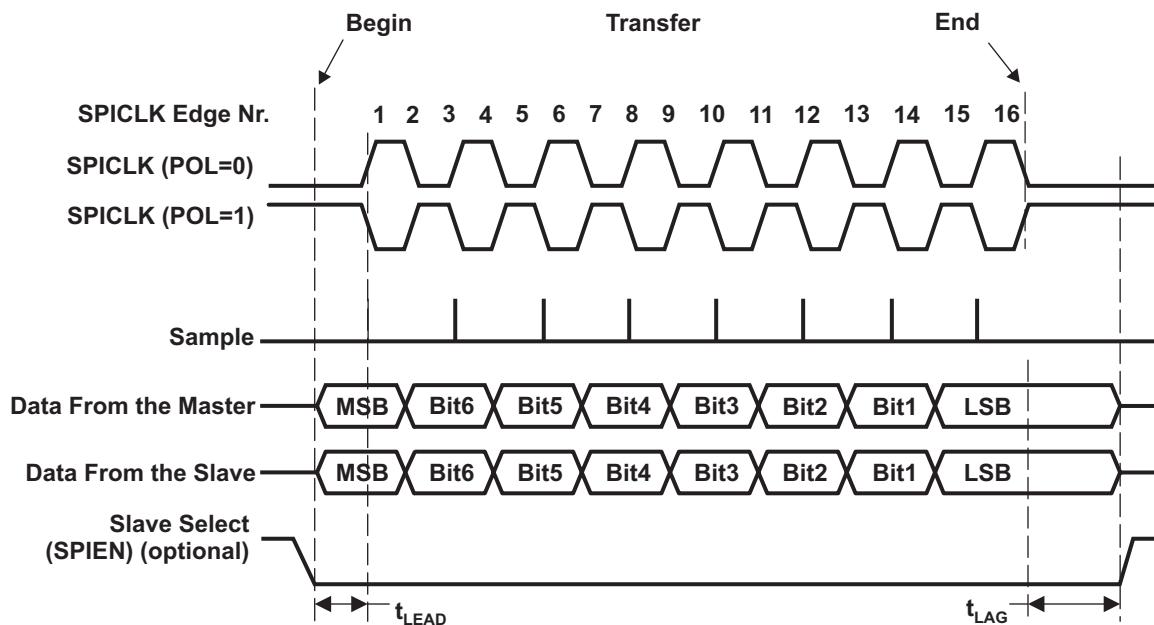
When the third edge occurs, the received data bit is shifted into the shift register. The next data bit of the master is provided to the serial input pin of the slave.

This process continues for a total of pulses on the SPICLK line defined by the word length programmed in the master device, with data being latched on even numbered edges and shifted on odd numbered edges.

[Figure 24-8](#) is a timing diagram of a SPI transfer for the SPI mode1 and the SPI mode3, when McSPI is master or slave, with the frequency of SPICLK equals to the frequency of CLKSPIREF. It should not be used as a replacement for SPI timing information and requirements detailed in the data manual.

The SPIEN line may remain active between successive transfers. In 3-pin mode without using the SPIEN signal, the controller provides the same waveform but with SPIEN forced to low state. In slave mode SPIEN is useless.

**Figure 24-8. Full Duplex Single Transfer Format With PHA = 1**



## 24.3.2 Master Mode

McSPI is in master mode when the bit MS of the register MCSPI\_MODULCTRL is cleared.

In master mode McSPI supports multi-channel communication with up to 4 independent SPI communication channel contexts. McSPI initiates a data transfer on the data lines (SPIDAT [1;0]) and generates clock (SPICLK) and control signals (SPIEN) to a single SPI slave device at a time.

### 24.3.2.1 Dedicated Resources Per Channel

In the following sections, the letter "I" indicates the channel number that can be 0, 1, 2 or 3. Each channel has the following dedicated resources:

- Its own channel enable, programmable with the bit EN of the register MCSPI\_CH(I)CTRL. Disabling the channel, outside data word transmission, remains under user responsibility.
- Its own transmitter register MCSPI\_TX on top of the common shift register. If the transmitter register is empty, the status bit TXS of the register MCSPI\_CH(I)STAT is set.
- Its own receiver register MCSPI\_RX on top of the common shift register. If the receiver register is full, the status bit RXS of the register MCSPI\_CH(I)STAT is set.
- A fixed SPI ENABLE line allocation (SPIEN[i] port for channel "I"), SPI enable management is optional.
- Its own communication configuration with the following parameters via the register (I)CONF
  - Transmit/Receive modes, programmable with the bit TRM.
  - Interface mode (Two data pins or Single data pin) and data pins assignment, both programmable with the bits IS and DPE.
  - SPI word length, programmable with the bits WL.
  - SPIEN polarity, programmable with the bit EPOL.
  - SPIEN kept active between words, programmable with the bit FORCE.
  - Turbo mode, programmable with the bit TURBO.
  - SPICLK frequency, programmable with the bit CLKD, the granularity of clock division can be changed using CLKG bit, the clock ratio is then concatenated with MCSPI\_CHCTRL[EXTCLK] value.
  - SPICLK polarity, programmable with the bit POL
  - SPICLK phase, programmable with the bit PHA.
  - Start bit polarity, programmable with the bit SBPOL
  - Use a FIFO Buffer or not (see the following note), programmable with FFER and FFEW, depending on transfer mode, (MCSPI\_CH(I)CONF[TRM]).
- Two DMA requests events, read and write, to synchronize read/write accesses of the DMA controller with the activity of McSPI. The DMA requests are enabled with the bits DMAR and DMAW.
- Three interrupts events

**Note:** When more than one channel has an FIFO enable bit field (FFER or FFEW) set, the FIFO will not be used on any channel. Software must ensure that only one enabled channel is configured to use the FIFO buffer.

The transfers will use the latest loaded parameters of the register MCSPI\_CH(I)CONF.

The configuration parameters SPIEN polarity, Turbo mode, SPICLK phase and SPICLK polarity can be loaded in the MCSPI\_CH(I)CONF register only when the channel is disabled. The user has the responsibility to change the other parameters of the MCSPI\_CH(I)CONF register when no transfer occurs on the SPI interface.

### 24.3.2.2 Interrupt Events in Master Mode

In master mode, the interrupt events related to the transmitter register state are TX\_empty and TX\_underflow. The interrupt event related to the receiver register state is RX\_full.

#### 24.3.2.2.1 TX\_empty

The event TX\_empty is activated when a channel is enabled and its transmitter register becomes empty (transient event). Enabling channel automatically raises this event, except for the Master receive only mode. (See [Section 24.3.2.5](#)). When the FIFO buffer is enabled (MCSPI\_CH(I)CONF[FFEW] set to 1), the TX\_empty is asserted as soon as there is enough space in the buffer to write a number of bytes defined by MCSPI\_XFERLEVEL[AEL].

Transmitter register must be loaded to remove the source of the interrupt and the TX\_empty interrupt status bit must be cleared for interrupt line de-assertion (if event enabled as interrupt source) . (See [Section 24.3.4](#)).

When FIFO is enabled, no new TX\_empty event will be asserted as soon as CPU has not performed the number of write into transmitter register defined by MCSPI\_XFERLEVEL[AEL]. It is the responsibility of CPU to perform the right number of writes.

#### 24.3.2.2.2 TX\_underflow

The event TX\_underflow is activated when the channel is enabled and if the transmitter register or FIFO is empty (not updated with new data) at the time of shift register assignment.

The TX\_underflow is a harmless warning in master mode.

To avoid having TX\_underflow event at the beginning of a transmission, the event TX\_underflow is not activated when no data has been loaded into the transmitter register since channel has been enabled.

To avoid having a TX\_underflow event, the Transmit Register (MCSPI\_TX(i)) should be loaded as infrequently as possible.

TX\_underflow interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

**Note:** When more than one channel has an FIFO enable bit field (FFER or FFEW) set, the FIFO will not be used on any channel. Software must ensure that only one enabled channel is configured to use the FIFO buffer.

#### 24.3.2.2.3 RX\_full

The event RX\_full is activated when channel is enabled and receiver register becomes filled (transient event). When FIFO buffer is enabled (MCSPI\_CH(I)CONF[FFER] set to 1), the RX\_full is asserted when the number of bytes in the buffer equals the level defined by MCSPI\_XFERLEVEL[AFL].

Receiver register must be read to remove source of interrupt and RX\_full interrupt status bit must be cleared for interrupt line de-assertion (if event enabled as interrupt source).

When the FIFO is enabled, no new RX\_FULL event will be asserted once the CPU has read the number of bytes defined by MCSPI\_XFERLEVEL[AFL]. It is the responsibility of the CPU to perform the correct number of read operations.

#### 24.3.2.2.4 End of Word Count

The event end of word (EOW) count is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in MCSPI\_XFERLEVEL[WCNT] register. If the value was programmed to 0000h, the counter is not enabled and this interrupt is not generated.

The EOW count interrupt also indicates that the SPI transfer has halted on the channel using the FIFO buffer.

The EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

#### 24.3.2.3 Master Transmit and Receive Mode

This mode is programmable per channel (bit TRM of register MCSPI\_CH(I)CONF).

The channel access to the shift registers, for transmission/reception, is based on its transmitter and receiver register state and round robin arbitration.

The channel that meets the rules below is included in the round robin list of already active channels scheduled for transmission and/or reception. The arbiter skips the channel that does not meet the rules and search for the next following enabled channel, in rotation.

**Rule 1:** Only enabled channels (bit EN of the register MCSPI\_CH(I)CTRL), can be scheduled for transmission and/or reception.

**Rule 2:** An enabled channel can be scheduled if its transmitter register is not empty (bit TXS of the register MCSPI\_CH(I)STAT) or its FIFO is not empty when the buffer is used for the corresponding channel (bit FFE of the register MCSPI\_CH(I)STAT) at the time of shift register assignment. If the transmitter register or FIFO is empty, at the time of shift register assignment, the event TX\_underflow is activated and the next enabled channel with new data to transmit is scheduled. (See also transmit only mode).

**Rule 3:** An enabled channel can be scheduled if its receive register is not full (bit RXS of the register MCSPI\_CH(I)STAT) or its FIFO is not full when the buffer is used for the corresponding channel (bit FFF of the register MCSPI\_CH(I)STAT) at the time of shift register assignment. (See also receive only mode). Therefore the receiver register of FIFO cannot be overwritten. The RX\_overflow bit, in the MCSPI\_IRQSTATUS register is never set in this mode.

On completion of SPI word transfer (bit EOT of the register MCSPI\_CH(I)STAT is set) the updated transmitter register for the next scheduled channel is loaded into the shift register. This bit is meaningless when using the Buffer for this channel. The serialization (transmit and receive) starts according to the channel communication configuration. On serialization completion the received data is transferred to the channel receive register.

The built-in FIFO is available in this mode and if configured in one data direction, transmit or receive, then the FIFO is seen as a unique 64-byte buffer. If configured in both data directions, transmit and receive, then the FIFO is split into two separate 32-byte buffers with their own address space management. In this last case, the definition of AEL and AFL levels is based on 32 bytes and is under CPU responsibility.

#### 24.3.2.4 Master Transmit-Only Mode

This mode eliminates the need for the CPU to read the receiver register (minimizing data movement) when only transmission is meaningful.

The master transmit only mode is programmable per channel (bits TRM of the register MCSPI\_CH(I)CONF).

In master transmit only mode, transmission starts after data is loaded into the transmitter register.

**Rule 1** and **Rule 2**, defined above, are applicable in this mode.

**Rule 3**, defined above, is not applicable: In master transmit only mode, the receiver register or FIFO state "full" does not prevent transmission, and the receiver register is always overwritten with the new SPI word. This event in the receiver register is not significant when only transmission is meaningful. So, the RX\_overflow bit, in the MCSPI\_IRQSTATUS register is never set in this mode.

The McSPI module automatically disables the RX\_full interrupt status. The corresponding interrupt request and DMA Read request are not generated in master transmit only mode.

The status of the serialization completion is given by the bit EOT of the register MCSPI\_CH(I)STAT. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFEW bit field in the MCSPI\_CH(I)CONF register, then the FIFO is seen as a unique 64-byte buffer.

### 24.3.2.5 Master Receive-Only Mode

This mode eliminates the need for the CPU to refill the transmitter register (minimizing data movement) when only reception is meaningful.

The master receive mode is programmable per channel (bits TRM of the register MCSPI\_CH(I)CONF).

The master receive only mode enables channel scheduling only on empty state of the receiver register.

**Rule 1** and **Rule 3**, defined above, are applicable in this mode.

**Rule 2**, defined above, is not applicable: In master receive only mode, after the first loading of the transmitter register of the enabled channel, the transmitter register state is maintained as full. The content of the transmitter register is always loaded into the shift register, at the time of shift register assignment. So, after the first loading of the transmitter register, the bits TX\_empty and TX\_underflow, in the MCSPI\_IRQSTATUS register are never set in this mode.

The status of the serialization completion is given by the bit EOT of the register MCSPI\_CH(I)STAT. The bit RX\_full in the MCSPI\_IRQSTATUS register is set when a received data is loaded from the shift register to the receiver register. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI\_CH(I)CONF register, then the FIFO is seen as a unique 64-byte buffer.

### 24.3.2.6 Single-Channel Master Mode

When the SPI is configured as a master device with a single enabled channel, the assertion of the SPIM\_CSX signal can be controlled in two different ways:

- In 3 pin mode : MCSPI\_MODULCTRL[1] PIN34 and MCSPI\_MODULCTRL[0] SINGLE bit are set to 1, the controller transmit SPI word as soon as transmit register or FIFO is not empty.
- In 4 pin mode : MCSPI\_MODULCTRL[1] PIN34 bit is cleared to 0 and MCSPI\_MODULCTRL[0] SINGLE bit is set to 1, SPIEN assertion/deassertion controlled by Software. (See [Section 24.3.2.6.1](#) using the MCSPI\_CHxCONF[20] FORCE bit.)

#### 24.3.2.6.1 Programming Tips When Switching to Another Channel

When a single channel is enabled and data transfer is ongoing:

- Wait for completion of the SPI word transfer (bit EOT of the register MCSPI\_CH(I)STAT is set) before disabling the current channel and enabling a different channel.
- Disable the current channel first, and then enable the other channel.

#### 24.3.2.6.2 Keep SPIEN Active Mode (Force SPIEN)

Continuous transfers are manually allowed by keeping the SPIEN signal active for successive SPI words transfer. Several sequences (configuration/enable/disable of the channel) can be run without deactivating the SPIEN line. This mode is supported by all channels and any master sequence can be used (transmit-receive, transmit-only, receive-only).

Keeping the SPIEN active mode is supported when:

- A single channel is used (bit MCSPI\_MODULCTRL[Single] is set to 1).
- Transfer parameters of the transfer are loaded in the configuration register (MCSPI\_CH(I)CONF) in the appropriate channel.

The state of the SPIEN signal is programmable.

- Writing 1 into the bit FORCE of the register MCSPI\_CH(I)CONF drives high the SPIEN line when MCSPI\_CHCONF(I)[EPOL] is set to zero, and drives it low when MCSPI\_CHCONF(I)[EPOL] is set.
- Writing 0 into the bit FORCE of the register MCSPI\_CH(I)CONF drives low the SPIEN line when MCSPI\_CHCONF(I)[EPOL] is set to zero, and drives it high when MCSPI\_CHCONF(I)[EPOL] is set.
- A single channel is enabled (MCSPI\_CH(I)CTRL[En] set to 1) . The first enabled channel activates the SPIEN line.

Once the channel is enabled, the SPIEN signal is activated with the programmed polarity.

As in multi-channel master mode, the start of the transfer depends on the status of the transmitter register, the status of the receiver register and the mode defined by the bits TRM in the configuration register (transmit only, receive only or transmit and receive) of the enabled channel.

The status of the serialization completion of each SPI word is given by the bit EOT of the register MCSPI\_CH(I)STAT. The bit RX\_full in the MCSPI\_IRQSTATUS register is set when a received data is loaded from the shift register to the receiver register.

A change in the configuration parameters is propagated directly on the SPI interface. If the SPIEN signal is activated the user must insure that the configuration is changed only between SPI words, in order to avoid corrupting the current transfer.

---

**NOTE:** The SPIEN polarity, the SPICLK phase and SPICLK polarity must not be modified when the SPIEN signal is activated. The Transmit/Receive mode, programmable with the bit TRM can be modified only when the channel is disabled. The channel can be disabled and enabled while the SPIEN signal is activated.

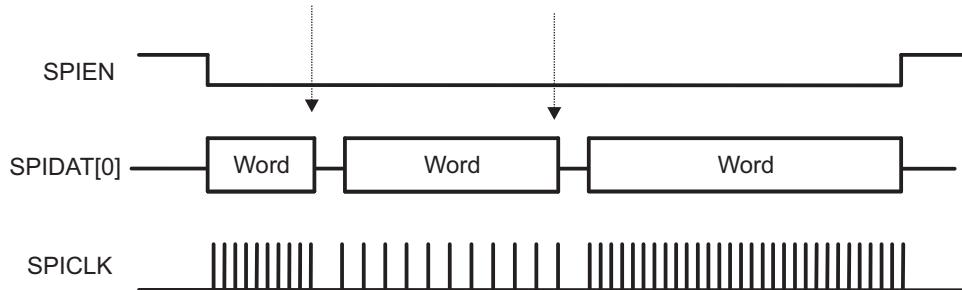
---

The delay between SPI words that requires the connected SPI slave device to switch from one configuration (Transmit only for instance) to another (receive only for instance) must be handled under software responsibility.

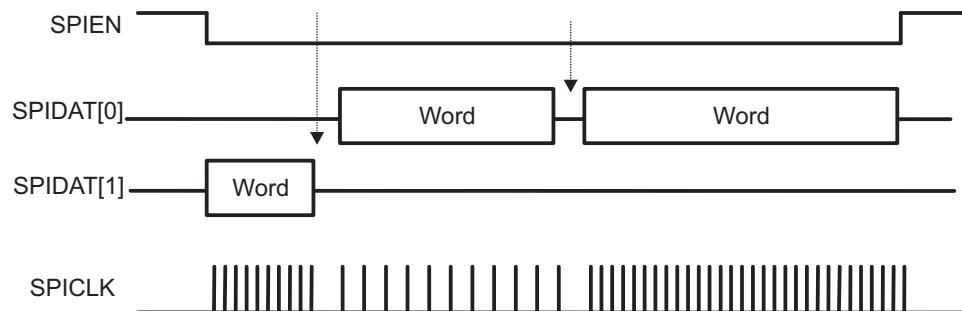
At the end of the last SPI word, the channel must be deactivated (MCSPI\_CH(I)CTRL[En] is cleared to 0) and the SPIEN can be forced to its inactive state (MCSPI\_CH(I)CONF[Force]).

[Figure 24-9](#) and [Figure 24-10](#) show successive transfers with SPIEN kept active low with a different configuration for each SPI word in respectively single data pin interface mode and two data pins interface mode. The arrows indicate when the channel is disabled before a change in the configuration parameters and enabled again.

**Figure 24-9. Continuous Transfers With SPIEN Maintained Active (Single-Data-Pin Interface Mode)**



**Figure 24-10. Continuous Transfers With SPIEN Maintained Active (Dual-Data-Pin Interface Mode)**



**NOTE:** The turbo mode is also supported for the Keep SPIEN active mode when the following conditions are met:

- A single channel will be explicitly used (bit MCSPI\_MODULCTRL[Single] is set to 1).
- The turbo mode is enabled in the configuration of the channel (bit Turbo of the register (I)CONF).

#### 24.3.2.6.3 Turbo Mode

The purpose of the Turbo mode is to improve the throughput of the SPI interface when a single channel is enabled, by allowing transfers until the shift register and the receiver register are full.

This mode is programmable per channel (bit Turbo of the register (I)CONF). When several channels are enabled, the bit Turbo of the registers MCSPI\_CH(I)CONF has no effect, and the channel access to the shift registers remains as described in [Section 24.3.2.3](#).

In Turbo mode, **Rule 1** and **Rule 2** defined in [Section 24.3.2.3](#) are applicable but Rule 3 is not applicable. An enabled channel can be scheduled if its receive register is full (bit RXS of the register MCSPI\_CH(I)STAT) at the time of shift register assignment until the shift register is full.

In Turbo mode, **Rule 1** and **Rule 2** defined in [Section 24.3.2.3](#) are applicable but Rule 3 is not applicable. An enabled channel can be scheduled if its receive register is full (bit RXS of the register MCSPI\_CH(I)STAT) at the time of shift register assignment until the shift register is full.

The receiver register cannot be overwritten in turbo mode. In consequence the RX\_overflow bit, in MCSPI\_IRQSTATUS register is never set in this mode.

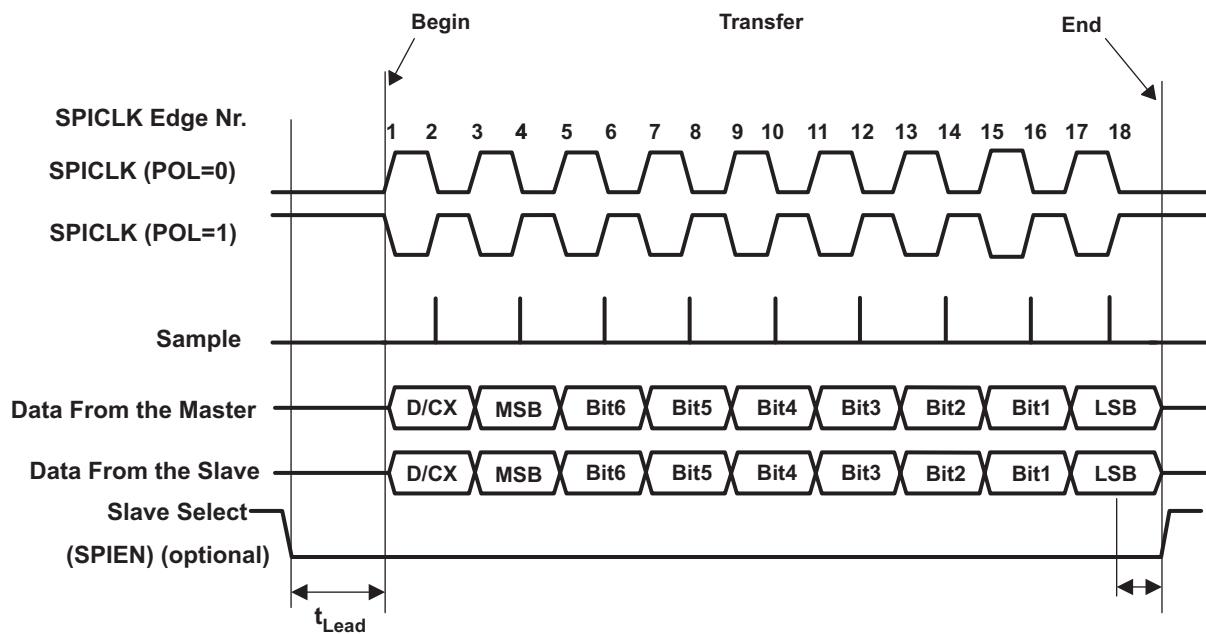
#### 24.3.2.7 Start Bit Mode

The purpose of the start bit mode is to add an extended bit before the SPI word transmission specified by word length WL. This feature is only available in master mode.

This mode is programmable per channel (bit Start bit enable SBE of the register MCSPI\_CH(I)CONF).

The polarity of the extended bit is programmable per channel and it indicates whether the next SPI word must be handled as a command when SBPOL is cleared to 0 or as a data or a parameter when SBPOL is set to 1. Moreover start bit polarity SBPOL can be changed dynamically during start bit mode transfer without disabling the channel for reconfiguration, in this case you have the responsibility to configure the SBPOL bit before writing the SPI word to be transmitted in TX register.

The start bit mode could be used at the same time as turbo mode and/or manual chip select mode. In this case only one channel could be used, no round-robin arbitration is possible.

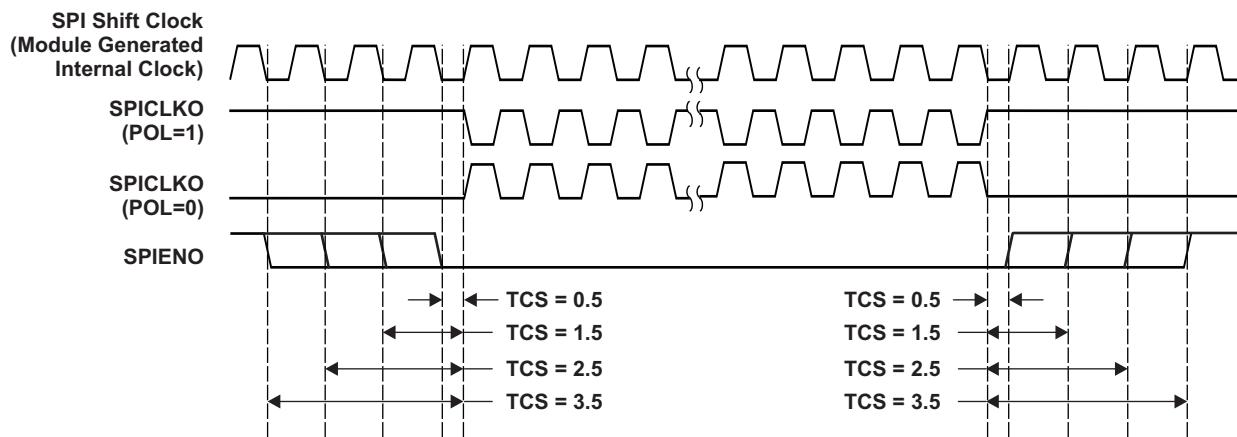
**Figure 24-11. Extended SPI Transfer With Start Bit PHA = 1**


#### 24.3.2.8 Chip-Select Timing Control

The chip select timing control is only available in master mode with automatic chip select generation (FORCE bit field is cleared to 0), to add a programmable delay between chip select assertion and first clock edge or chip select removal and last clock edge. The option is available only in 4 pin mode MCSPI\_MODULCTRL[1] PIN34 is cleared to 0.

This mode is programmable per channel (bit TCS of the register MCSPI\_CH(I)CONF). Figure 24-12 shows the chip-select SPIEN timing control.

**Figure 24-12. Chip-Select SPIEN Timing Controls**



**NOTE:** Because of the design implementation for transfers using a clock divider ratio set to 1 (clock bypassed), a half cycle must be added to the value between chip-select assertion and the first clock edge with PHA = 1 or between chip-select removal and the last clock edge with PHA = 0.

With an odd clock divider ratio which occurs when granularity is one clock cycle, that means that MCSPI\_CH(I)CONF[CLKG] is set to 1 and MCSPI\_CH(I)CONF[CLKD] has an even value, the clock duty cycle is not 50%, then one of the high level or low level duration is selected to be added to TCS delay.

Table 24-6 summarizes all delays between chip select and first (setup) or last (hold) clock edge.

In 3-pin mode this option is useless, the chip select SPIEN is forced to low state.

**Table 24-6. Chip Select ↔ Clock Edge Delay Depending on Configuration**

| Clock Ratio $F_{ratio}$                                 | Clock Phase PHA | Chip Select ↔ Clock Edge Delay  |   |
|---|-----------------|---|---|
|   |                 | Setup   | Hold  |
| 1   | 0               | $T_{ref} \times (TCS + \frac{1}{2})$                                  | $T_{ref} \times (TCS + 1)$  |
|   | 1               | $T_{ref} \times (TCS + 1)$  | $T_{ref} \times (TCS + \frac{1}{2})$                                  |
| Even $\geq 2$   | x               | $T_{ref} \times F_{ratio} \times (TCS + \frac{1}{2})$                 | $T_{ref} \times F_{ratio} \times (TCS + 1)$                           |
| Odd $\geq 3$ (only with MCSPI_CH(I)CONF[CLKG] set to 1) | 0               | $T_{ref} \times [(F_{ratio} \times TCS) + (F_{ratio} + \frac{1}{2})]$ | $T_{ref} \times [(F_{ratio} \times TCS) + (F_{ratio} + \frac{1}{2})]$ |
|   | 1               | $T_{ref} \times [(F_{ratio} \times TCS) + (F_{ratio} - \frac{1}{2})]$ | $T_{ref} \times [(F_{ratio} \times TCS) + (F_{ratio} - \frac{1}{2})]$ |

$T_{ref}$  = CLKSPIREF period in ns.  $F_{ratio}$  = SPI clock division ratio

The clock divider ratio depends on divider granularity MCSPI\_CH(I)CONF[CLKG]:

- MCSPI\_CH(I)CONF[CLKG] = 0 : granularity is power of two.  
 $F_{ratio} = 2^{MCSPI\_CH(I)\_CONF[CLKD]}$
- MCSPI\_CH(I)CONF[CLKG] = 1 : granularity is power of two.

$$F_{ratio} = MCSPI\_CH(I)CNTRL[EXTCLK].MCSPI\_CH(I)CONF[CLKD] + 1$$

### 24.3.2.9 Clock Ratio Granularity

By default the clock division ratio is defined by the register MCSPI\_CH(I)CONF[CLKD] with power of two granularity leading to a clock division in range 1 to 32768, in this case the duty cycle is always 50%. With bit MCSPI\_CH(I)CONF[CLKG] the clock division granularity can be changed to one clock cycle, in that case the register MCSPI\_CH(I)CTRL[EXTCLK] is concatenated with MCSPI\_CH(I)CONF[CLKD] to give a 12-bit width division ratio in range 1 to 4096.

When granularity is one clock cycle (MCSPI\_CH(I)CONF[CLKG] set to 1), for odd value of clock ratio the clock duty cycle is kept to 50-50 using falling edge of clock reference CLKSPIREF.

**Table 24-7. CLKSPIO High/Low Time Computation**

| Clock Ratio $F_{ratio}$ | CLKSPIO High Time              | CLKSPIO Low Time               |
|-------------------------|--------------------------------|--------------------------------|
| 1                       | $T_{high\_ref}$                | $T_{low\_ref}$                 |
| Even $\geq 2$           | $t_{ref} \times (F_{ratio}/2)$ | $t_{ref} \times (F_{ratio}/2)$ |
| Odd $\geq 3$            | $t_{ref} \times (F_{ratio}/2)$ | $t_{ref} \times (F_{ratio}/2)$ |

$T_{ref}$  = CLKSPIREF period in ns.  $T_{high\_ref}$  = CLKSPIREF high Time period in ns.  $T_{low\_ref}$  = CLKSPIREF low Time period in ns.  $F_{ratio}$  = SPI clock division ratio

$$F_{ratio} = MCSPI\_CH(I)CTRL[EXTCLK].MCSPI\_CH(I)CONF[CLKD] + 1$$

For odd ratio value the duty cycle is calculated as below:

$$\text{Duty\_cycle} = \frac{1}{2}$$

Granularity examples: With a clock source frequency of 48 MHz:

**Table 24-8. Clock Granularity Examples**

| MCSPI_CH(I)CTRL | MCSPI_CH(I)CONF | MCSPI_CH(I)CONF | $F_{ratio}$ | MCSPI_CH(I)CONF | MCSPI_CH(I)CONF | T <sub>high</sub> (ns) | T <sub>low</sub> (ns) | T <sub>period</sub> (ns) | Duty Cycle | F <sub>out</sub> (MHz) |
|-----------------|-----------------|-----------------|-------------|-----------------|-----------------|------------------------|-----------------------|--------------------------|------------|------------------------|
|                 |                 |                 |             | PHASE           | POL             |                        |                       |                          |            |                        |
| X               | 0               | 0               | 1           | X               | X               | 10.4                   | 10.4                  | 20.8                     | 50-50      | 48                     |
| X               | 1               | 0               | 2           | X               | X               | 20.8                   | 20.8                  | 41.6                     | 50-50      | 24                     |
| X               | 2               | 0               | 4           | X               | X               | 41.6                   | 41.6                  | 83.2                     | 50-50      | 12                     |
| X               | 3               | 0               | 8           | X               | X               | 83.2                   | 83.2                  | 166.4                    | 50-50      | 6                      |
| 0               | 0               | 1               | 1           | X               | X               | 10.4                   | 10.4                  | 20.8                     | 50-50      | 48                     |
| 0               | 1               | 1               | 2           | X               | X               | 20.8                   | 20.8                  | 41.6                     | 50-50      | 24                     |
| 0               | 2               | 1               | 3           | 1               | 0               | 31.2                   | 31.2                  | 62.4                     | 50-50      | 16                     |
| 0               | 2               | 1               | 3           | 1               | 1               | 31.2                   | 31.2                  | 62.4                     | 50-50      | 16                     |
| 0               | 3               | 1               | 4           | X               | X               | 41.6                   | 41.6                  | 83.2                     | 50-50      | 12                     |
| 5               | 0               | 1               | 81          | 1               | 0               | 842.4                  | 842.4                 | 1684.8                   | 50-50      | 0.592                  |
| 5               | 7               | 1               | 88          | X               | X               | 915.2                  | 915.2                 | 1830.4                   | 50-50      | 0.545                  |

#### 24.3.2.10 FIFO Buffer Management

The McSPI controller has a built-in 64-byte buffer in order to unload DMA or interrupt handler and improve data throughput.

This buffer can be used by only one channel and is selected by setting MCSPI\_CH(I)CONF[FFER] and/or MCSPI\_CH(I)CONF[FFEW] to 1.

If several channels are selected and several FIFO enable bit fields set to 1, the controller forces the buffer to be disabled for all channels. It is the responsibility of the driver to enable the buffer for only one channel.

The buffer can be used in the modes defined below:

- Master or Slave mode.
- Transmit only, Receive only or Transmit/Receive mode.
- Single channel or turbo mode, or in normal round robin mode. In round robin mode the buffer is used by only one channel.
- All word length MCSPI\_CH(I)CONF[WL] are supported.

Two levels AEL and AFL located in MCSPI\_XFERLEVEL register rule the buffer management. The granularity of these levels is one byte, then it is not aligned with SPI word length. It is the responsibility of the driver to set these values as a multiple of SPI word length defined in MCSPI\_CH(I)CONF[WL]. The number of byte written in the FIFO depends on word length (see [Table 24-9](#)).

**Table 24-9. FIFO Writes, Word Length Relationship**

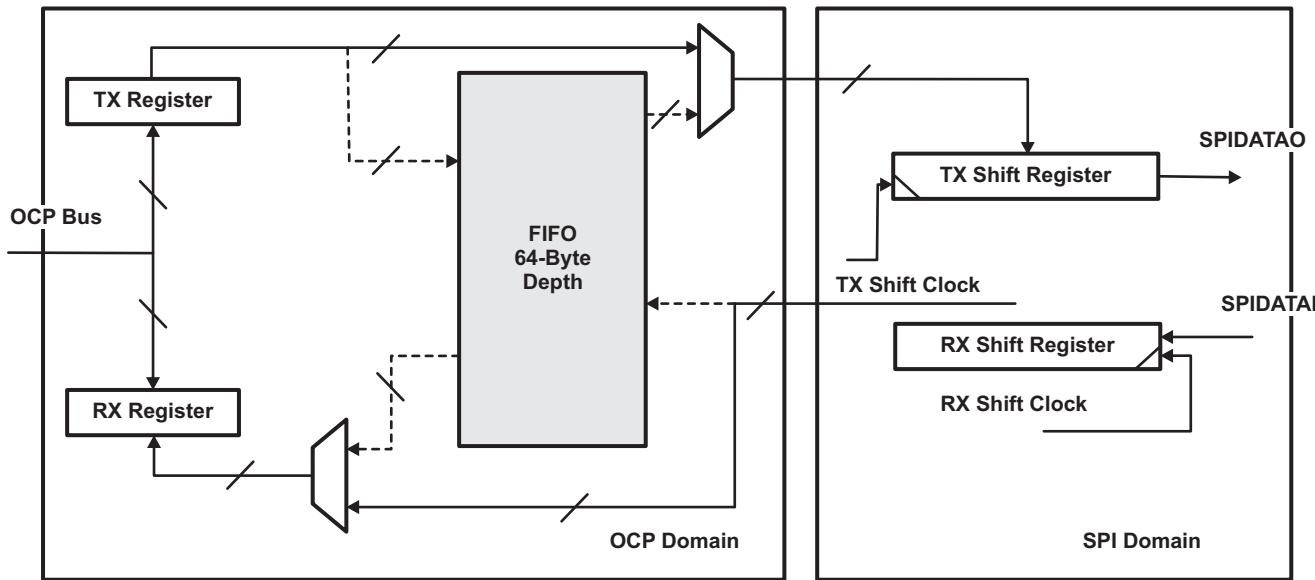
|                                    | <b>SPI Word Length WL</b> |                    |                     |
|------------------------------------|---------------------------|--------------------|---------------------|
|                                    | <b>3 ≤ WL ≤ 7</b>         | <b>8 ≤ WL ≤ 15</b> | <b>16 ≤ WL ≤ 31</b> |
| Number of byte written in the FIFO | 1 byte                    | 2 bytes            | 4 byte              |

#### 24.3.2.10.1 Split FIFO

The FIFO can be split into two part when module is configured in transmit/receive mode MCSPI\_CH(I)CONF[TRM] is cleared to 0 and MCSPI\_CH(I)CONF[FFER] and MCSPI\_CH(I)CONF[FFEW] asserted. Then system can access a 32-byte depth FIFO per direction.

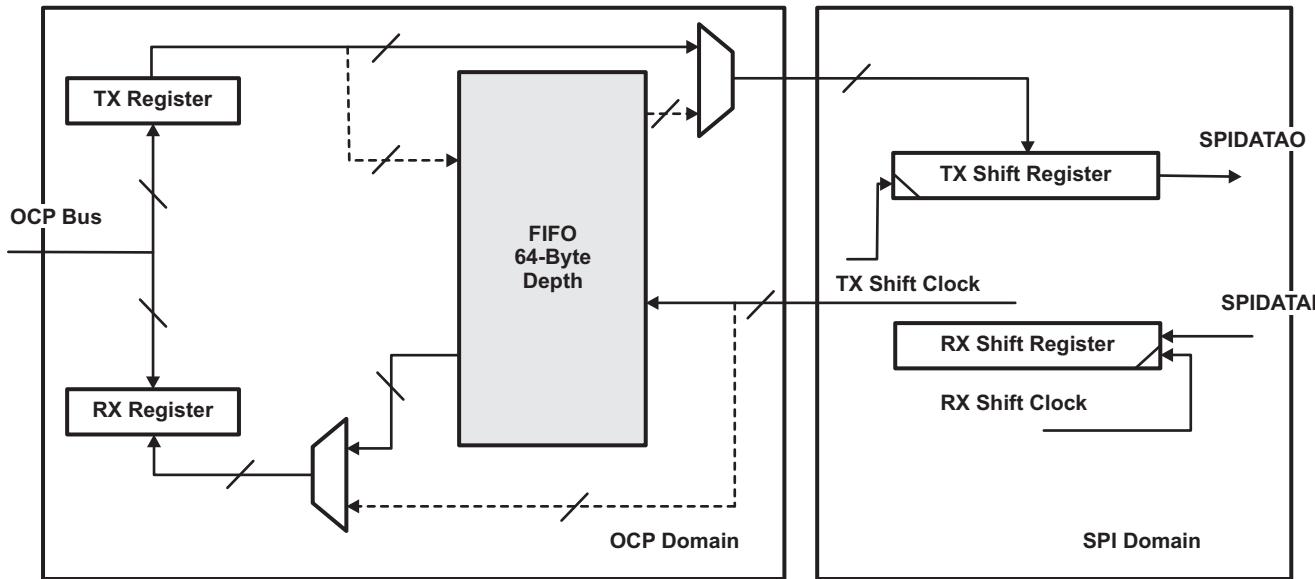
The FIFO buffer pointers are reset when the corresponding channel is enabled or FIFO configuration changes.

Figure 24-13. Transmit/Receive Mode With No FIFO Used



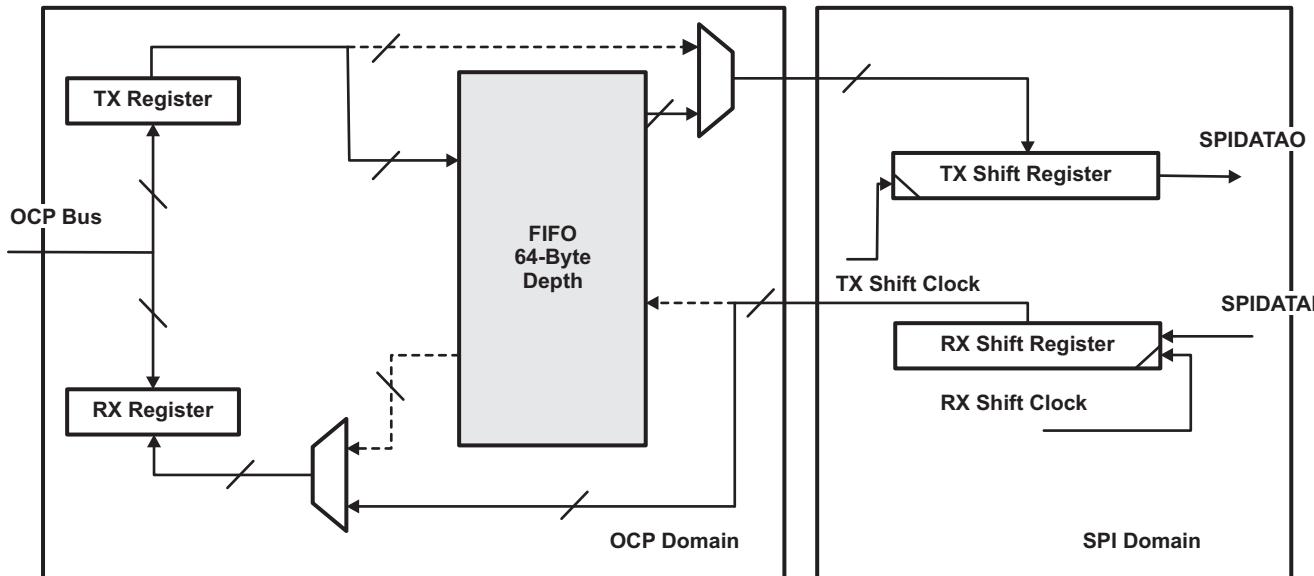
**Configuration:**  
**MCSPI\_CH(i)CONF[TRM]=0x0** Transmit/receive mode  
**MCSPI\_CH(i)CONF[FFRE]=0x0** FIFO disabled on receive path  
**MCSPI\_CH(i)CONF[FFWE]=0x0** FIFO disabled on transmit path

Figure 24-14. Transmit/Receive Mode With Only Receive FIFO Enabled



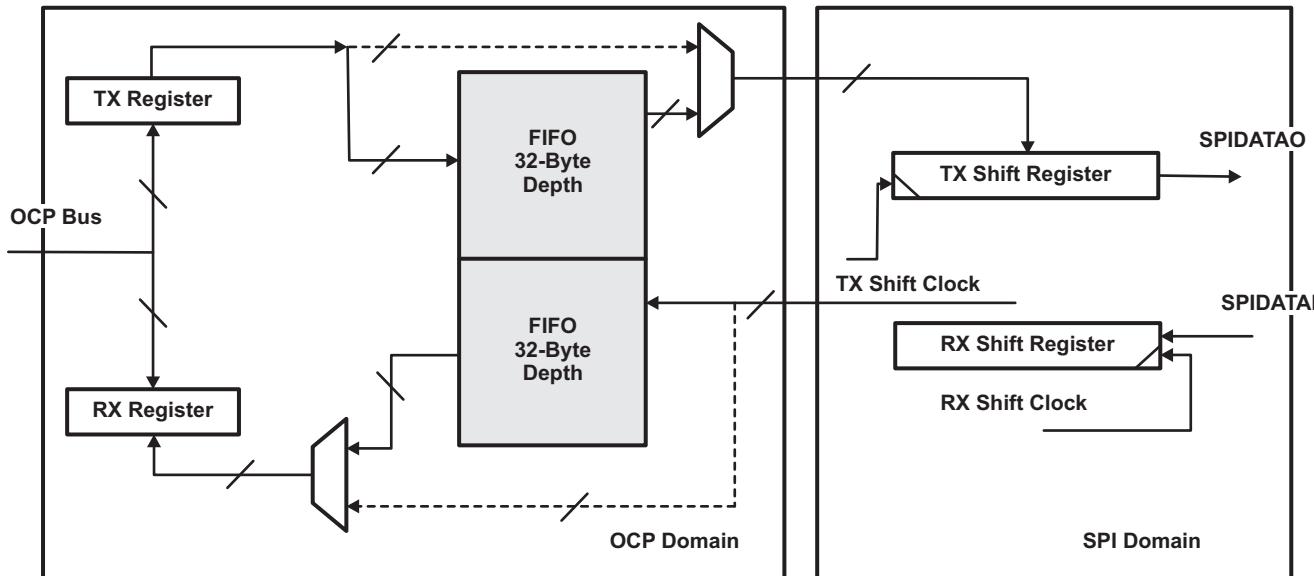
**Configuration:**  
**MCSPI\_CH(i)CONF[TRM]=0x0** Transmit/receive mode  
**MCSPI\_CH(i)CONF[FFRE]=0x1** FIFO enabled on receive path  
**MCSPI\_CH(i)CONF[FFWE]=0x0** FIFO disabled on transmit path

**Figure 24-15. Transmit/Receive Mode With Only Transmit FIFO Used**



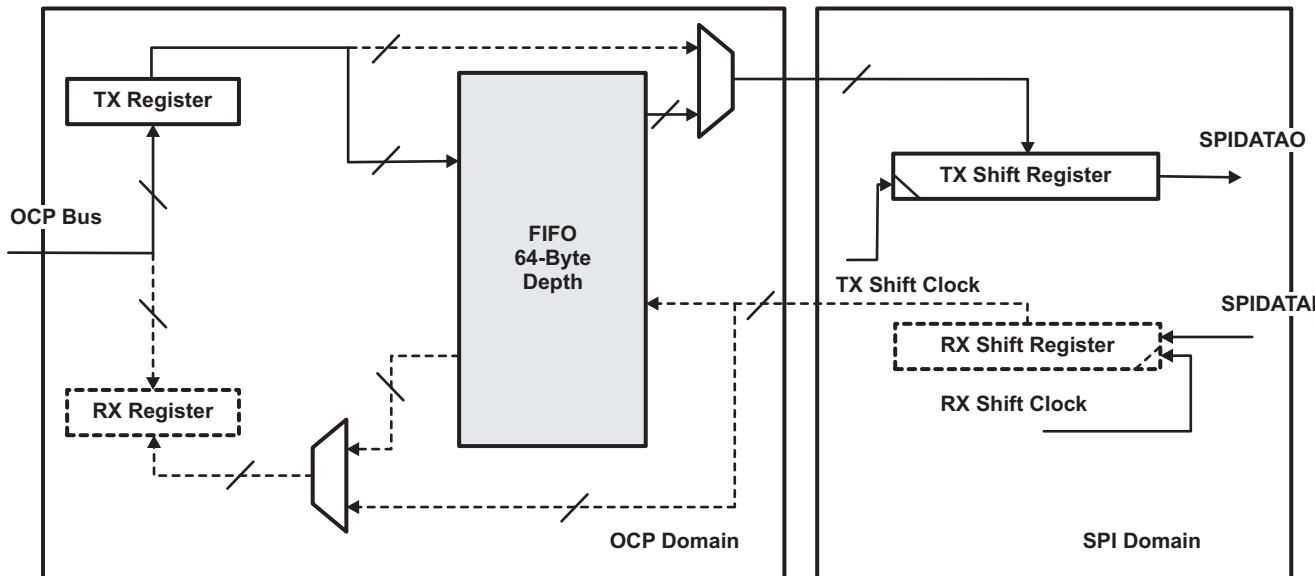
**Configuration:**  
`MCSPI_CH(i)CONF[TRM]=0x0` Transmit/receive mode  
`MCSPI_CH(i)CONF[FFRE]=0x0` FIFO disabled on receive path  
`MCSPI_CH(i)CONF[FFWE]=0x1` FIFO enabled on transmit path

**Figure 24-16. Transmit/Receive Mode With Both FIFO Direction Used**



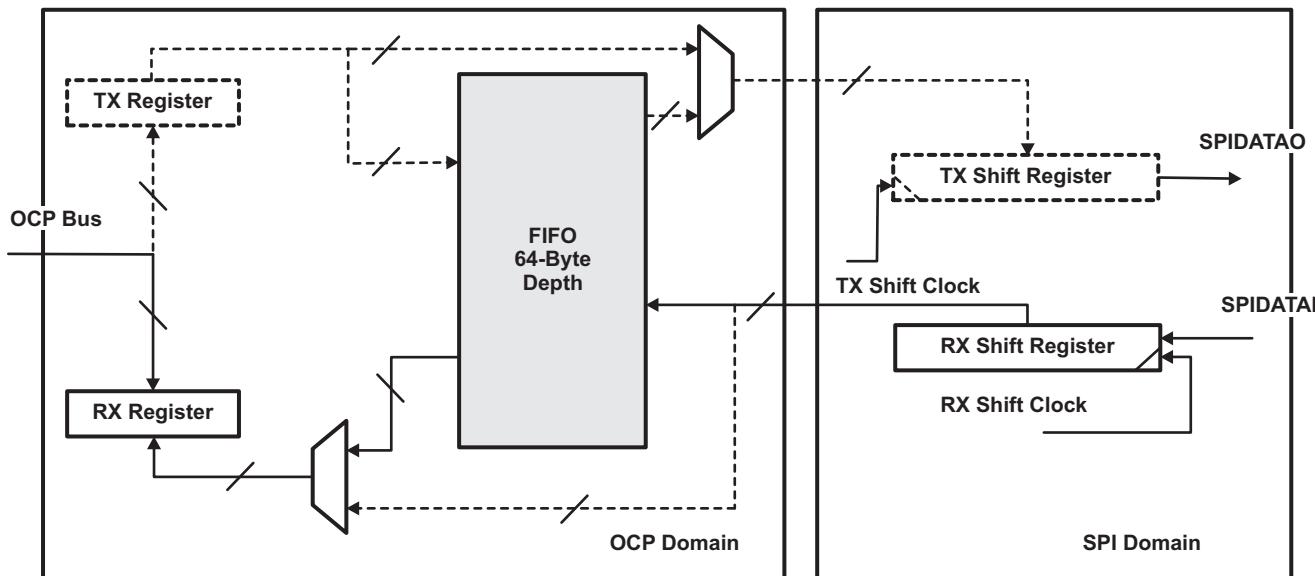
**Configuration:**  
`MCSPI_CH(i)CONF[TRM]=0x0` Transmit/receive mode  
`MCSPI_CH(i)CONF[FFRE]=0x1` FIFO enabled on receive path  
`MCSPI_CH(i)CONF[FFWE]=0x0` FIFO disabled on transmit path

Figure 24-17. Transmit-Only Mode With FIFO Used



**Configuration:**  
**MCSPI\_CH(i)CONF[TRM]=0x2** Transmit only mode  
**MCSPI\_CH(i)CONF[FFRE]=0x1** FIFO enabled on transmit path  
**MCSPI\_CH(i)CONF[FFWE]** not applicable

Figure 24-18. Receive-Only Mode With FIFO Used



**Configuration:**  
**MCSPI\_CH(i)CONF[TRM]=012** Receive only mode  
**MCSPI\_CH(i)CONF[FFRE]=0x1** FIFO enabled on receive path  
**MCSPI\_CH(i)CONF[FFWE]** not applicable

#### 24.3.2.10.2 Buffer Almost Full

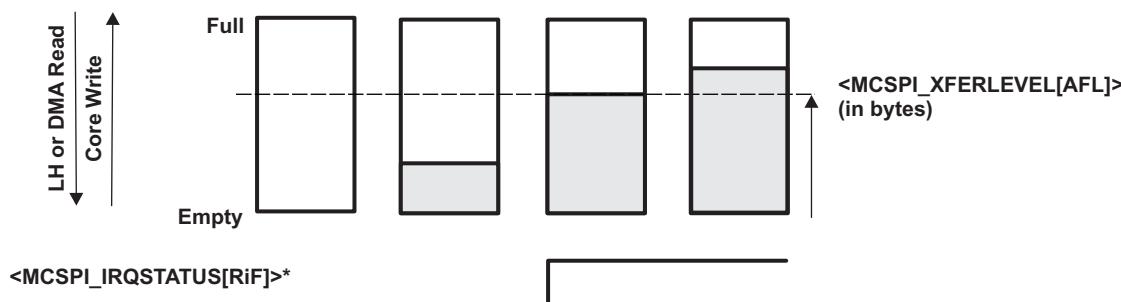
The bit field MCSPI\_XFERLEVEL[AFL] is needed when the buffer is used to receive SPI word from a slave (MCSPI\_CH(I)CONF[FFER] must be set to 1). It defines the almost full buffer status.

When FIFO pointer reaches this level an interrupt or a DMA request is sent to the CPU to enable system to read AFL+1 bytes from receive register. Be careful AFL+1 must correspond to a multiple value of MCSPI\_CH(I)CONF[WL].

When DMA is used, the request is de-asserted after the first receive register read.

No new request will be asserted until the system has performed the correct number of read operations from the buffer.

**Figure 24-19. Buffer Almost Full Level (AFL)**



\* non-DMA mode only. In DMA mode, the DMA RX request is asserted to its active level under identical conditions.

---

**NOTE:** SPI\_IRQSTATUS register bits are not available in DMA mode. In DMA mode, the SPIm\_DMA\_RXn request is asserted on the same conditions as the SPI\_IRQSTATUS RXn\_FULL flag.

---

#### 24.3.2.10.3 Buffer Almost Empty

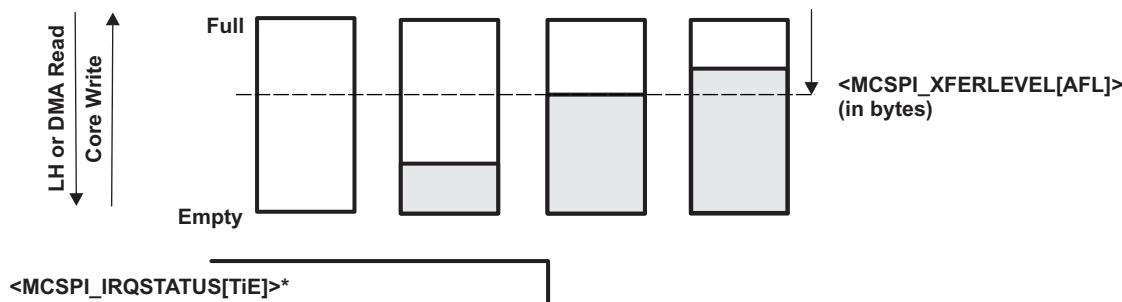
The bitfield MCSPI\_XFERLEVEL[AEL] is needed when the buffer is used to transmit SPI word to a slave (MCSPI\_CH(I)CONF[FFEW] must be set to 1). It defines the almost empty buffer status.

When FIFO pointer has not reached this level an interrupt or a DMA request is sent to the CPU to enable system to write AEL+1 bytes to transmit register. Be careful AEL+1 must correspond to a multiple value of MCSPI\_CH(I)CONF[WL].

When DMA is used, the request is de-asserted after the first transmit register write.

No new request will be asserted until the system has performed the correct number of write operations.

**Figure 24-20. Buffer Almost Empty Level (AEL)**



\* non-DMA mode only. In DMA mode, the DMA TX request is asserted to its active level under identical conditions.

#### 24.3.2.10.4 End of Transfer Management

When the FIFO buffer is enabled for a channel, the user should configure the MCSPI\_XFERLEVEL register, the AEL and AFL levels, and, especially, the WCNT bit field to define the number of SPI word to be transferred using the FIFO. This should be done before enabling the channel.

This counter allows the controller to stop the transfer correctly after a defined number of SPI words have been transferred. If WNCT is cleared to 0, the counter is not used and the user must stop the transfer manually by disabling the channel, in this case the user doesn't know how many SPI transfers have been done. For receive transfer, software shall poll the corresponding FFE bit field and read the Receive register to empty the FIFO buffer.

When End Of Word count interrupt is generated, the user can disable the channel and poll on MCSPI\_CH(I)STAT[FFE] register to know if SPI word is still there in FIFO buffer and read last words.

#### 24.3.2.10.5 Multiple SPI Word Access

The CPU has the ability to perform multiple SPI word access to the receive or transmit registers within a single 32-bit OCP access by setting the bit field MCSPI\_MODULCTRL[MOA] to '1' under specific conditions:

- The channel selected has the FIFO enable.
- Only FIFO sense enabled support the kind of access.
- The bit field MCSPI\_MODULCTRL[MOA] is set to 1
- Only 32-bit OCP access and data width can be performed to receive or transmit registers, for other kind of access the CPU must de-assert MCSPI\_MODULCTRL[MOA] bit fields.
- The Level MCSPI\_XFERLEVEL[AEL] and MCSPI\_XFERLEVEL[AFL] must be 32-bit aligned , it means that AEL[0] = AEL[1] = 1 or AFL[0] = AFL[1] = 1.
- If MCSPI\_XFERLEVEL[WCNT] is used it must be configured according to SPI word length.
- The word length of SPI words allows to perform multiple SPI access, that means that MCSPI\_CH(I)CONF[WL] < 16

Number of SPI word access depending on SPI word length:

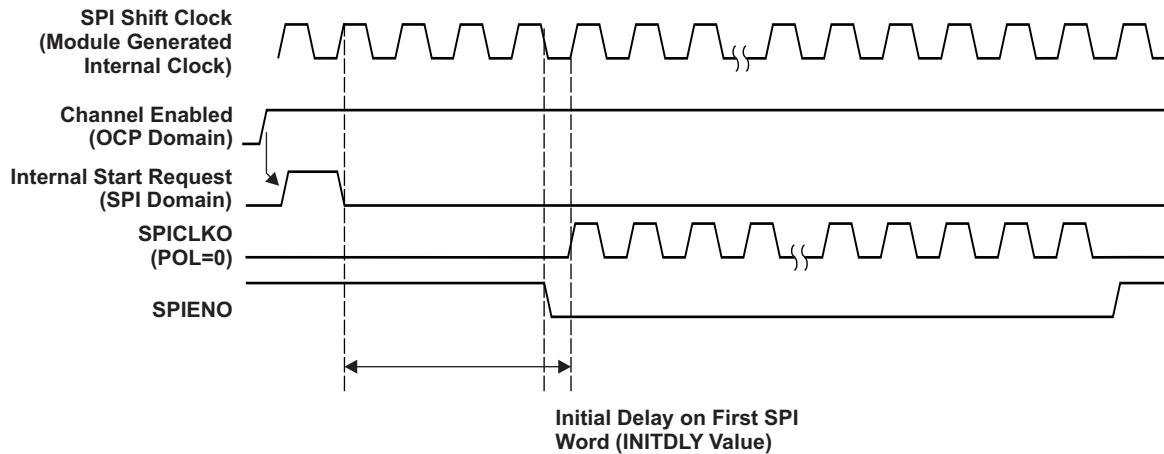
- $3 \leq WL \leq 7$ , SPI word length smaller or equal to byte length, four SPI words accessed per 32-bit OCP read/write. If word count is used (MCSPI\_XFERLEVEL[WCNT]), set the bit field to WCNT[0]=WCNT[1]=0
- $8 \leq WL \leq 15$ , SPI word length greater than byte or equal to 16-bit length, two SPI words accessed per 32-bit OCP read/write. If word count is used (MCSPI\_XFERLEVEL[WCNT]), set the bit field to WCNT[0]=0.
- $16 \leq WL$  multiple SPI word access not applicable.

#### 24.3.2.11 First SPI Word Delayed

The McSpi controller has the ability to delay the first SPI word transfer to give time for system to complete some parallel processes or fill the FIFO in order to improve transfer bandwidth. This delay is applied only on first SPI word after SPI channel enabled and first write in Transmit register. It is based on output clock frequency.

This option is meaningful in master mode and single channel mode MCSPI\_MODULECTRL[SINGLE] asserted.

**Figure 24-21. Master Single Channel Initial Delay**



Few delay values are available: No delay, 4/8/16/32 Spi cycles.

Its accuracy is half cycle in clock bypass mode and depends on clock polarity and phase.

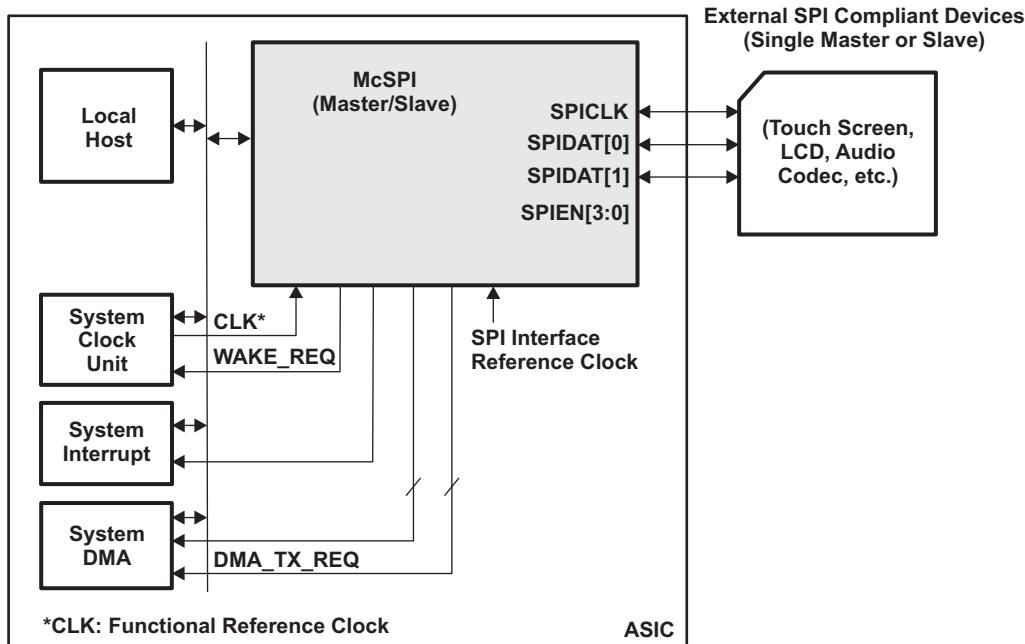
#### 24.3.2.12 3- or 4-Pin Mode

External SPI bus interface can be configured to use a restricted set of pin using the bit field MCSPI\_MODULCTRL[PIN34] and depending on targeted application:

- If MCSPI\_MODULCTRL[PIN34] is cleared to 0 (default value) the controller is in 4-pin mode using the SPI pins CLKSPI, SOMI, SIMO and chip enable CS.
- If MCSPI\_MODULCTRL[PIN34] is set to 1 the controller is in 3-pin mode using the SPI pins CLKSPI, SOMI and SIMO.

In 3-pin mode it is mandatory to put the controller in single channel master mode (MCSPI\_MODULECTRL[SINGLE] asserted) and to connect only one SPI device on the bus.

**Figure 24-22. 3-Pin Mode System Overview**



In 3-pin mode all options related to chip select management are useless:

- MCSPI\_CHxCONF[EPOL]
- MCSPI\_CHxCONF[TCS0]
- MCSPI\_CHxCONF[FORCE]

The chip select pin SPIEN is forced to '0' in this mode.

### 24.3.3 Slave Mode

McSPI is in slave mode when the bit MS of the register MCSPI\_MODULCTRL is set.

In slave mode, McSPI can be connected to up to 4 external SPI master devices. McSPI handles transactions with a single SPI master device at a time.

In slave mode, McSPI initiates data transfer on the data lines (SPIDAT[1:0]) when it receives an SPI clock (SPICLK) from the external SPI master device.

The controller is able to work with or without a chip select SPIEN depending on MCSPI\_MODULCTRL[PIN34] bit setting. It also supports transfers without a dead cycle between two successive words.

#### 24.3.3.1 Dedicated Resources

In slave mode, enabling a channel that is not channel 0 has no effect. Only channel 0 can be enabled. The channel 0, in slave mode has the following resources:

- Its own channel enable, programmable with the bit EN of the register MCSPI\_CH0CTRL. This channel should be enabled before transmission and reception. Disabling the channel, outside data word transmission, remains under user responsibility.
- Any of the 4 ports SPIEN[3:0] can be used as a slave SPI device enable. This is programmable with the bits SPIENSLV of the register MCSPI\_CH0CONF.
- Its own transmitter register MCSPI\_TX on top of the common shift register. If the transmitter register is empty, the status bit TXS of the register MCSPI\_CH0STAT is set. When McSPI is selected by an external master (active signal on the SPIEN port assigned to channel 0), the transmitter register content of channel0 is always loaded in shift register whether it has been updated or not. The transmitter register should be loaded before McSPI is selected by a master.
- Its own receiver register MCSPI\_RX on top of the common shift register. If the receiver register is full, the status bit RXS of the register MCSPI\_CH0STAT is set.

---

**NOTE:** The transmitter register and receiver registers of the other channels are not used. Read from or Write in the registers of a channel other than 0 has no effect.

---

- Its own communication configuration with the following parameters via the register MCSPI\_CH0CONF:
  - Transmit/Receive modes, programmable with the bit TRM.
  - Interface mode (Two data pins or Single data pin) and data pins assignment, both programmable with the bits IS and DPE.
  - SPI word length, programmable with the bits WL.
  - SPIEN polarity, programmable with the bit EPOL.
  - SPICLK polarity, programmable with the bit POL.
  - SPICLK phase, programmable with the bit PHA.
  - Use a FIFO buffer or not, programmable with FFER and FFEW, depending on transfer mode TRM.

The SPICLK frequency of a transfer is controlled by the external SPI master connected to McSPI. The bits CLKD0 of the register 0CONF are not used in slave mode.

---

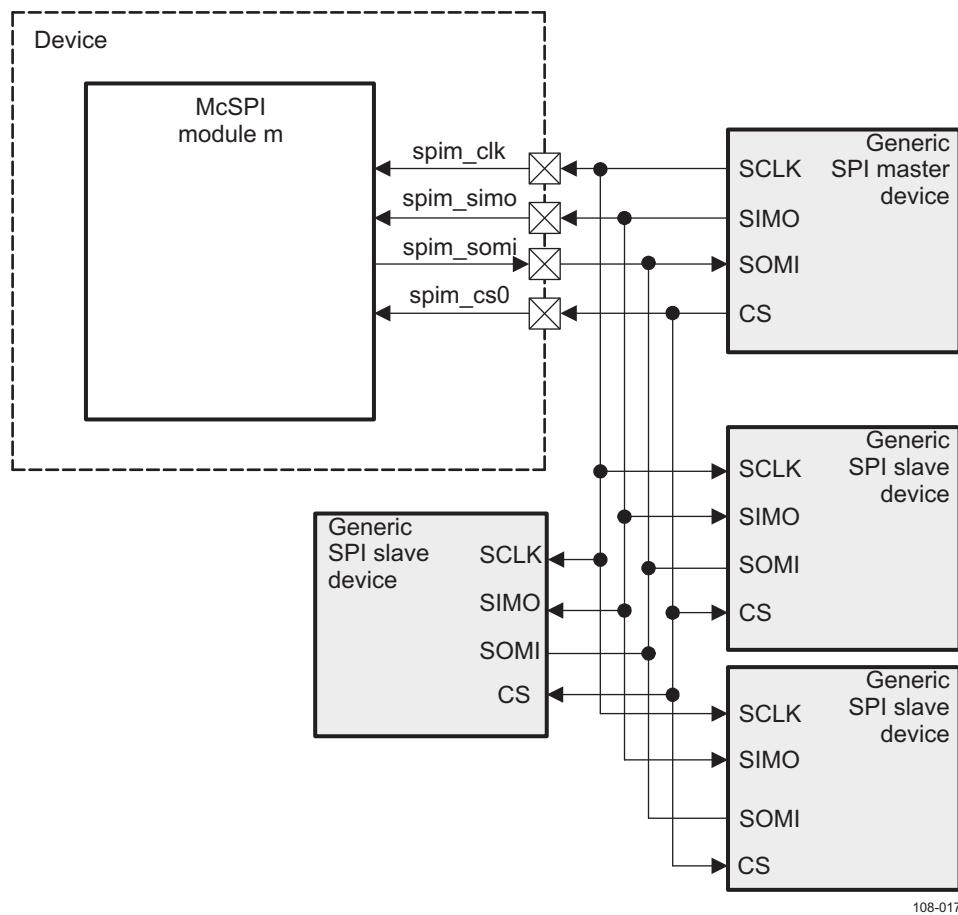
**NOTE:** The configuration of the channel can be loaded in the MCSPI\_CH0CONF register only when the channel is disabled.

---

- Two DMA requests events, read and write, to synchronize read/write accesses of the DMA controller with the activity of McSPI. The DMA requests are enabled with the bits DMAR and DMAW of the register 0CONF.
- Four interrupts events.

Figure 24-23 shows an example of four slaves wired on a single master device.

**Figure 24-23. Example of SPI Slave with One Master and Multiple Slave Devices on Channel 0**



### 24.3.3.2 Interrupt Events in Slave Mode

The interrupt events related to the transmitter register state are TX\_empty and TX\_underflow. The interrupt events related to the receiver register state are RX\_full and RX\_overflow.

#### 24.3.3.2.1 TX\_EMPTY

The event TX\_empty is activated when the channel is enabled and its transmitter register becomes empty. Enabling channel automatically raises this event. When FIFO buffer is enabled (MCSPI\_CH(I)CONF[FFEW] set to 1), the TX\_empty is asserted as soon as there is enough space in buffer to write a number of byte defined by MCSPI\_XFERLEVEL[AEL].

Transmitter register must be load to remove source of interrupt and TX\_empty interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

When FIFO is enabled, no new TX\_empty event will be asserted unless the host performs the number of writes to the transmitter register defined by MCSPI\_XFERLEVEL[AEL]. It is the responsibility of the Local Host to perform the right number of writes.

#### 24.3.3.2.2 TX\_UNDERFLOW

The event TX\_underflow is activated when channel is enabled and if the transmitter register or FIFO (if use of buffer is enabled) is empty (not updated with new data) when an external master device starts a data transfer with McSPI (transmit and receive).

When the FIFO is enabled, the data read while the underflow flag is set will not be the last word written to the FIFO.

The TX\_underflow indicates an error (data loss) in slave mode.

To avoid having TX\_underflow event at the beginning of a transmission, the event TX\_underflow is not activated when no data has been loaded into the transmitter register since channel has been enabled.

TX\_underflow interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

#### 24.3.3.2.3 RX\_FULL

The event RX\_FULL is activated when channel is enabled and receiver becomes filled (transient event). When FIFO buffer is enabled (MCSPI\_CH(I)CONF[FFER] set to 1), the RX\_FULL is asserted as soon as there is a number of bytes holds in buffer to read defined by MCSPI\_XFERLEVEL[AFL].

Receiver register must be read to remove source of interrupt and RX\_full interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

When FIFO is enabled, no new RX\_FULL event will be asserted unless the host has performed the number of reads from the receive register defined by MCSPI\_XFERLEVEL[AFL]. It is the responsibility of Local Host to perform the right number of reads.

#### 24.3.3.2.4 RX\_OVERFLOW

The RX0\_OVERFLOW event is activated in slave mode in either transmit-and-receive or receive-only mode, when a channel is enabled and the SPI\_RXn register or FIFO is full when a new SPI word is received. The SPI\_RXn register is always overwritten with the new SPI word. If the FIFO is enabled, data within the FIFO is overwritten, it must be considered as corrupted. The RX0\_OVERFLOW event should not appear in slave mode using the FIFO.

The RX0\_OVERFLOW indicates an error (data loss) in slave mode.

The SPI\_IRQSTATUS[3] RX0\_OVERFLOW interrupt status bit must be cleared for interrupt line deassertion (if the event is enabled as the interrupt source).

#### 24.3.3.2.5 End of Word Count

The event end of word (EOW) count is activated when channel is enabled and configured to use the built-in FIFO. This interrupt is raised when the controller had performed the number of transfer defined in MCSPI\_XFERLEVEL[WCNT] register. If the value was programmed to 0000h, the counter is not enabled and this interrupt is not generated.

The EOW count interrupt also indicates that the SPI transfer has halted on the channel using the FIFO buffer.

The EOW interrupt status bit must be cleared for interrupt line de-assertion (if event enable as interrupt source).

#### 24.3.3.3 Slave Transmit-and-Receive Mode

The slave transmit and receive mode is programmable (TRM bit cleared to 0 in the register MCSPI\_CH(I)CONF).

After the channel is enabled, transmission and reception proceeds with interrupt and DMA request events.

In slave transmit and receive mode, transmitter register should be loaded before McSPI is selected by an external SPI master device.

Transmitter register or FIFO (if enabled) content is always loaded into the shift register whether it has been updated or not. The event TX\_underflow is activated accordingly, and does not prevent transmission.

On completion of SPI word transfer (bit EOT of the register MCSPI\_CH(I)STAT is set) the received data is transferred to the channel receive register. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured in one data direction, transmit or receive, then the FIFO is seen as a unique 64-byte buffer. It can also be configured in both data directions, transmit and receive, then the FIFO is split into two separate 32-byte buffers with their own address space management.

#### 24.3.3.4 Slave Receive-Only Mode

The slave receive mode is programmable (MCSPI\_CH(i)CONF[TRM set to 01]).

In receive-only mode, the transmitter register should be loaded before McSPI is selected by an external SPI master device. Transmitter register or FIFO (if enabled) content is always loaded into the shift register whether it has been updated or not. The event TX\_underflow is activated accordingly, and does not prevent transmission.

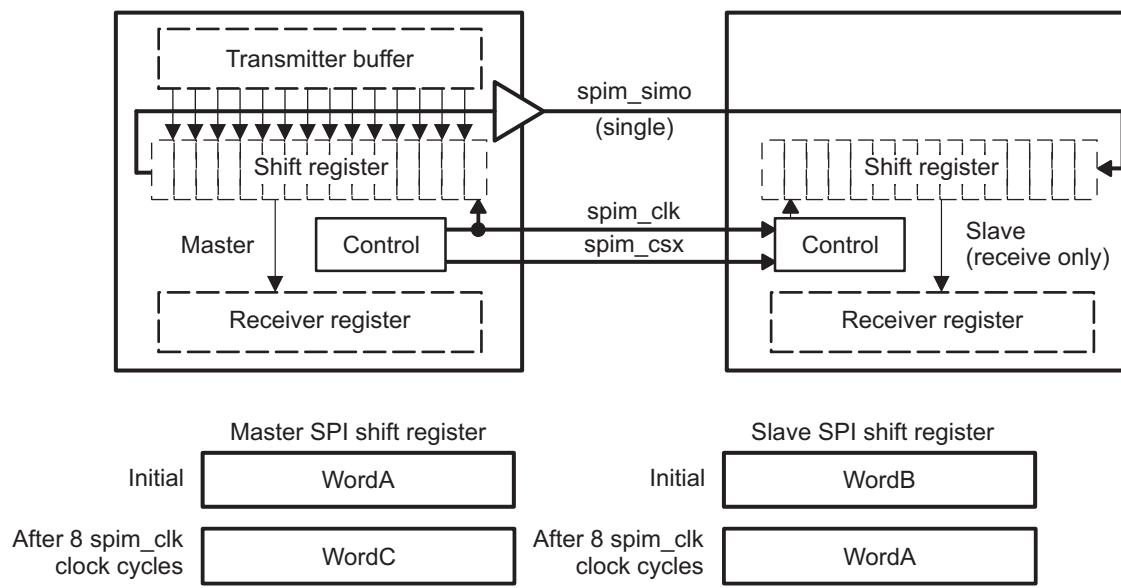
When an SPI word transfer completes (the MCSPI\_CH(I)STAT[EOT] bit (with  $I = 0$ ) is set to 1), the received data is transferred to the channel receive register.

To use McSPI as a slave receive-only device with MCSPI\_CH(I)CONF[TRM]=00, the user has the responsibility to disable the TX\_empty and TX\_underflow interrupts and DMA write requests due to the transmitter register state.

On completion of SPI word transfer (bit EOT of the register MCSPI\_CH(I)STAT is set) the received data is transferred to the channel receive register. This bit is meaningless when using the Buffer for this channel. The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI\_CH(I)CONF register, then the FIFO is seen as a unique 64-byte buffer.

**Figure 24-24** shows an example of a half-duplex system with a master device on the left and a receive-only slave device on the right. Each time one bit transfers out from the master, one bit transfers in to the slave. If WordA is 8 bits, then after eight cycles of the serial clock spim\_clk, WordA transfers from the master to the slave.

**Figure 24-24. SPI Half-Duplex Transmission (Receive-Only Slave)**



108-032

#### 24.3.3.5 Slave Transmit-Only Mode

The slave transmit-only mode is programmable (MCSPI\_CH(i)CONF[TRM set to 10). This mode eliminates the need for the CPU to read the receiver register (minimizing data movement) when only transmission is meaningful.

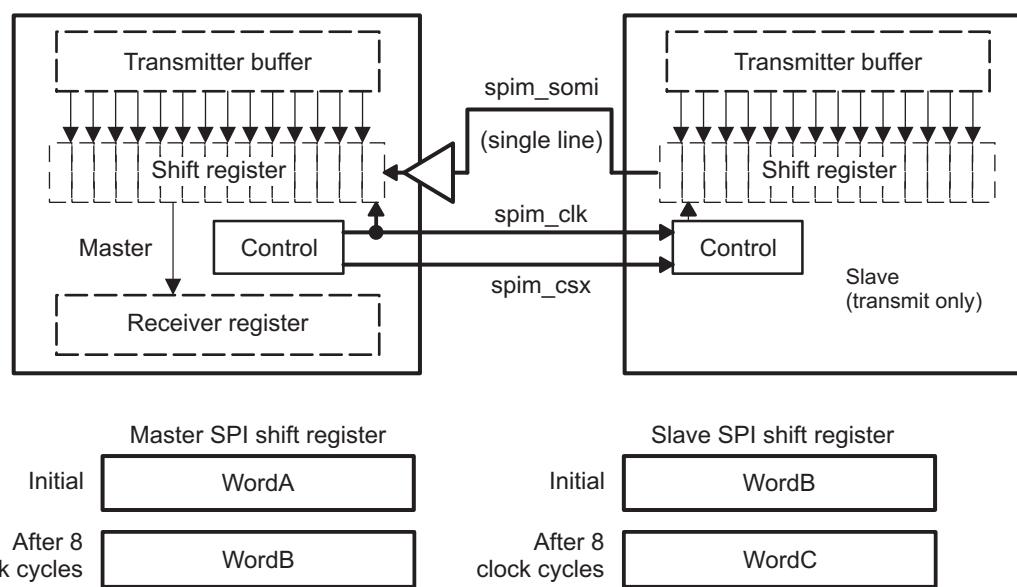
To use McSPI as a slave transmit-only device with MCSPI\_CH(I)CONF[TRM]=10, the user should disable the RX\_full and RX\_overflow interrupts and DMA read requests due to the receiver register state.

On completion of SPI word transfer the bit EOT of the register MCSPI\_CH(I)STAT is set. This bit is meaningless when using the Buffer for this channel.

The built-in FIFO is available in this mode and can be configured with FFER bit field in the MCSPI\_CH(I)CONF register, then the FIFO is seen as a unique 64-byte buffer.

[Figure 24-25](#) shows a half-duplex system with a master device on the left and a transmit-only slave device on the right. Each time a bit transfers out from the slave device, one bit transfers in the master. If WordB is 8-bits, then after eight cycles of the serial clock spim\_clk, WordB transfers from the slave to the master.

**Figure 24-25. SPI Half-Duplex Transmission (Transmit-Only Slave)**



108-031

#### 24.3.4 Interrupts

According to its transmitter register state and its receiver register state each channel can issue interrupt events if they are enabled.

The interrupt events are listed in the [Section 24.3.2.2](#) and in [Section 24.3.3.2](#).

Each interrupt event has a status bit, in the MCSPI\_IRQSTATUS register, which indicates service is required, and an interrupt enable bit, in the MCSPI\_IRQENABLE register, which enables the status to generate hardware interrupt requests.

When an interrupt occurs and it is later masked (IRQENABLE), the interrupt line is not asserted again even if the interrupt source has not been serviced.

McSPI supports interrupt driven operation and polling.

#### 24.3.4.1 Interrupt-Driven Operation

Alternatively, an interrupt enable bit, in the MCSPI\_IRQENABLE register, can be set to enable each of the events to generate interrupt requests when the corresponding event occurs. Status bits are automatically set by hardware logic conditions.

When an event occurs (the single interrupt line is asserted), the CPU must:

- Read the MCSPI\_IRQSTATUS register to identify which event occurred,
- Read the receiver register that corresponds to the event in order to remove the source of an RX\_full event, or write into the transmitter register that corresponds to the event in order to remove the source of a TX\_empty event. No action is needed to remove the source of the events TX\_underflow and RX\_overflow.
- Write a 1 into the corresponding bit of MCSPI\_IRQSTATUS register to clear the interrupt status, and release the interrupt line.

The interrupt status bit should always be reset after the channel is enabled and before the event is enabled as an interrupt source.

#### 24.3.4.2 Polling

When the interrupt capability of an event is disabled in the MCSPI\_IRQENABLE register, the interrupt line is not asserted and:

- The status bits in the MCSPI\_IRQSTATUS register can be polled by software to detect when the corresponding event occurs.
- Once the expected event occurs, CPU must read the receiver register that corresponds to the event in order to remove the source of an RX\_full event, or write into the transmitter register that corresponds to the event in order to remove the source of a TX\_empty event. No action is needed to remove the source of the events TX\_underflow and RX\_overflow.
- Writing a 1 into the corresponding bit of MCSPI\_IRQSTATUS register clears the interrupt status and does not affect the interrupt line state.

#### 24.3.5 DMA Requests

McSPI can be interfaced with a DMA controller. At system level, the advantage is to free the local host of the data transfers.

According to its transmitter register state, its receiver register state or FIFO level (if use of buffer for the channel) each channel can issue DMA requests if they are enabled.

The DMA requests need to be disabled in order to get TX and RX interrupts, in order to define either the end of the transfer or the transfer of the last words for the modes listed below:

- Master Transmit On
- Master normal receive only mode
- Master turbo receive only m
- Slave Transmit Only

There are two DMA request lines per channel. The management of DMA requests differ according to use of FIFO buffer or not:

##### 24.3.5.1 FIFO Buffer Disabled

The DMA Read request line is asserted when the channel is enabled and a new data is available in the receive register of the channel. DMA Read request can be individually masked with the bit DMAR of the register MCSPI\_CH(I)CONF. The DMA Read request line is de-asserted on read completion of the receive register of the channel.

The DMA Write request line is asserted when the channel is enabled and the transmitter register of the channel is empty. DMA Write request can be individually masked with the bit DMAW of the register MCSPI\_CH(I)CONF. The DMA Write request line is de-asserted on load completion of the transmitter register of the channel.

Only one SPI word can be transmitted/received per OCP bus access to write/read the transmit or receive register.

#### 24.3.5.2 FIFO Buffer Enabled

The DMA Read request line is asserted when the channel is enabled and a number of bytes defined in MCSPI\_XFERLEVEL[AFL] bit field is hold in FIFO buffer for the receive register of the channel. DMA Read request can be individually masked with the bit DMAR of the register MCSPI\_CH(I)CONF. The DMA Read request line is de-asserted on the first SPI word read completion of the receive register of the channel. No new DMA request will be asserted again as soon as user has not performed the right number of read accesses defined by MCSPI\_XFERLEVEL[AFL] it is under user responsibility.

The DMA Write request line is asserted when the channel is enabled and the number of bytes hold in FIFO buffer is below the level defined by the MCSPI\_XFERLEVEL[AEL] bit field. DMA Write request can be individually masked with the bit DMAW of the register MCSPI\_CH(I)CONF. The DMA Write request line is de-asserted on load completion of the first SPI word in the transmitter register of the channel. No new DMA request will be asserted again as soon as user has not performed the right number of write accesses defined by MCSPI\_XFERLEVEL[AEL] it is under user responsibility.

Only one SPI word can be transmitted/received per OCP bus access to write/read the transmit or receive FIFO.

#### 24.3.5.3 DMA 256-Bit Aligned Addresses

The controller has two registers, MCSPI\_DAFTX and MCSPI\_DAFRX, used only with an enabled channel which manages the FIFO to be compliant the a DMA handler providing only 256-bit aligned addresses.

This features is activated when the bit field MCSPI\_MODULCTRL[FDFA] is set to '1' and only one enabled channel have its bit field MCSPI\_CH(I)CONF[FFEW] or MCSPI\_CH(I)CONF[FFER] enabled.

In this case the registers MCSPI\_TX(I) and MCSPI\_RX(I) are not used and data is managed through registers MCSPI\_DAFTX and MCSPI\_DAFRX.

#### 24.3.6 Emulation Mode

The MReqDebug input differentiates a regular access of a processor (application access), from an emulator access.

Application access: MReqDebug = 0

In functional mode, the consequences of a read of a receiver register MCSPI\_RX(i) are the following:

- The source of an RXi\_Full event in the MCSPI\_IRQSTATUS register is removed, if it was enabled in the MCSPI\_IRQENABLE register.
- The RXiS status bit in the MCSPI\_IRQSTATUS register is cleared.
- In master mode, depending on the round robin arbitration, and the transmitter register state, the channel may access to the shift register for transmission/reception.

Emulator access: MReqDebug = 1

In emulation mode, McSPI behavior is the same as in functional mode but a read of a receiver register MCSPI\_RX(i) is not intrusive:

- MCSPI\_RX(i) is still considered as not read. When the FIFO buffer is enabled, pointers are not updated.
- The source of an RXi\_Full event in the MCSPI\_IRQSTATUS register is not removed. The RXiS status bit in the MCSPI\_CH(i)STAT register is held steady.

In emulation mode, as in functional mode, based on the ongoing data transfers, the status bits of the MCSPI\_CH(i)STAT register may be optionally updated, the interrupt and DMA request lines may be optionally asserted.

### 24.3.7 Power Saving Management

Independently of the module operational modes (Transmit and/or Receive), two modes of operations are defined from a power management perspective: normal and idle modes.

The two modes are temporally fully exclusive.

#### 24.3.7.1 Normal Mode

Both the OCP clock and SPI clock (CLKSPIREF) provided to McSPI must be active for both master and slave modes. The auto-gating of the module OCP clock and SPI clock occurs when the following conditions are met:

- The bit Autoldle of the register MCSPI\_SYSConfig is set.
- In master mode, there is no data to transmit or receive in all channels.
- In slave mode, the SPI is not selected by the external SPI master device and no OCP accesses.

Autogating of the module OCP clock and SPI clock stops when the following conditions are met:

- In master mode, an OCP access occurs.
- In slave mode, an OCP access occurs or McSPI is selected by an external SPI master device.

#### 24.3.7.2 Idle Mode

The OCP clock and SPI clock provided to McSPI may be switched off on system power manager request and switched back on module request.

McSPI is compliant with the power management handshaking protocol: idle request from the system power manager, idle acknowledgement from McSPI.

The idle acknowledgement in response to an idle request from the system power manager varies according to a programmable mode in the MCSPI\_SYSConfig register: No idle mode, force idle mode, and smart idle mode.

- When programmed for no idle mode (the bit SIdleMode of the register MCSPI\_SYSConfig is set to "01"), the module ignores the system power manager request, and behaves normally, as if the request was not asserted.
- When programmed for smart idle mode (the bit SIdleMode of the register MCSPI\_SYSConfig is set to "10"), the module acknowledges the system power manager request according to its internal state.
- When programmed for force idle mode (the bit SIdleMode of the register MCSPI\_SYSConfig is set to "00"), the module acknowledges the system power manager request unconditionally.

The OCP clock will be optionally switched off, during the smart idle mode period, if the bit ClockActivity of the register MCSPI\_SYSConfig is set.

The SPI clock will be optionally switched off, during the smart idle mode period, if the second bit ClockActivity of the register MCSPI\_SYSConfig is set.

McSPI assumes that both clocks may be switched off whatever the value set in the field ClockActivity of the register MCSPI\_SYSConfig.

##### 24.3.7.2.1 Transitions from Normal Mode to Smart-Idle Mode

The module detects an idle request when the synchronous signal IdleReq is asserted.

When IdleReq is asserted, any access to the module will generate an error as long as the OCP clock is alive.

When configured as a slave device, McSPI responds to the idle request by asserting the SIdleAck signal (idle acknowledgement) only after completion of the current transfer (SPIEN slave selection signal deasserted by the external master) and if interrupt or DMA request lines are not asserted.

As a master device, McSPI responds to the idle request by asserting the SIdleAck signal (idle acknowledgement) only after completion of all the channel data transfers and if interrupt or DMA request lines are not asserted.

As long as SIdleAck is not asserted, if an event occurs, the module can still generate an interrupt or a DMA request after IdleReq assertion. In this case, the module ignores the idle request and SIdleAck will not get asserted: The system power manager will abort the power mode transition procedure. It is then the responsibility of the system to de-assert IdleReq before attempting to access the module.

When SIdleAck is asserted, the module does not assert any new interrupt or DMA request.

#### **24.3.7.2.2 Transition From Smart-Idle Mode to Normal mode**

McSPI detects the end of the idle period when the idle request signal (IdleReq) is deasserted.

Upon IdleReq de-assertion, the module switches back to normal mode and de-asserts SIdleAck signal. The module is fully operational.

#### **24.3.7.2.3 Force-Idle Mode**

Force-idle mode is enabled as follows:

- The bit SIdleMode of the register MCSPI\_SYS CONFIG is cleared to “00” (Force Idle).

The force idle mode is an idle mode where McSPI responds unconditionally to the idle request by asserting the SIdleAck signal and by deasserting unconditionally the interrupt and DMA request lines if asserted.

The transition from normal mode to idle mode does not affect the interrupt event bits of the MCSPI\_IRQSTATUS register.

In force-idle mode, the module is supposed to be disabled at that time, so the interrupt and DMA request lines are likely deasserted. OCP clock and SPI clock provided to McSPI can be switched off.

An idle request during an SPI data transfer can lead to an unexpected and unpredictable result, and is under software responsibility.

Any access to the module in force idle mode will generate an error as long as the OCP clock is alive and IdleReq is asserted.

The module exits the force idle mode when:

- The idle request signal (IdleReq) is de-asserted.

Upon IdleReq de-assertion, the module switches back to normal mode and de-asserts SIdleAck signal. The module is fully operational. The interrupt and DMA request lines are optionally asserted a clock cycle later.

#### **24.3.8 System Test Mode**

McSPI is in system test mode (SYSTEST) when the bit System\_Test of the register MCSPI\_MODULCTRL is set.

The SYSTEST mode is used to check in a very simple manner the correctness of the system interconnect either internally to interrupt handler, or power manager, or externally to SPI I/Os.

I/O verification can be performed in SYSTEST mode by toggling the outputs and capturing the logic state of the inputs. (See MCSPI\_SYST register definition in [Section 24.4.1.6](#))

#### **24.3.9 Reset**

##### **24.3.9.1 Internal Reset Monitoring**

The module is reset by the hardware when an active-low reset signal, synchronous to the OCP interface clock is asserted on the input pin RESETN.

This hardware reset signal has a global reset action on the module. All configuration registers and all state machines are reset, in all clock domains.

Additionally, the module can be reset by software through the bit SoftReset of the register MCSPI\_SYS CONFIG. This bit has exactly the same action on the module logic as the hardware RESETN signal. The register MCSPI\_SYS CONFIG is not sensitive to software reset. The SoftReset control bit is active high. The bit is automatically reset to 0 by the hardware.

A global ResetDone status bit is provided in the status register MCSPI\_SYSSTATUS. This bit is set to 1 when all the different clock domains resets (OCP domain and SPI domains) have been released (logical AND).

The global ResetDone status bit can be monitored by the software to check if the module is ready-to-use following a reset (either hardware or software).

The clock CLKSPIREF must be provided to the module, in order to allow the ResetDone status bit to be set.

When used in slave mode, the clock CLKSPIREF is needed only during the reset phase. The clock CLKSPIREF can be switched off after the ResetDone status is set.

#### 24.3.9.2 Reset Values of Registers

The reset values of registers and signals are described in [Section 24.4.1](#).

#### 24.3.10 Access to Data Registers

This section details the supported data accesses (read or write) from/to the data receiver registers MCSPI\_RX(i) and data transmitter registers MCSPI\_TX(i).

Supported access:

McSPI supports only one SPI word per register (receiver or transmitter) and does not support successive 8-bit or 16-bit accesses for a single SPI word.

The SPI word received is always right justified on LSbit of the 32bit register MCSPI\_RX(i), and the SPI word to transmit is always right justified on LSbit of the 32bit register MCSPI\_TX(i).

The upper bits, above SPI word length, are ignored and the content of the data registers is not reset between the SPI data transfers.

The coherence between the number of bits of the SPI Word, the number of bits of the access and the enabled byte remains under the user's responsibility. Only aligned accesses are supported.

In Master mode, data should not be written in the transmit register when the channel is disabled.

#### 24.3.11 Programming Aid

##### 24.3.11.1 Module Initialization

- Hard or soft reset.
- Read MCSPI\_SYSSTATUS.
- Check if reset is done.
- Module configuration: (a) Write into MCSPI\_MODULCTRL (b) Write into MCSPI\_SYSCONFIG.
- Before the ResetDone bit is set, the clocks CLK and CLKSPIREF must be provided to the module.
- To avoid hazardous behavior, it is advised to reset the module before changing from MASTER mode to SLAVE mode or from SLAVE mode to MASTER mode.

##### 24.3.11.2 Common Transfer Sequence

McSPI module allows the transfer of one or several words, according to different modes:

- MASTER, MASTER Turbo, SLAVE
- TRANSMIT - RECEIVE, TRANSMIT ONLY, RECEIVE ONLY
- Write and Read requests: Interrupts, DMA
- SPIEN lines assertion/deassertion: automatic, manual

For all these flows, the host process contains the main process and the interrupt routines. The interrupt routines are called on the interrupt signals or by an internal call if the module is used in polling mode.

In multi-channel master mode, the flows of different channels can be run simultaneously.

### 24.3.11.3 Main Program

- Interrupt Initialization: (a) Reset status bits in MCSPI\_IRQSTATUS (b) Enable interrupts in MCSPI\_IRQENA.
- Channel Configuration: Write MCSPI\_CH(i)CONF.
- Start the channel: Write 0000 0001h in MCSPI\_CH(i)CTRL.
- First write request: TX empty - Generate DMA write event/ polling TX empty flag by CPU to write First transmit word into MCSPI\_TX(i).
- End of transfer: Stop the channel by writing 0000 0000h in MCSPI\_CH(i)CTRL

The end of transfer depends on the transfer mode.

In multi-channel master mode, be careful not to overwrite the bits of other channels when initializing MCSPI\_IRQSTATUS and MCSPI\_IRQENABLE.

### 24.3.12 Interrupt and DMA Events

McSPI has two DMA requests (Rx and Tx) per channel. It also has one interrupt line for all the interrupt requests.

## 24.4 McSPI Registers

### 24.4.1 SPI Registers

Table 24-10 lists the McSPI registers.

**Table 24-10. SPI Registers**

| Offset Address | Acronym          | Register Name  | Section                           |
|----------------|------------------|--|-----------------------------------|
| 000h           | MCSPI_REVISION   | McSPI revision register                              | <a href="#">Section 24.4.1.1</a>  |
| 110h           | MCSPI_SYS CONFIG | McSPI system configuration register                  | <a href="#">Section 24.4.1.2</a>  |
| 114h           | MCSPI_SYS STATUS | McSPI system status register                         | <a href="#">Section 24.4.1.3</a>  |
| 118h           | MCSPI_IRQSTATUS  | McSPI interrupt status register                      | <a href="#">Section 24.4.1.4</a>  |
| 11Ch           | MCSPI_IRQENABLE  | McSPI interrupt enable register                      | <a href="#">Section 24.4.1.5</a>  |
| 124h           | MCSPI_SYST       | McSPI system register                                | <a href="#">Section 24.4.1.6</a>  |
| 128h           | MCSPI_MODULCTRL  | McSPI module control register                        | <a href="#">Section 24.4.1.7</a>  |
| 12Ch           | MCSPI_CH0CONF    | McSPI channel <i>i</i> configuration register        | <a href="#">Section 24.4.1.8</a>  |
| 130h           | MCSPI_CH0STAT    | McSPI channel <i>i</i> status register               | <a href="#">Section 24.4.1.9</a>  |
| 134h           | MCSPI_CH0CTRL    | McSPI channel <i>i</i> control register              | <a href="#">Section 24.4.1.10</a> |
| 138h           | MCSPI_TX0        | McSPI channel <i>i</i> FIFO transmit buffer register | <a href="#">Section 24.4.1.11</a> |
| 13Ch           | MCSPI_RX0        | McSPI channel <i>i</i> FIFO receive buffer register  | <a href="#">Section 24.4.1.12</a> |
| 140h           | MCSPI_CH1CONF    | McSPI channel <i>i</i> configuration register        | <a href="#">Section 24.4.1.8</a>  |
| 144h           | MCSPI_CH1STAT    | McSPI channel <i>i</i> status register               | <a href="#">Section 24.4.1.9</a>  |
| 148h           | MCSPI_CH1CTRL    | McSPI channel <i>i</i> control register              | <a href="#">Section 24.4.1.10</a> |
| 14Ch           | MCSPI_TX1        | McSPI channel <i>i</i> FIFO transmit buffer register | <a href="#">Section 24.4.1.11</a> |
| 150h           | MCSPI_RX1        | McSPI channel <i>i</i> FIFO receive buffer register  | <a href="#">Section 24.4.1.12</a> |
| 154h           | MCSPI_CH2CONF    | McSPI channel <i>i</i> configuration register        | <a href="#">Section 24.4.1.8</a>  |
| 158h           | MCSPI_CH2STAT    | McSPI channel <i>i</i> status register               | <a href="#">Section 24.4.1.9</a>  |
| 15Ch           | MCSPI_CH2CTRL    | McSPI channel <i>i</i> control register              | <a href="#">Section 24.4.1.10</a> |
| 160h           | MCSPI_TX2        | McSPI channel <i>i</i> FIFO transmit buffer register | <a href="#">Section 24.4.1.11</a> |
| 164h           | MCSPI_RX2        | McSPI channel <i>i</i> FIFO receive buffer register  | <a href="#">Section 24.4.1.12</a> |
| 168h           | MCSPI_CH3CONF    | McSPI channel <i>i</i> configuration register        | <a href="#">Section 24.4.1.8</a>  |
| 16Ch           | MCSPI_CH3STAT    | McSPI channel <i>i</i> status register register      | <a href="#">Section 24.4.1.9</a>  |
| 170h           | MCSPI_CH3CTRL    | McSPI channel <i>i</i> control register              | <a href="#">Section 24.4.1.10</a> |

**Table 24-10. SPI Registers (continued)**

| Offset Address | Acronym         | Register Name  | Section                           |
|----------------|-----------------|--|-----------------------------------|
| 174h           | MCSPI_TX3       | McSPI channel <i>i</i> FIFO transmit buffer register | <a href="#">Section 24.4.1.11</a> |
| 178h           | MCSPI_RX3       | McSPI channel <i>i</i> FIFO receive buffer register  | <a href="#">Section 24.4.1.12</a> |
| 17Ch           | MCSPI_XFERLEVEL | McSPI transfer levels register                       | <a href="#">Section 24.4.1.13</a> |
| 180h           | MCSPI_DAFTX     | McSPI DMA address aligned FIFO transmitter register  |                                   |
| 1A0h           | MCSPI_DAFRX     | McSPI DMA address aligned FIFO receiver register     | <a href="#">Section 24.4.1.15</a> |

#### 24.4.1.1 McSPI Revision Register (MCSPI\_REVISION)

The McSPI system configuration register (MCSPI\_REVISION) allows control of various parameters of the module interface. It is not sensitive to software reset. The MCSPI\_REVISION register is shown in [Figure 24-26](#) and described in [Table 24-11](#).

**Figure 24-26. McSPI Revision Register (MCSPI\_REVISION)**

| 31     | 30       | 29 | 28 | 27      | 16 |
|--------|----------|----|----|---------|----|
| SCHEME | Reserved |    |    | FUNC    |    |
| R-1    | R-0      |    |    | R-030h  |    |
| 15     |          |    |    | 11      | 10 |
|        | RRTL     |    |    | X_MAJOR | 8  |
|        | R-0      |    |    | R-0     |    |
| 7      | 6        | 5  | 4  | 3       | 2  |
| CUSTOM |          |    |    | Y_MINOR | 1  |
|        | R/W-0    |    |    | R/W-2h  | 0  |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-11. McSPI Revision Register (MCSPI\_REVISION) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-30 | SCHEME   | 0     | Used to distinguish between old scheme and current.  |
|       |          | 1     | Legacy ASP or WTBU scheme  |
|       |          |       | Revision 0.8 scheme  |
| 29-28 | Reserved | 0     | Read returns 0.  |
| 27-16 | FUNC     | 030h  | Function indicates a software compatible module family.<br>If there is no level of software compatibility a new Func number (and hence REVISION) should be assigned. |
| 15-11 | RRTL     |       |  |
| 10-8  | X_MAJOR  |       |  |
| 7-6   | CUSTOM   |       |  |
| 5-0   | Y_MINOR  |       |  |

#### 24.4.1.2 McSPI System Configuration Register (MCSPI\_SYSCONFIG)

The McSPI system configuration register (MCSPI\_SYSCONFIG) allows control of various parameters of the module interface. It is not sensitive to software reset. The MCSPI\_SYSCONFIG is shown in [Figure 24-27](#) and described in [Table 24-12](#).

**Figure 24-27. McSPI System Configuration Register (MCSPI\_SYSCONFIG)**

|    |          |           |   |          |               |          |   |    |
|----|----------|-----------|---|----------|---------------|----------|---|----|
| 31 |          |           |   |          |               |          |   | 16 |
|    | Reserved |           |   |          |               |          |   |    |
|    | R/W-0    |           |   |          |               |          |   |    |
| 15 | Reserved |           |   |          | 10            | 9        | 8 |    |
|    | R/W-0    |           |   |          | CLOCKACTIVITY |          |   |    |
|    | R/W-0    |           |   |          | R/W-0         |          |   |    |
| 7  | 5        | 4         | 3 | 2        | 1             | 0        |   |    |
|    | Reserved | SIDLEMODE |   | Reserved | SOFTRESET     | AUTOIDLE |   |    |
|    | R/W-0    | R/W-2h    |   | R        | R/W-0         | R/W-1    |   |    |

LEGEND: R/W = Read/Write; -n = value after reset

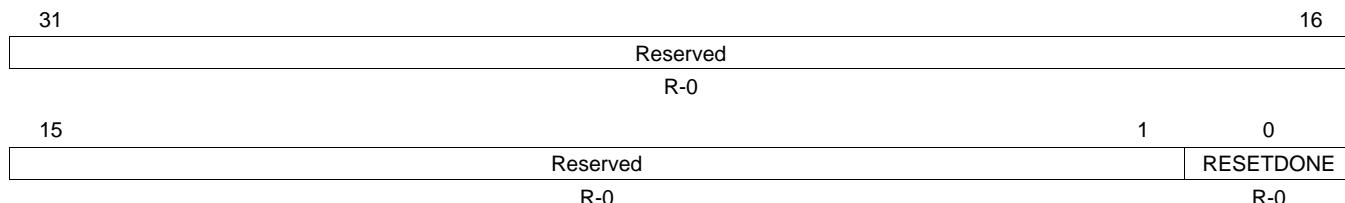
**Table 24-12. McSPI System Configuration Register (MCSPI\_SYSCONFIG) Field Descriptions**

| Bit   | Field         | Value               | Description  |
|-------|---------------|---------------------|--|
| 31-10 | Reserved      | 0                   | Reads returns 0  |
| 9-8   | CLOCKACTIVITY | 0<br>1h<br>2h<br>3h | Clocks activity during wake-up mode period.<br>0 OCP and Functional clocks may be switched off.<br>1h OCP clock is maintained. Functional clock may be switched-off.<br>2h Functional clock is maintained. OCP clock may be switched-off.<br>3h OCP and Functional clocks are maintained.  |
| 7-5   | Reserved      | 0                   | Reads returns 0  |
| 4-3   | SIDLEMODE     | 0<br>1h<br>2h<br>3h | Power management<br>0 If an idle request is detected, the McSPI acknowledges it unconditionally and goes in Inactive mode. Interrupt, DMA requests are unconditionally de-asserted.<br>1h If an idle request is detected, the request is ignored and keeps on behaving normally.<br>2h Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements.<br>3h Reserved |
| 2     | Reserved      | 0                   | Reserved   |
| 1     | SOFTRESET     | 0<br>1              | Software reset. During reads it always returns 0.<br>(write) Normal mode<br>1 (write) Set this bit to 1 to trigger a module reset. The bit is automatically reset by the hardware.   |
| 0     | AUTOIDLE      | 0<br>1              | Internal OCP Clock gating strategy<br>0 OCP clock is free-running<br>1 Automatic OCP clock gating strategy is applied, based on the OCP interface activity   |

#### 24.4.1.3 McSPI System Status Register (MCSPI\_SYSSTATUS)

The McSPI system status register (MCSPI\_SYSSTATUS) provides status information about the module excluding the interrupt status information. The MCSPI\_SYSSTATUS is shown in [Figure 24-28](#) and described in [Table 24-13](#).

**Figure 24-28. McSPI System Status Register (MCSPI\_SYSSTATUS)**



LEGEND: R = Read only; -n = value after reset

**Table 24-13. McSPI System Status Register (MCSPI\_SYSSTATUS) Field Descriptions**

| Bit  | Field     | Value | Description  |
|------|-----------|-------|--|
| 31-1 | Reserved  | 0     | Reserved for module specific status information. Read returns 0. |
| 0    | RESETDONE | 0     | Internal Reset Monitoring  |
|      |           | 1     | Internal module reset is on-going                                |
|      |           |       | Reset completed  |

#### 24.4.1.4 McSPI Interrupt Status Register (MCSPI\_IRQSTATUS)

The McSPI interrupt status register (MCSPI\_IRQSTATUS) regroups all the status of the module internal events that can generate an interrupt. The MCSPI\_IRQSTATUS is shown in [Figure 24-29](#) and described in [Table 24-14](#).

**NOTE:** In SYSTEST mode, the bits of this register have no meaning and always read 0.

**Figure 24-29. McSPI Interrupt Status Register (MCSPI\_IRQSTATUS)**

| 31    | Reserved |               |           |              |          |               |           |     | 18   | 17    | 16  |
|-------|----------|---------------|-----------|--------------|----------|---------------|-----------|-----|------|-------|-----|
| Rsvd  | RX3_FULL | TX3_UNDERFLOW | TX3_EMPTY | Reserved     | RX2_FULL | TX2_UNDERFLOW | TX2_EMPTY | EOW | Rsvd | R/W-0 | R-0 |
| 15    | 14       | 13            | 12        | 11           | 10       | 9             | 8         |     |      |       |     |
| R/W-0 | R/W-0    | R/W-0         | R/W-0     | R/W-0        | R/W-0    | R/W-0         | R/W-0     |     |      |       |     |
| 7     | 6        | 5             | 4         | 3            | 2        | 1             | 0         |     |      |       |     |
| Rsvd  | RX1_FULL | TX1_UNDERFLOW | TX1_EMPTY | RX0_OVERFLOW | RX0_FULL | TX0_UNDERFLOW | TX0_EMPTY |     |      |       |     |
| R/W-0 | R/W-0    | R/W-0         | R/W-0     | R/W-0        | R/W-0    | R/W-0         | R/W-0     |     |      |       |     |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-14. McSPI Interrupt Status Register (MCSPI\_IRQSTATUS) Field Descriptions**

| Bit   | Field         | Value                                  | Description   |
|-------|---------------|--|---|
| 31-18 | Reserved      | 0                                      | Reads returns 0   |
| 17    | EOW           | Write 0<br>Read 0<br>Write 1<br>Read 1 | End of word (EOW) count event when a channel is enabled using the FIFO buffer and the channel has sent the number of McSPI words defined by the MCSPI_XFERLEVEL[WCNT].<br>Event status bit is unchanged.<br>Event false.<br>Event status bit is reset.<br>Event is pending.   |
| 16    | Reserved      | 0                                      | Reserved  |
| 15    | Reserved      | 0                                      | Reads returns 0   |
| 14    | RX3_FULL      | Write 0<br>Read 0<br>Write 1<br>Read 1 | Receiver register is full or almost full. Only when Channel 3 is enabled. This bit indicate FIFO almost full status when built-in FIFO is used for receive register (MCSPI_CH3CONF[FFE3R] is set).<br>Event status bit is unchanged.<br>Event false.<br>Event status bit is reset.<br>Event is pending.   |
| 13    | TX3_UNDERFLOW | Write 0<br>Read 0<br>Write 1<br>Read 1 | Transmitter register underflow. Only when Channel 3 is enabled. The transmitter register is empty (not updated by Host or DMA with new data) before its time slot assignment.<br>Exception: No TX_underflow event when no data has been loaded into the transmitter register since channel has been enabled.<br>Event status bit is unchanged.<br>Event false.<br>Event status bit is reset.<br>Event is pending. |

**Table 24-14. McSPI Interrupt Status Register (MCSPI\_IRQSTATUS) Field Descriptions (continued)**

| Bit | Field         | Value   | Description  |
|-----|---------------|---------|--|
| 12  | TX3_EMPTY     |         | Transmitter register is empty or almost empty. This bit indicate FIFO almost full status when built-in FIFO is used for transmit register (MCSPI_CH3CONF[FFE3W] is set).<br><b>Note:</b> Enabling the channel automatically raises this event. |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 11  | Reserved      | 0       | Reads returns 0  |
| 10  | RX2_FULL      |         | Receiver register full or almost full. Channel 2. This bit indicate FIFO almost full status when built-in FIFO is used for receive register (MCSPI_CH3CONF[FFE2R] is set).   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 9   | TX2_UNDERFLOW |         | Transmitter register underflow. Channel 2  |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 8   | TX2_EMPTY     |         | Transmitter register empty or almost empty. Channel 2. This bit indicate FIFO almost full status when built-in FIFO is used for transmit register (MCSPI_CH3CONF[FFE2W] is set).   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 7   | Reserved      | 0       | Reads returns 0  |
| 6   | RX1_FULL      |         | Receiver register full or almost full. Channel 1. This bit indicate FIFO almost full status when built-in FIFO is used for receive register (MCSPI_CH3CONF[FFE1R] is set).   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 5   | TX1_UNDERFLOW |         | Transmitter register underflow. Channel 1.   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 4   | TX1_EMPTY     |         | Transmitter register empty or almost empty. Channel 1. This bit indicate FIFO almost full status when built-in FIFO is used for transmit register (MCSPI_CH3CONF[FFE1W] is set).   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |
| 3   | RX0_OVERFLOW  |         | Receiver register overflow (slave mode only). Channel 0.   |
|     |               | Write 0 | Event status bit is unchanged.   |
|     |               | Read 0  | Event false.   |
|     |               | Write 1 | Event status bit is reset.   |
|     |               | Read 1  | Event is pending.  |

**Table 24-14. McSPI Interrupt Status Register (MCSPI\_IRQSTATUS) Field Descriptions (continued)**

| Bit | Field         | Value | Description  |
|-----|---------------|-------|--|
| 2   | RX0_FULL      |       | Receiver register full or almost full. Channel 0. Receiver register full or almost full. Channel 0<br>Write 0 Event status bit is unchanged.<br>Read 0 Event false.<br>Write 1 Event status bit is reset.<br>Read 1 Event is pending.  |
| 1   | TX0_UNDERFLOW |       | Transmitter register underflow. Channel 0.<br>Write 0 Event status bit is unchanged.<br>Read 0 Event false.<br>Write 1 Event status bit is reset.<br>Read 1 Event is pending.  |
| 0   | TX0_EMPTY     |       | Transmitter register empty or almost empty. Channel 0. This bit indicate FIFO almost full status when built-in FIFO is use for transmit register (MCSPI_CH3CONF[FFE0W] is set).<br>Write 0 Event status bit is unchanged.<br>Read 0 Event false.<br>Write 1 Event status bit is reset.<br>Read 1 Event is pending. |

#### 24.4.1.5 McSPI Interrupt Enable Register (MCSPI\_IRQENABLE)

This McSPI interrupt enable register (MCSPI\_IRQENABLE) enables/disables the module internal sources of interrupt, on an event-by-event basis. The MCSPI\_IRQENABLE is shown in [Figure 24-30](#) and described in [Table 24-15](#).

**Figure 24-30. McSPI Interrupt Enable Register (MCSPI\_IRQENABLE)**

| 31    | Reserved        |                      |                  |                     |                 |                      |                  |       | 18    | 17    | 16    |
|-------|-----------------|----------------------|------------------|---------------------|-----------------|----------------------|------------------|-------|-------|-------|-------|
|       | R/W-0           |                      |                  |                     |                 |                      |                  |       | R/W-0 | R/W-0 | R-0   |
| 15    | 14              | 13                   | 12               | 11                  | 10              | 9                    | 8                |       |       |       |       |
| Rsvd  | RX3_FULL_ENABLE | TX3_UNDERFLOW_ENABLE | TX3_EMPTY_ENABLE | Reserved            | RX2_FULL_ENABLE | TX2_UNDERFLOW_ENABLE | TX2_EMPTY_ENABLE |       |       |       |       |
| R/W-0 | R/W-0           | R/W-0                | R/W-0            | R/W-0               | R/W-0           | R/W-0                | R/W-0            | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7     | 6               | 5                    | 4                | 3                   | 2               | 1                    | 0                |       |       |       |       |
| Rsvd  | RX1_FULL_ENABLE | TX1_UNDERFLOW_ENABLE | TX1_EMPTY_ENABLE | RX0_OVERFLOW_ENABLE | RX0_FULL_ENABLE | TX0_UNDERFLOW_ENABLE | TX0_EMPTY_ENABLE |       |       |       |       |
| R/W-0 | R/W-0           | R/W-0                | R/W-0            | R/W-0               | R/W-0           | R/W-0                | R/W-0            | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-15. McSPI Interrupt Enable Register (MCSPI\_IRQENABLE) Field Descriptions**

| Bit   | Field                | Value | Description  |
|-------|----------------------|-------|--|
| 31-18 | Reserved             | 0     | Reads return 0   |
| 17    | EOWKE                | 0     | End of word count interrupt enable.<br>Interrupt is disabled.  |
|       |                      | 1     | Interrupt is enabled.  |
| 16    | Reserved             |       | Reserved   |
| 15    | Reserved             | 0     | Reads return 0   |
| 14    | RX3_FULL_ENABLE      | 0     | MCSPI_RX3 receiver register full or almost full interrupt enable (channel 3).<br>Interrupt is disabled.      |
|       |                      | 1     | Interrupt is enabled.  |
| 13    | TX3_UNDERFLOW_ENABLE | 0     | MCSPI_TX3 transmitter register underflow interrupt enable (channel 3).<br>Interrupt is disabled.             |
|       |                      | 1     | Interrupt is enabled.  |
| 12    | TX3_EMPTY_ENABLE     | 0     | MCSPI_TX3 transmitter register empty or almost empty interrupt enable (channel 3).<br>Interrupt is disabled. |
|       |                      | 1     | Interrupt is enabled.  |
| 11    | Reserved             | 0     | Reads return 0   |
| 10    | RX2_FULL_ENABLE      | 0     | MCSPI_RX2 receiver register full or almost full interrupt enable (channel 2).<br>Interrupt is disabled.      |
|       |                      | 1     | Interrupt is enabled.  |
| 9     | TX2_UNDERFLOW_ENABLE | 0     | MCSPI_TX2 transmitter register underflow interrupt enable (channel 2).<br>Interrupt is disabled.             |
|       |                      | 1     | Interrupt is enabled.  |
| 8     | TX2_EMPTY_ENABLE     | 0     | MCSPI_TX2 transmitter register empty or almost empty interrupt enable (channel 2).<br>Interrupt is disabled. |
|       |                      | 1     | Interrupt is enabled.  |
| 7     | Reserved             | 0     | Reads return 0   |

**Table 24-15. McSPI Interrupt Enable Register (MCSPI\_IRQENABLE) Field Descriptions (continued)**

| Bit | Field                | Value | Description  |
|-----|----------------------|-------|--|
| 6   | RX1_FULL_ENABLE      | 0     | MCSPI_RX1 receiver register full or almost full interrupt enable (channel 1).<br>Interrupt is disabled.      |
|     |                      | 1     | Interrupt is enabled.  |
| 5   | TX1_UNDERFLOW_ENABLE | 0     | MCSPI_TX1 transmitter register underflow interrupt enable (channel 1).<br>Interrupt is disabled.             |
|     |                      | 1     | Interrupt is enabled.  |
| 4   | TX1_EMPTY_ENABLE     | 0     | MCSPI_TX1 transmitter register empty or almost empty interrupt enable (channel 1).<br>Interrupt is disabled. |
|     |                      | 1     | Interrupt is enabled.  |
| 3   | RX0_OVERFLOW_ENABLE  | 0     | MCSPI_RX0 receiver register overflow interrupt enable (channel 0).<br>Interrupt is disabled.                 |
|     |                      | 1     | Interrupt is enabled.  |
| 2   | RX0_FULL_ENABLE      | 0     | MCSPI_RX0 receiver register full or almost full interrupt enable (channel 0).<br>Interrupt is disabled.      |
|     |                      | 1     | Interrupt is enabled.  |
| 1   | TX0_UNDERFLOW_ENABLE | 0     | MCSPI_TX0 transmitter register underflow interrupt enable (channel 0).<br>Interrupt is disabled.             |
|     |                      | 1     | Interrupt is enabled.  |
| 0   | TX0_EMPTY_ENABLE     | 0     | MCSPI_TX0 transmitter register empty or almost empty interrupt enable (channel 0).<br>Interrupt is disabled. |
|     |                      | 1     | Interrupt is enabled.  |

#### 24.4.1.6 McSPI System Register (MCSPI\_SYST)

This McSPI system register (MCSPI\_SYST) is used to configure the system interconnect either internally to the peripheral bus or externally to the device I/O pads, when the module is configured in the system test (SYSTEST) mode. The MCSPI\_SYST is shown in [Figure 24-31](#) and described in [Table 24-16](#).

**Figure 24-31. McSPI System Register (MCSPI\_SYST)**

|          |          |          |          |         |         |         |         |  |    |
|----------|----------|----------|----------|---------|---------|---------|---------|--|----|
| 31       | Reserved |          |          |         |         |         |         |  | 16 |
|          | R/W-0    |          |          |         |         |         |         |  |    |
| 15       | Reserved |          | 12       | 11      | 10      | 9       | 8       |  |    |
|          | R/W-0    |          |          | R/W-0   | R/W-0   | R/W-0   | R/W-0   |  |    |
| 7        | 6        | 5        | 4        | 3       | 2       | 1       | 0       |  |    |
| Reserved | SPICLK   | SPIDAT_1 | SPIDAT_0 | SPIEN_3 | SPIEN_2 | SPIEN_1 | SPIEN_0 |  |    |
| R-0      | R/W-0    | R/W-0    | R/W-0    | R/W-0   | R/W-0   | R/W-0   | R/W-0   |  |    |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-16. McSPI System Register (MCSPI\_SYST) Field Descriptions**

| Bit   | Field      | Value  | Description  |
|-------|------------|--------|--|
| 31-12 | Reserved   | 0      | Reads returns 0  |
| 11    | SSB        | 0<br>1 | Set status bit. This bit must be cleared prior attempting to clear a status bit of the MCSPI_IRQSTATUS register.<br><br>No action. Writing 0 does not clear already set status bits.<br><br>This bit must be cleared prior attempting to clear a status bit of the <MCSPI_IRQSTATUS> register.<br><br>Writing 1 sets to 1 all status bits contained in the MCSPI_IRQSTATUS register.<br><br>Writing 1 into this bit sets to 1 all status bits contained in the <MCSPI_IRQSTATUS> register. |
| 10    | SPIENDIR   | 0<br>1 | Sets the direction of the SPIEN[3:0] lines and SPICLK line.<br><br>Output (as in master mode).<br><br>Input (as in slave mode).  |
| 9     | SPIDATDIR1 | 0<br>1 | Sets the direction of the SPIDAT[1].<br><br>Output<br><br>Input  |
| 8     | SPIDATDIR0 | 0<br>1 | Sets the direction of the SPIDAT[0].<br><br>Output<br><br>Input  |
| 7     | Reserved   | 0      | Reserved   |
| 6     | SPICLK     |        | SPICLK line (signal data value)<br><br>If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the CLKSPI line (high or low), and a write into this bit has no effect.<br><br>If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the CLKSPI line is driven high or low according to the value written into this register.  |
| 5     | SPIDAT_1   |        | SPIDAT[1] line (signal data value)<br><br>If MCSPI_SYST[SPIDATDIR1] = 0 (output mode direction), the SPIDAT[1] line is driven high or low according to the value written into this register.<br><br>If MCSPI_SYST[SPIDATDIR1] = 1 (input mode direction), this bit returns the value on the SPIDAT[1] line (high or low), and a write into this bit has no effect.   |
| 4     | SPIDAT_0   |        | SPIDAT[0] line (signal data value)<br><br>If MCSPI_SYST[SPIDATDIR0] = 0 (output mode direction), the SPIDAT[0] line is driven high or low according to the value written into this register.<br><br>If MCSPI_SYST[SPIDATDIR0] = 1 (input mode direction), this bit returns the value on the SPIDAT[0] line (high or low), and a write into this bit has no effect.   |

**Table 24-16. McSPI System Register (MCSPI\_SYST) Field Descriptions (continued)**

| Bit | Field   | Value | Description  |
|-----|---------|-------|--|
| 3   | SPIEN_3 |       | SPIEN[3] line (signal data value)<br>If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the SPIENT[3] line is driven high or low according to the value written into this register.<br>If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the SPIEN[3] line (high or low), and a write into this bit has no effect. |
| 2   | SPIEN_2 |       | SPIEN[2] line (signal data value)<br>If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the SPIENT[2] line is driven high or low according to the value written into this register.<br>If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the SPIEN[2] line (high or low), and a write into this bit has no effect. |
| 1   | SPIEN_1 |       | SPIEN[1] line (signal data value)<br>If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the SPIENT[1] line is driven high or low according to the value written into this register.<br>If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the SPIEN[1] line (high or low), and a write into this bit has no effect. |
| 0   | SPIEN_0 |       | SPIEN[0] line (signal data value)<br>If MCSPI_SYST[SPIENDIR] = 0 (output mode direction), the SPIENT[0] line is driven high or low according to the value written into this register.<br>If MCSPI_SYST[SPIENDIR] = 1 (input mode direction), this bit returns the value on the SPIEN[0] line (high or low), and a write into this bit has no effect. |

#### 24.4.1.7 McSPI Module Control Register (MCSPI\_MODULCTRL)

This McSPI module control register (MCSPI\_MODULCTRL) is used to configure the serial port interface. The MCSPI\_MODULCTRL is shown in [Figure 24-32](#) and described in [Table 24-17](#).

**Figure 24-32. McSPI Module Control Register (MCSPI\_MODULCTRL)**

|       |          |             |       |       |        |   |  |       |
|-------|----------|-------------|-------|-------|--------|---|--|-------|
| 31    | Reserved |             |       |       |        |   |  | 16    |
|       | R/W-0    |             |       |       |        |   |  |       |
| 15    | Reserved |             |       |       |        |   |  | 8     |
|       | R/W-0    |             |       |       |        |   |  | R/W-0 |
| 7     | 6        | 4           | 3     | 2     | 1      | 0 |  |       |
| MOA   | INITDLY  | SYSTEM_TEST | MS    | PIN34 | SINGLE |   |  |       |
| R/W-0 | R/W-0    | R/W-0       | R/W-0 | R/W-0 | R/W-0  |   |  |       |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-17. McSPI Module Control Register(MCSPI\_MODULCTRL) Field Descriptions**

| Bit  | Field       | Value | Description  |
|------|-------------|-------|--|
| 31-9 | Reserved    | 0     |  |
| 8    | FDAA        |       | FIFO DMA Address 256-bit aligned. This register is used when a FIFO is managed by the module and DMA connected to the controller provides only 256 bit aligned address. If this bit is set the enabled channel which uses the FIFO has its datas managed through MCSPI_DAFTX and MCSPI_DAFRX registers instead of MCSPI_TX(i) and MCSPI_RX(i) registers.                                       |
|      |             |       | 0 FIFO data managed by MCSPI_TX(i) and MCSPI_RX(i) registers.  |
|      |             |       | 1 FIFO data managed by MCSPI_DAFTX and MCSPI_DAFRX registers.  |
| 7    | MOA         |       | Multiple word ocp access. This register can only be used when a channel is enabled using a FIFO. It allows the system to perform multiple SPI word access for a single 32-bit OCP word access. This is possible for WL < 16.   |
|      |             |       | 0 Multiple word access disabled  |
|      |             |       | 1 Multiple word access enabled with FIFO   |
| 6-4  | INITDLY     |       | Initial SPI delay for first transfer. This register is an option only available in SINGLE master mode, The controller waits for a delay to transmit the first SPI word after channel enabled and corresponding TX register filled. This Delay is based on SPI output frequency clock, No clock output provided to the boundary and chip select is not active in 4 pin mode within this period. |
|      |             |       | 0 No delay for first SPI transfer  |
|      |             |       | 1h The controller wait 4 SPI bus clock   |
|      |             |       | 2h The controller wait 8 SPI bus clock   |
|      |             |       | 3h The controller wait 16 SPI bus clock  |
|      |             |       | 4h The controller wait 32 SPI bus clock  |
|      |             |       | 5h-Fh Reserved   |
| 3    | SYSTEM_TEST |       | Enables the system test mode   |
|      |             |       | 0 Functional mode  |
| 2    | MS          |       | System test mode (SYSTEST)   |
|      |             |       | Master/ Slave  |
|      |             |       | 0 Master - The module generates the SPICLK and SPIEN[3:0]  |
| 1    | PIN34       |       | Slave - The module receives the SPICLK and SPIEN[3:0]  |
|      |             |       | 0 SPIEN is used as a chip select.  |
|      |             |       | 1 SPIEN is not used.<br>In this mode all related option to chip select have no meaning.  |

**Table 24-17. McSPI Module Control Register(MCSPIMODULCTRL) Field Descriptions (continued)**

| Bit | Field  | Value | Description   |
|-----|--------|-------|---|
| 0   | SINGLE | 0     | Single channel / Multi Channel (master mode only)<br>More than one channel will be used in master mode. |
|     |        | 1     | Only one channel will be used in master mode. This bit must be set in Force SPIEN mode.                 |

#### 24.4.1.8 McSPI Channel (*i*) Configuration Register (MCSPI\_CH(*i*)CONF)

The McSPI channel *i* configuration register (MCSPI\_CH(*i*)CONF) is used to configure channel *i*. The (MCSPI\_CH(*i*)CONF) is shown in [Figure 24-33](#) and described in [Table 24-18](#).

**Figure 24-33. McSPI Channel (*i*) Configuration Register (MCSPI\_CH(*i*)CONF)**

| 31       | 30    | 29       | 28    | 27    | 26    | 25    | 24    |
|----------|-------|----------|-------|-------|-------|-------|-------|
| Reserved |       | CLKG     | FFER  | FFEW  |       | TCS   | SBPOL |
| R/W-0    |       | R/W-0    | R/W-0 | R/W-0 |       | R/W-0 | R/W-0 |
| 23       | 22    | 21       | 20    | 19    | 18    | 17    | 16    |
| SBE      |       | SPIENSLV | FORCE | TURBO | IS    | DPE1  | DPE0  |
| R/W-0    |       | R/W-0    | R/W-0 | R/W-0 | R/W-1 | R/W-1 | R/W-0 |
| 15       | 14    | 13       | 12    | 11    |       |       | 8     |
| DMAR     | DMAW  |          | TRM   |       |       | WL    |       |
| R/W-0    | R/W-0 |          | R/W-0 |       |       | R/W-0 |       |
| 7        | 6     | 5        |       |       | 2     | 1     | 0     |
| WL       | EPOL  |          |       | CLKD  |       | POL   | PHA   |
| R/W-0    | R/W-0 |          |       | R/W-0 |       | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 24-18. McSPI Channel (*i*) Configuration Register (MCSPI\_CH(*i*)CONF) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-30 | Reserved | 0     | Read returns 0  |
| 29    | CLKG     |       | Clock divider granularity. This register defines the granularity of channel clock divider: power of two or one clock cycle granularity.<br><br>When this bit is set the register MCSPI_CHCTRL[EXTCLK] must be configured to reach a maximum of 4096 clock divider ratio. Then The clock divider ratio is a concatenation of MCSPI_CHCONF[CLKD] and MCSPI_CHCTRL[EXTCLK] values.<br><br>0 Clock granularity of power of 2<br>1 1 clock cycle granularity |
| 28    | FFER     |       | FIFO enabled for receive. Only one channel can have this bit set.<br><br>0 The buffer is not used to receive data.<br>1 The buffer is used to receive data.   |
| 27    | FFEW     |       | FIFO enabled for transmit. Only one channel can have this bit set.<br><br>0 The buffer is not used to transmit data.<br>1 The buffer is used to transmit data.  |
| 26-25 | TCS      |       | Chip select time control. This 2-bits field defines the number of interface clock cycles between CS toggling and first or last edge of SPI clock.<br><br>0 0.5 clock cycles<br>1h 1.5 clock cycles<br>2h 2.5 clock cycles<br>3h 3.5 clock cycles  |
| 24    | SBPOL    |       | Start bit polarity.<br><br>0 Start bit polarity is held to 0 during SPI transfer.<br>1 Start bit polarity is held to 1 during SPI transfer.   |
| 23    | SBE      |       | Start bit enable for SPI transfer.<br><br>0 Default SPI transfer length as specified by WL bit field.<br>1 Start bit D/CX added before SPI transfer. Polarity is defined by MCSPI_CH( <i>i</i> )CONF[SBPOL].  |

**Table 24-18. McSPI Channel (i) Configuration Register (MCSPI\_CH(i)CONF) Field Descriptions (continued)**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 22-21 | SPIENSLV |       | Channel 0 only and slave mode only: SPI slave select signal detection. Reserved bits (read returns 0) for other cases.<br>0 Detection enabled only on SPIEN[0]<br>1h Detection enabled only on SPIEN[1]<br>2h Detection enabled only on SPIEN[2]<br>3h Detection enabled only on SPIEN[3] |
| 20    | FORCE    | 0     | Manual SPIEN assertion to keep SPIEN active between SPI words. (single channel master mode only)<br>Writing 0 into this bit drives the SPIEN line when MCSPI_CHCONF(i)[EPOL]=0, and drives it high when MCSPI_CHCONF(i)[EPOL]=1.  |
|       |          | 1     | Writing 1 into this bit drives the SPIEN line when MCSPI_CHCONF(i)[EPOL]=0, and drives it low when MCSPI_CHCONF(i)[EPOL]=1  |
| 19    | TURBO    | 0     | Turbo mode.<br>Turbo is deactivated (recommended for single SPI word transfer).   |
|       |          | 1     | Turbo is activated to maximize the throughput for multi-SPI word transfers.   |
| 18    | IS       | 0     | Input select<br>Data line 0 (SPIDAT[0]) selected for reception.   |
|       |          | 1     | Data line 1 (SPIDAT[1]) selected for reception.   |
| 17    | DPE1     | 0     | Transmission enable for data line 1 (SPIDATAGZEN[1])<br>Data line 1 (SPIDAT[1]) selected for transmission   |
|       |          | 1     | No transmission on data line 1 (SPIDAT[1])  |
| 16    | DPE0     | 0     | Transmission enable for data line 0 (SPIDATAGZEN[0])<br>Data line 0 (SPIDAT[0]) selected for transmission   |
|       |          | 1     | No transmission on data line 0 (SPIDAT[0])  |
| 15    | DMAR     | 0     | DMA read request. The DMA read request line is asserted when the channel is enabled and new data is available in the receive register of the channel. The DMA read request line is deasserted on read completion of the receive register of the channel.<br>DMA read request is disabled. |
|       |          | 1     | DMA read request is enabled.  |
| 14    | DMAW     | 0     | DMA write request. The DMA write request line is asserted when the channel is enabled and the MCSPI_TXn register of the channel is empty. The DMA write request line is deasserted on load completion of the MCSPI_TXn register of the channel.<br>DMA write request is disabled.         |
|       |          | 1     | DMA write request is enabled.   |
| 13-12 | TRM      | 0     | Transmit/receive modes.<br>Transmit and receive mode  |
|       |          | 1h    | Receive-only mode   |
|       |          | 2h    | Transmit-only mode  |
|       |          | 3h    | Reserved  |

**Table 24-18. McSPI Channel (*i*) Configuration Register (MCSPI\_CH(*i*)CONF) Field Descriptions (continued)**

| Bit  | Field | Value | Description                                 |
|------|-------|-------|---|
| 11-7 | WL    | 0     | SPI word length.                            |
|      |       | 0     | Reserved                                    |
|      |       | 1h    | Reserved                                    |
|      |       | 2h    | Reserved                                    |
|      |       | 3h    | The SPI word is 4-bits long.                |
|      |       | 4h    | The SPI word is 5-bits long                 |
|      |       | 5h    | The SPI word is 6-bits long                 |
|      |       | 6h    | The SPI word is 7-bits long                 |
|      |       | 7h    | The SPI word is 8-bits long                 |
|      |       | 8h    | The SPI word is 9-bits long                 |
|      |       | 9h    | The SPI word is 10-bits long                |
|      |       | Ah    | The SPI word is 11-bits long                |
|      |       | Bh    | The SPI word is 12-bits long                |
|      |       | Ch    | The SPI word is 13-bits long                |
|      |       | Dh    | The SPI word is 14-bits long                |
|      |       | Eh    | The SPI word is 15-bits long                |
|      |       | Fh    | The SPI word is 16-bits long                |
|      |       | 10h   | The SPI word is 17-bits long                |
|      |       | 11h   | The SPI word is 18-bits long                |
|      |       | 12h   | The SPI word is 19-bits long                |
|      |       | 13h   | The SPI word is 20-bits long                |
|      |       | 14h   | The SPI word is 21-bits long                |
|      |       | 15h   | The SPI word is 22-bits long                |
|      |       | 16h   | The SPI word is 23-bits long                |
|      |       | 17h   | The SPI word is 24-bits long                |
|      |       | 18h   | The SPI word is 25-bits long                |
|      |       | 19h   | The SPI word is 26-bits long                |
|      |       | 1Ah   | The SPI word is 27-bits long                |
|      |       | 1Bh   | The SPI word is 28-bits long                |
|      |       | 1Ch   | The SPI word is 29-bits long                |
|      |       | 1Dh   | The SPI word is 30-bits long                |
|      |       | 1Eh   | The SPI word is 31-bits long                |
|      |       | 1Fh   | The SPI word is 32-bits long                |
| 6    | EPOL  | 0     | SPIEN polarity                              |
|      |       | 0     | SPIEN is held high during the active state. |
|      |       | 1     | SPIEN is held low during the active state.  |

**Table 24-18. McSPI Channel (i) Configuration Register (MCSPI\_CH(i)CONF) Field Descriptions (continued)**

| Bit | Field | Value | Description  |
|-----|-------|-------|--|
| 5-2 | CLKD  |       | Frequency divider for SPICLK. (only when the module is a Master SPI device). A programmable clock divider divides the SPI reference clock (CLKSPIREF) with a 4-bit value, and results in a new clock SPICLK available to shift-in and shift-out data. By default the clock divider ratio has a power of two granularity when MCSPI_CHCONF[CLKG] is cleared, Otherwise this register is the 4 LSB bit of a 12-bit register concatenated with clock divider extension MCSPI_CHCTRL[EXTCLK] register. |
|     |       | 0     | The value description below defines the clock ratio when MCSPI_CHCONF[CLKG] is cleared to 0.   |
|     |       | 1h    | Divide by 1.   |
|     |       | 2h    | 2  |
|     |       | 3h    | 4  |
|     |       | 4h    | 8  |
|     |       | 5h    | 16   |
|     |       | 6h    | 32   |
|     |       | 7h    | 64   |
|     |       | 8h    | 128  |
|     |       | 9h    | 256  |
|     |       | Ah    | 512  |
|     |       | Bh    | 1024   |
|     |       | Ch    | 2048   |
|     |       | Dh    | 4096   |
|     |       | Eh    | 8192   |
|     |       | Fh    | 16384  |
|     |       |       | 32768  |
| 1   | POL   | 0     | SPICLK polarity  |
|     |       | 1     | SPICLK is held high during the active state  |
| 0   | PHA   | 0     | SPICLK phase   |
|     |       | 1     | Data are latched on odd numbered edges of SPICLK   |
|     |       |       | Data are latched on even numbered edges of SPICLK  |

**Table 24-19. Data Lines Configurations**

| TRMi |       |       |   |              |   |
|------|-------|-------|---|--------------|---|
| ISi  | DPEi1 | DPEi0 | Transmit and Receive                    | Receive Only | Transmit Only                           |
| 0    | 0     | 0     | Supported                               | Supported    | Supported                               |
| 0    | 0     | 1     | Supported                               | Supported    | Supported                               |
| 0    | 1     | 0     | Supported                               | Supported    | Supported                               |
| 0    | 1     | 1     | Not supported<br>(unpredictable result) | Supported    | Not supported<br>(unpredictable result) |
| 1    | 0     | 0     | Supported                               | Supported    | Supported                               |
| 1    | 0     | 1     | Supported                               | Supported    | Supported                               |
| 1    | 1     | 0     | Supported                               | Supported    | Supported                               |
| 1    | 1     | 1     | Not supported<br>(unpredictable result) | Supported    | Not supported<br>(unpredictable result) |

#### 24.4.1.9 McSPI Channel (*i*) Status Register (MCSPI\_CH(*i*)STAT)

The McSPI channel *i* status register register (MCSPI\_CH(*i*)STAT) provides status information about the McSPI channel *i* FIFO transmit buffer register (MCSPI\_TX*n*) and the McSPI channel *i* FIFO receive buffer register (MCSPI\_RX*n*) of channel *i*. The (MCSPI\_CH(*i*)STAT) is shown in [Figure 24-34](#) and described in [Table 24-20](#).

**Figure 24-34. McSPI Channel (*i*) Status Register (MCSPI\_CH(*i*)STAT)**

| Reserved |       |       |       |       |     |     |     | 8  |
|----------|-------|-------|-------|-------|-----|-----|-----|----|
| R-0      |       |       |       |       |     |     |     | 31 |
| 7        | 6     | 5     | 4     | 3     | 2   | 1   | 0   |    |
| Reserved | RXFFF | RXXFE | TXFFF | TXFFE | EOT | TXS | RXS |    |
| R-0      | R-0   | R-0   | R-0   | R-0   | R-0 | R-0 | R-0 |    |

LEGEND: R = Read only; -n = value after reset

**Table 24-20. McSPI Channel (*i*) Status Register (MCSPI\_CH(*i*)STAT) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-7 | Reserved | 0     | Read returns 0  |
| 6    | RXFFF    | 0     | Channel <i>i</i> FIFO receive buffer full status.   |
|      |          | 0     | FIFO receive buffer is not full.  |
|      |          | 1     | FIFO receive buffer is full.  |
| 5    | RXXFE    | 0     | Channel <i>i</i> FIFO receive buffer empty status.  |
|      |          | 0     | FIFO receive buffer is not empty.   |
|      |          | 1     | FIFO receive buffer is empty.   |
| 4    | TXFFF    | 0     | Channel <i>i</i> FIFO transmit buffer full status.  |
|      |          | 0     | FIFO transmit buffer is not full.   |
|      |          | 1     | FIFO transmit buffer is full.   |
| 3    | TXFFE    | 0     | Channel <i>i</i> FIFO transmit buffer empty status.   |
|      |          | 0     | FIFO transmit buffer is not empty.  |
|      |          | 1     | FIFO transmit buffer is empty.  |
| 2    | EOT      | 0     | Channel <i>i</i> end-of-transfer status. The definitions of beginning and end of transfer vary with master versus slave and the transfer format (transmit/receive mode, turbo mode).  |
|      |          | 0     | This flag is automatically cleared when the shift register is loaded with the data from the transmitter register (beginning of transfer).   |
|      |          | 1     | This flag is automatically set to one at the end of an SPI transfer.  |
| 1    | TXS      | 0     | Channel <i>i</i> transmitter register status. The bit is cleared when the host writes the most significant byte of the SPI word in the MCSPI_TX( <i>i</i> ) register. The bit is set when enabling the channel <i>i</i> , and also when the SPI word is transferred from the MCSPI_TX( <i>i</i> ) register to the shift register.                 |
|      |          | 0     | Register is full.   |
|      |          | 1     | Register is empty.  |
| 0    | RXS      | 0     | Channel <i>i</i> receiver register status. The bit is cleared when enabling the channel <i>i</i> , and also when the host reads the most significant byte of the received SPI word from the MCSPI_RX( <i>i</i> ) register. The bit is set when the received SPI word is transferred from the shift register to the MCSPI_RX( <i>i</i> ) register. |
|      |          | 0     | Register is empty.  |
|      |          | 1     | Register is full.   |

#### 24.4.1.10 McSPI Channel (*i*) Control Register (MCSPI\_CH(*i*)CTRL)

The McSPI channel *i* control register (MCSPI\_CH/*i*CTRL) is used to enable channel *i*. The MCSPI\_CH/*i*CTRL is shown in [Figure 24-35](#) and described in [Table 24-21](#).

**Figure 24-35. McSPI Channel (*i*) Control Register (MCSPI\_CH(*i*)CTRL)**

|    |          |          |             |
|----|----------|----------|-------------|
| 31 | Reserved |          | 16          |
|    | R/W-0    |          |             |
| 15 | EXTCLK   | 8      7 | 1      0    |
|    | R/W-0    | Reserved | EN<br>R/W-0 |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-21. McSPI Channel (*i*) Control Register (MCSPI\_CH(*i*)CTRL) Field Descriptions**

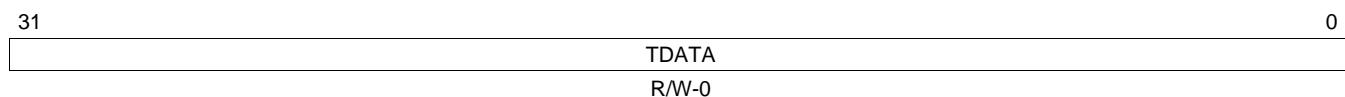
| Bit   | Field    | Value                 | Description   |
|-------|----------|-----------------------|---|
| 31-16 | Reserved | 0                     | Reserved  |
| 15-8  | EXTCLK   | 0<br>1h<br>...<br>FFh | Clock ratio extension. Used to concatenate with the CLKD bit field in MCSPI_CH <i>n</i> CONF for clock ratio only when granularity is 1 clock cycle (CLKG bit in MCSPI_CH <i>n</i> CONF set to 1). Then the maximum value reached is a 4096 clock divider ratio.<br><br>0 Clock ratio is CLKD + 1<br>1h Clock ratio is CLKD + 1 + 16<br>...<br>FFh Clock ratio is CLKD + 1 + 4080 |
| 7-1   | Reserved | 0                     | Reserved  |
| 0     | EN       | 0<br>1                | Channel <i>n</i> enable.<br><br>0 Channel <i>n</i> is not active.<br>1 Channel <i>n</i> is active.  |

#### 24.4.1.11 McSPI Channel (*i*) Transmit Register (MCSPI\_TX(*i*))

The McSPI channel *i* transmit register (MCSPI\_TX(*i*)) contains a single McSPI word to transmit on the serial link. The (MCSPI\_TX(*i*)) is shown in [Figure 24-36](#) and described in [Table 24-22](#).

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.
- The SPI words are transferred with MSB first.

**Figure 24-36. McSPI Channel (*i*) Transmit Register (MCSPI\_TX(*i*))**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-22. McSPI Channel (*i*) Transmit Register (MCSPI\_TX(*i*)) Field Descriptions**

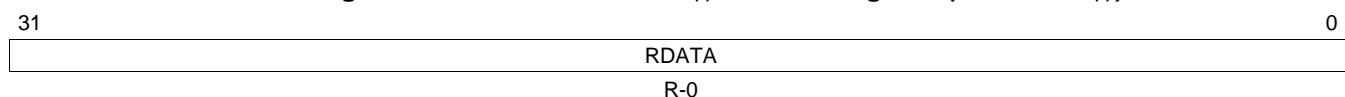
| Bit  | Field | Value        | Description                        |
|------|-------|--------------|------------------------------------|
| 31-0 | TDATA | 0xFFFF FFFFh | Channel <i>i</i> data to transmit. |

#### 24.4.1.12 McSPI Channel (*i*) Receive Register (MCSPI\_RX(*i*))

The McSPI channel *i* FIFO receive buffer register (MCSPI\_RX(*i*)) contains a single McSPI word received through the serial link. The (MCSPI\_RX(*i*)) is shown in [Figure 24-37](#) and described in [Table 24-23](#).

- Little endian host access SPI 8 bit word on 0; big endian host accesses on 3h.

**Figure 24-37. McSPI Channel (*i*) Receive Register (MCSPI\_RX(*i*))**



LEGEND: R = Read only; -n = value after reset

**Table 24-23. McSPI Channel (*i*) Receive Register (MCSPI\_RX(*i*)) Field Descriptions**

| Bit  | Field | Value        | Description                     |
|------|-------|--------------|---------------------------------|
| 31-0 | RDATA | 0xFFFF FFFFh | Channel <i>i</i> received data. |

#### 24.4.1.13 McSPI Transfer Levels Register (MCSPI\_XFERLEVEL)

The McSPI transfer levels register (MCSPI\_XFERLEVEL) provides the transfer levels needed while using the FIFO buffer during transfer. The MCSPI\_XFERLEVEL is shown in [Figure 24-38](#) and described in [Table 24-24](#).

**Figure 24-38. McSPI Transfer Levels Register (MCSPI\_XFERLEVEL)**

|          |       |    |       |          |     |   |       |  |    |
|----------|-------|----|-------|----------|-----|---|-------|--|----|
| 31       | WCNT  |    |       |          |     |   |       |  | 16 |
|          | R/W-0 |    |       |          |     |   |       |  |    |
| 15       | 14    | 13 | 8     | 7        | 6   | 5 | 0     |  |    |
| Reserved | AFL   |    |       | Reserved | AEL |   |       |  |    |
|          | R-0   |    | R/W-0 |          | R-0 |   | R/W-0 |  |    |

LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-24. McSPI Transfer Levels Register (MCSPI\_XFERLEVEL) Field Descriptions**

| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-16 | WCNT     | 0     | SPI word counter. Holds the programmable value of the number of SPI words to be transferred on the channel that is using the FIFO buffer. When the transfer has started, a read back of this register returns the current SPI word transfer index.   |
|       |          | 1     | Counter not used   |
|       |          | ...   | ...  |
|       |          | FFFEh | 65534 SPI word   |
|       |          | FFFFh | 65535 SPI word   |
| 15-14 | Reserved | 0     | Reserved.  |
| 13-8  | AFL      | 0     | Buffer almost full. Holds the programmable almost full level value used to determine almost full buffer condition. If you want an interrupt or a DMA read request to be issued during a receive operation when the data buffer holds at least n bytes, then the buffer MCSPI_MODULCTRL[AFL] must be set with n - 1.          |
|       |          | 1     | 1 byte   |
|       |          | ...   | ...  |
|       |          | 3Fh   | 64 bytes   |
| 7-6   | Reserved | 0     | Reserved.  |
| 5-0   | AEL      | 0     | Buffer almost empty. Holds the programmable almost empty level value used to determine almost empty buffer condition. If you want an interrupt or a DMA write request to be issued during a transmit operation when the data buffer is able to receive n bytes, then the buffer MCSPI_MODULCTRL[AEL] must be set with n - 1. |
|       |          | 1     | 1 byte   |
|       |          | ...   | ...  |
|       |          | 3Fh   | 64 bytes   |

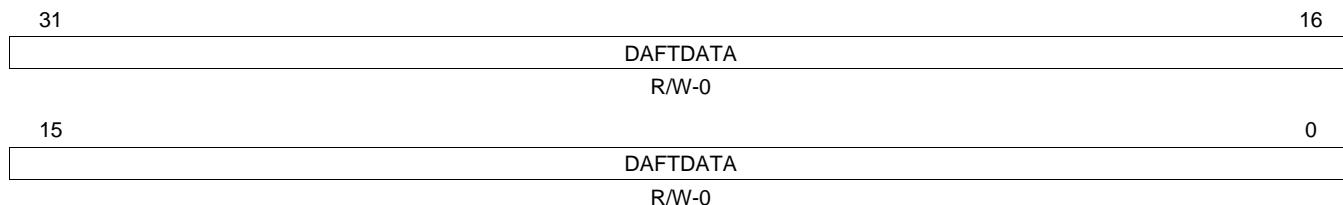
#### 24.4.1.14 McSPI DMA Address Aligned FIFO Transmitter Register (MCSPI\_DAFTX)

The McSPI DMA address aligned FIFO transmitter register (MCSPI\_DAFTX) contains the SPI words to transmit on the serial link when FIFO is used and the DMA address is aligned on 256 bit. This register is an image of one of the MCSPI\_TX(i) registers corresponding to the channel which have its FIFO enabled. The MCSPI\_DAFTX register is shown in [Figure 24-39](#) and described in [Table 24-25](#).

**NOTE:** See Chapter Access to data registers for the list of supported accesses.

The SPI words are transferred with MSB first.

**Figure 24-39. McSPI DMA Address Aligned FIFO Transmitter Register (MCSPI\_DAFTX)**



LEGEND: R/W = Read/Write; -n = value after reset

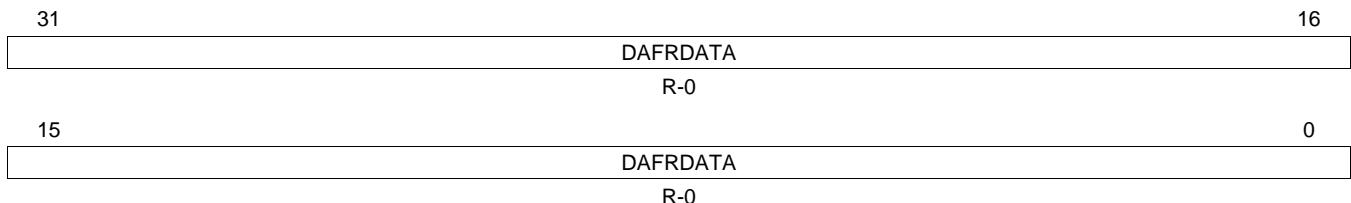
**Table 24-25. McSPI DMA Address Aligned FIFO Transmitter Register (MCSPI\_DAFTX) Field Descriptions**

| Bit  | Field    | Value | Description   |
|------|----------|-------|---|
| 31-0 | DAFTDATA |       | FIFO Data to transmit with DMA 256 bit aligned address.<br>This register is used only when MCSPI_MODULCTRL[FDAA] is set to '1', and only one of the MCSPI_CH(i)CONF[FFEW] of enabled channels is set. Without these conditions, any access to this register will return a null value. |

#### 24.4.1.15 McSPI DMA Address Aligned FIFO Receiver Register (MCSPI\_DAFRX)

The McSPI DMA address aligned FIFO receiver register (MCSPI\_DAFRX) contains the SPI words to received on the serial link when FIFO used and DMA address is aligned on 256 bit. This register is an image of one of MCSPI\_RX(i) register corresponding to the channel which have its FIFO enabled. The MCSPI\_DAFRX register is shown in [Figure 24-40](#) and described in [Table 24-26](#).

**Figure 24-40. McSPI DMA Address Aligned FIFO Receiver Register (MCSPI\_DAFRX)**



LEGEND: R/W = Read/Write; -n = value after reset

**Table 24-26. McSPI DMA Address Aligned FIFO Receiver Register (MCSPI\_DAFRX) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-0 | DAFRDATA |       | FIFO Received Data with DMA 256 bit aligned address.<br>This register is used only when MCSPI_MODULCTRL[FDAA] is set to '1', and only one of the MCSPI_CH(i)CONF[FFER] of enabled channels is set. Without these conditions, any access to this register will return a null value. |

## **General-Purpose Input/Output**

This chapter describes the GPIO of the device.

| Topic                             | Page |
|-----------------------------------|------|
| 25.1 Introduction .....           | 4010 |
| 25.2 Integration .....            | 4011 |
| 25.3 Functional Description ..... | 4014 |
| 25.4 GPIO Registers .....         | 4020 |

## 25.1 Introduction

### 25.1.1 Purpose of the Peripheral

The general-purpose interface combines four general-purpose input/output (GPIO) modules. Each GPIO module provides 32 dedicated general-purpose pins with input and output capabilities; thus, the general-purpose interface supports up to 128 ( $4 \times 32$ ) pins. These pins can be configured for the following applications:

- Data input (capture)/output (drive)
- Keyboard interface with a debounce cell
- Interrupt generation in active mode upon the detection of external events. Detected events are processed by two parallel independent interrupt-generation submodules to support biprocessor operations.

### 25.1.2 GPIO Features

Each GPIO module is made up of 32 identical channels. Each channel can be configured to be used in the following applications:

- Data input/output
- Keyboard interface with a de-bouncing cell
- Synchronous interrupt generation (in active mode) upon the detection of external events (signal transition(s) and/or signal level(s))
- Wake-up request generation (in Idle mode) upon the detection of signal transition(s)

Global features of the GPIO interface are:

- Synchronous interrupt requests from each channel are processed by two identical interrupt generation sub-modules to be used independently by the ARM Subsystem
- Wake-up requests from input channels are merged together to issue one wake-up signal to the system
- Shared registers can be accessed through "Set & Clear" protocol

### 25.1.3 Unsupported GPIO Features

The wake-up feature of the GPIO modules is only supported on GPIO0.

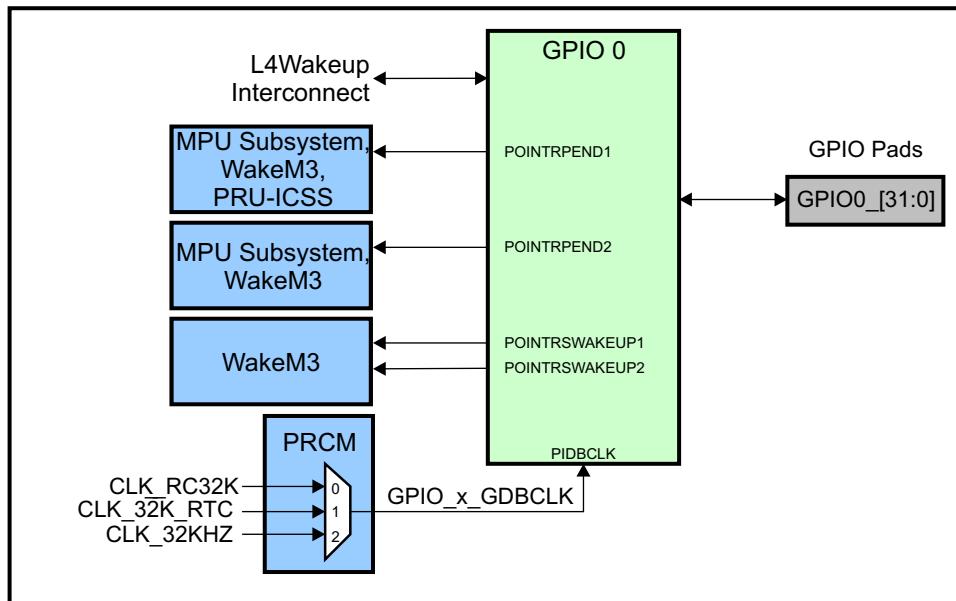
## 25.2 Integration

The device instantiates four GPIO\_V2 modules. Each GPIO module provides the support for 32 dedicated pins with input and output configuration capabilities. Input signals can be used to generate interruptions and wake-up signal. Two Interrupt lines are available for bi-processor operation. Pins can be dedicated to be used as a keyboard controller.

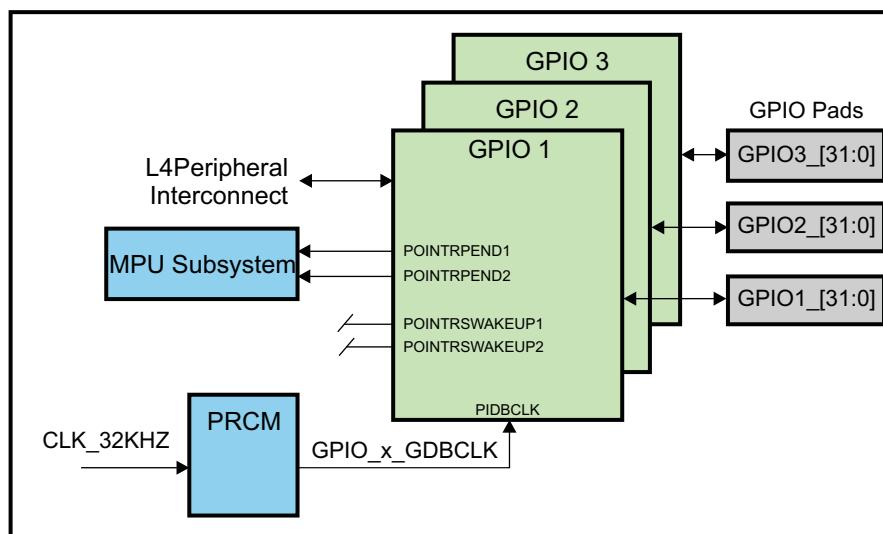
With four GPIO modules, the device allows for a maximum of 128 GPIO pins. (The exact number available varies as a function of the device configuration and pin muxing.) GPIO0 is in the Wakeup domain and may be used to wakeup the device via external sources. GPIO[1:3] are located in the peripheral domain.

[Figure 25-1](#) and [Figure 25-2](#) show the GPIO integration.

**Figure 25-1. GPIO0 Module Integration**



**Figure 25-2. GPIO[1–3] Module Integration**



### 25.2.1 GPIO Connectivity Attributes

The general connectivity attributes for the GPIO modules in the device are shown in [Table 25-1](#) and [Table 25-2](#).

**Table 25-1. GPIO0 Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Wakeup Domain  |
| Clock Domain        | PD_WKUP_L4_WKUP_GCLK (OCP)<br>GPIO_0_GDBCLK (Debounce)   |
| Reset Signals       | WKUP_DOM_RST_N   |
| Idle/Wakeup Signals | Smart Idle / Slave Wakeup  |
| Interrupt Requests  | Two Interrupts:<br>INTRPEND1 (GPIOINT0A) to MPU subsystem, PRU-ICSS (POINTRPEND1), and WakeM3<br>INTRPEND2 (GPIOINT0B) to MPU subsystem and WakeM3 |
| DMA Requests        | Interrupt Requests are redirected as DMA requests: 1 DMA request (GPIOEVT0)  |
| Physical Address    | L4 Wakeup slave port   |

**Table 25-2. GPIO[1:3] Connectivity Attributes**

| Attributes          | Type   |
|---------------------|--|
| Power Domain        | Peripheral Domain  |
| Clock Domain        | PD_PER_L4LS_GCLK (OCP)<br>GPIO_1_GDBCLK (GPIO1 Debounce)<br>GPIO_2_GDBCLK (GPIO2 Debounce)<br>GPIO_3_GDBCLK (GPIO3 Debounce) |
| Reset Signals       | PER_DOM_RST_N  |
| Idle/Wakeup Signals | Smart Idle   |
| Interrupt Requests  | Two Interrupts:<br>INTRPEND1 (GPIOINTxA) to MPU subsystem<br>INTRPEND2 (GPIOINTxB) to MPU subsystem                          |
| DMA Requests        | Interrupt Requests are redirected as DMA requests: 1 DMA request only for GPIO1 (GPIOEVT1) and GPIO2 (GPIOEVT2)              |
| Physical Address    | L4 Peripheral slave port   |

### 25.2.2 GPIO Clock and Reset Management

The GPIO modules require two clocks: The de-bounce clock is used for the de-bouncing cells. The interface clock provided by the peripheral bus (L4 interface) is also the functional clock and is used through the entire GPIO module (except within the de-bouncing sub-module). It clocks the OCP interface and the internal logic. For GPIO0 the debounce clock is selected from one of three sources using the CLKSEL\_GPIO0\_DBCLK register in the PRCM:

- The on-chip ~32.768 KHz oscillator (CLK\_RC32K)
- The PER PLL generated 32.768 KHz clock (CLK\_32KHZ)
- The external 32.768 KHz oscillator/clock (CLK\_32K\_RTC)

**Table 25-3. GPIO Clock Signals**

| Clock Signal                 | Max Freq   | Reference / Source   | Comments                          |
|------------------------------|------------|--|-----------------------------------|
| <b>GPIO0</b>                 |            |  |                                   |
| Functional / Interface clock | 100 MHz    | CORE_CLKOUTM4 / 2  | pd_wkup_l4_wkup_gclk<br>From PRCM |
| Debounce Functional clock    | 32.768 KHz | CLK_RC32K<br>CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375)<br>CLK_32K_RTC | pd_wkup_gpio0_gdbclk<br>From PRCM |
| <b>GPIO[1:3]</b>             |            |  |                                   |

**Table 25-3. GPIO Clock Signals (continued)**

| Clock Signal                      | Max Freq   | Reference / Source                     | Comments                          |
|-----------------------------------|------------|--|-----------------------------------|
| Functional / Interface clock      | 100 MHz    | CORE_CLKOUTM4 / 2                      | pd_per_l4ls_gclk<br>From PRCM     |
| Debounce Functional clock (GPIO1) | 32.768 KHz | CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375) | pd_per_gpio_1_gdbclk<br>From PRCM |
| Debounce Functional clock (GPIO2) | 32.768 KHz | CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375) | pd_per_gpio_2_gdbclk<br>From PRCM |
| Debounce Functional clock (GPIO3) | 32.768 KHz | CLK_32KHZ<br>(PER_CLKOUTM2 / 5859.375) | pd_per_gpio_3_gdbclk<br>From PRCM |

### 25.2.3 GPIO Pin List

Each GPIO module includes 32 interface I/Os. These signals are designated as shown in [Table 25-4](#). Note that for this device, most of these signals will be multiplexed with functional signals from other interfaces.

**Table 25-4. GPIO Pin List**

| Pin          | Type | Description                       |
|--------------|------|-----------------------------------|
| <b>GPIO0</b> |      |                                   |
| GPIO0_[31:0] | I/O  | General Purpose Input-Output pins |
| <b>GPIO1</b> |      |                                   |
| GPIO1_[31:0] | I/O  | General Purpose Input-Output pins |
| <b>GPIO2</b> |      |                                   |
| GPIO2_[31:0] | I/O  | General Purpose Input-Output pins |
| <b>GPIO3</b> |      |                                   |
| GPIO3_[31:0] | I/O  | General Purpose Input-Output pins |

## 25.3 Functional Description

This section discusses the operational details and basic functions of the GPIO peripheral.

### 25.3.1 Operating Modes

Four operating modes are defined for the module:

- **Active mode:** the module is running synchronously on the interface clock, interrupt can be generated according to the configuration and the external signals.
- **Idle mode:** the module is in a waiting state, interface clock can be stopped , no interrupt can be generated. Check the chip top-level functional specification for the availability of the debouncing clock while in Idle mode.
- **Inactive mode:** the module has no activity, interface clock can be stopped, no interrupt can be generated.
- **Disabled mode:** the module is not used, internal clock paths are gated, no interrupt request can be generated.

The Idle and Inactive modes are configured within the module and activated on request by the host processor through system interface sideband signals. The Disabled mode is set by software through a dedicated configuration bit. It unconditionally gates the internal clock paths not use for the system interface. All module registers are 8, 16 or 32-bit accessible through the OCP compatible interface (little endian encoding). In active mode, the event detection (level or transition) is performed in the GPIO module using the interface clock. The detection's precision is set by the frequency of this clock and the selected internal gating scheme.

### 25.3.2 Clocking and Reset Strategy

#### 25.3.2.1 Clocks

GPIO module runs using two clocks:

- The debouncing clock is used for the debouncing sub-module logic (without the corresponding configuration registers). This module can sample the input line and filters the input level using a programmed delay.
- The interface clock provided by the peripheral bus (OCP compatible system interface). It is used through the entire GPIO module (except within the debouncing sub-module logic). It clocks the OCP interface and the internal logic. Clock gating features allow adapting the module power consumption to the activity.

#### 25.3.2.2 Clocks, Gating and Active Edge Definitions

The interface clock provided by the peripheral bus (OCP compatible system interface) is used through the entire GPIO module. Two clock domains are defined: the OCP interface and the internal logic. Each clock domain can be controlled independently. Sampling operations for the data capture and for the events detection are done using the rising edge. The data loaded in the data output register (GPIO\_DATAOUT) is set at the output GPIO pins synchronously with the rising edge of the interface clock.

Five clock gating features are available:

- Clock for the system interface logic can be gated when the module is not accessed, if the AUTOIDLE configuration bit in the system configuration register (GPIO\_SYSConfig) is set. Otherwise, this logic is free running on the interface clock.
- Clock for the input data sample logic can be gated when the data in register is not accessed.
- Four clock groups are used for the logic in the synchronous events detection. Each 8 input GPIO\_V2 pins group will have a separate enable signal depending on the edge/level detection register setting. If a group requires no detection, then the corresponding clock will be gated. All channels are also gated using a 'one out of N' scheme. N can take the values 1, 2, 4 or 8. The interface clock is enabled for this logic one cycle every N cycles. When N is equal to 1, there is no gating and this logic is free running on the interface clock. When N is between 2 to 8, this logic is running at the equivalent frequency of interface clock frequency divided by N.

- In Inactive mode, all internal clock paths are gated.
- In Disabled mode, all internal clock paths not used for the system interface are gated. All GPIO registers are accessible synchronously with the interface clock.

### 25.3.2.3 Sleep Mode Request and Acknowledge

Upon a Sleep mode request issued by the host processor, the GPIO module goes to the Idle mode according to the IDLEMODE field in the system configuration register (GPIO\_SYSCONFIG).

- IDLEMODE = 0 (Force-Idle mode): the GPIO goes in Inactive mode independently of the internal module state and the Idle acknowledge is unconditionally sent. In Force-Idle mode, the module is in Inactive mode.
- IDLEMODE = 1h (No-Idle mode): the GPIO does not go to the Idle mode and the Idle acknowledge is never sent.
- IDLEMODE = 2h (Smart-Idle mode) or IDLEMODE = 3h (Smart-Idle mode): the GPIO module evaluates its internal capability to have the interface clock switched off. Once there is no more internal activity (the data input register completed to capture the input GPIO pins, there is no pending interrupt, all interrupt status bits are cleared, and there is no write access to GPIO\_DEBOUNCINGTIME register pending to be synchronized), the Idle acknowledge is asserted and the GPIO enters Idle mode. When the system is awake, the Idle Request goes inactive, the Idle acknowledge signals are immediately deasserted.

---

**NOTE:** Idle mode request and Idle acknowledge are system interface sideband signals. Once the GPIO acknowledges the Sleep mode request (Idle acknowledge has been sent), the interface clock can be stopped anytime.

Upon a Sleep mode request issued by the host processor, the GPIO module goes to the Idle mode only if there is no active bit in GPIO\_IRQSTATUS\_RAW\_n registers.

---

### 25.3.2.4 Reset

The OCP hardware Reset signal has a global reset action on the GPIO. All configuration registers, all DFFs clocked with the Interface clock or Debouncing clock and all internal state machines are reset when the OCP hardware Reset is active (low level). The RESETDONE bit in the system status register (GPIO\_SYSSTATUS) monitors the internal reset status: it is set when the Reset is completed on both OCP and Debouncing clock domains. The software Reset (SOFTRESET bit in the system configuration register) has the same effect as the OCP hardware Reset signal, and the RESETDONE bit in GPIO\_SYSSTATUS is updated in the same condition.

## 25.3.3 Interrupt Features

### 25.3.3.1 Functional Description

In order to generate an interrupt request to a host processor upon a defined event (level or logic transition) occurring on a GPIO pin, the GPIO configuration registers have to be programmed as follows:

- Interrupts for the GPIO channel must be enabled in the GPIO\_IRQSTATUS\_SET\_0 and/or GPIO\_IRQSTATUS\_SET\_1 registers.
- The expected event(s) on input GPIO to trigger the interrupt request has to be selected in the GPIO\_LEVELDETECT0, GPIO\_LEVELDETECT1, GPIO\_RISINGDETECT, and GPIO\_FALLINGDETECT registers.

For instance, interrupt generation on both edges on input k is configured by setting to 1 the  $k$ th bit in registers GPIO\_RISINGDETECT and GPIO\_FALLINGDETECT along with the interrupt enabling for one or both interrupt lines (GPIO\_IRQSTATUS\_SET\_n).

---

**NOTE:** All interrupt sources (the 32 input GPIO channels) are merged together to issue two synchronous interrupt requests 1 and 2.

---

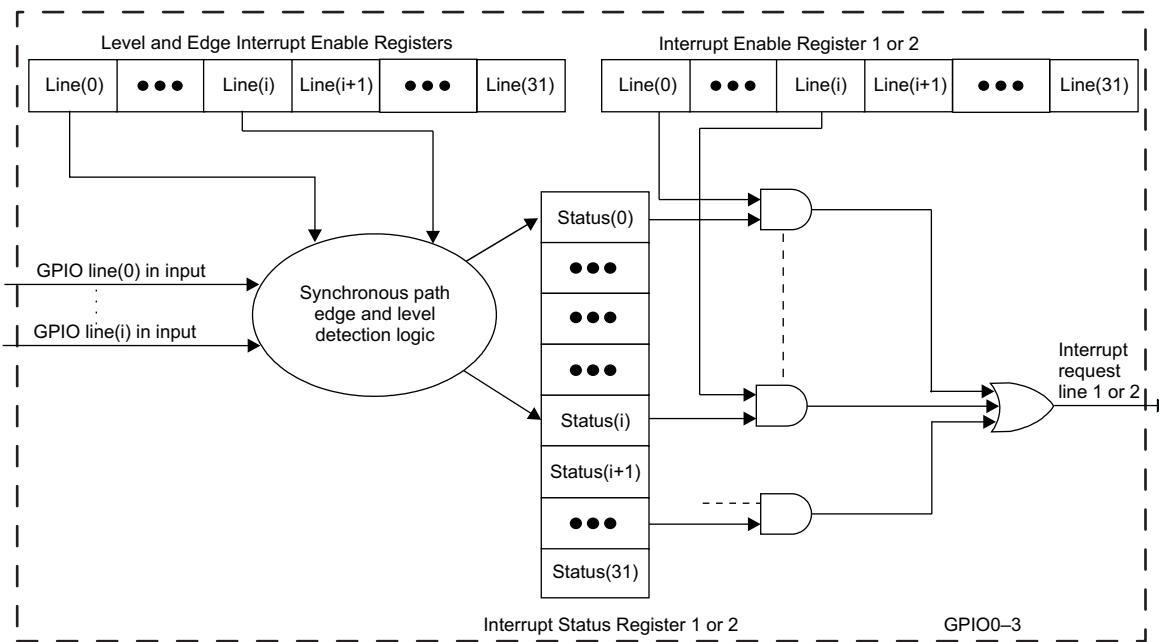
### 25.3.3.2 Synchronous Path: Interrupt Request Generation

In Active mode, once the GPIO configuration registers have been set to enable the interrupt generation, a synchronous path (Figure 25-3) samples the transitions and levels on the input GPIO with the internally gated interface clock. When an event matches the programmed settings, the corresponding bit in the GPIO\_IRQSTATUS\_RAW\_n registers is set to 1 and, on the following interface clock cycle, the interrupt lines 1 and/or 2 are activated (depending on the GPIO\_IRQSTATUS\_SET\_n registers).

Due to the sampling operation, the minimum pulse width on the input GPIO to trigger a synchronous interrupt request is two times the internally gated interface clock period (the internally gated interface clock period is equal to N times the interface clock period). This minimum pulse width has to be met before and after any expected level transition detection. Level detection requires the selected level to be stable for at least two times the internally gated interface clock period to trigger a synchronous interrupt.

As the module is synchronous, latency is minimal between the expected event occurrence and the activation of the interrupt line(s). This should not exceed 3 internally gated interface clock cycles + 2 interface clock cycles when the debouncing feature is not used. When the debouncing feature is active, the latency depends on the GPIO\_DEBOUNCINGTIME register value and should be less than 3 internally gated interface clock cycles + 2 interface clock cycles + GPIO\_DEBOUNCINGTIME value debouncing clock cycles + 3 debouncing clock cycles.

**Figure 25-3. Interrupt Request Generation**



### 25.3.3.3 Interrupt Line Release

When the host processor receives an interrupt request issued by the GPIO module, it can read the corresponding GPIO\_IRQSTATUS\_n register to find out which input GPIO has triggered the interrupt. After servicing the interrupt, the processor resets the status bit and releases the interrupt line by writing a 1 in the corresponding bit of the GPIO\_IRQSTATUS\_n register. If there is still a pending interrupt request to serve (all bits in the GPIO\_IRQSTATUS\_RAW\_n register not masked by the GPIO\_IRQSTATUS\_SET\_n, which are not cleared by setting the GPIO\_IRQSTATUS\_CLR\_n), the interrupt line will be re-asserted.

## 25.3.4 General-Purpose Interface Basic Programming Model

### 25.3.4.1 Power Saving by Grouping the Edge/Level Detection

Each GPIO module implements four gated clocks used by the edge/level detection logic to save power. Each group of eight input GPIO pins generates a separate enable signal depending on the edge/level detection register setting (because the input is 32 bits, four groups of eight inputs are defined for each GPIO module). If a group requires no edge/level detection, then the corresponding clock is gated (cut off). Grouping the edge/level enable can save the power consumption of the module as described in the following example.

If any of the registers:

- GPIO\_LEVELDETECT0
- GPIO\_LEVELDETECT1
- GPIO\_RISINGDETECT
- GPIO\_FALLINGDETECT

are set to 0101 0101h, then all clocks are active (power consumption is high);

are set to 0000 00FFh, then a single clock is active.

---

**NOTE:** When the clocks are enabled by writing to the GPIO\_LEVELDETECT0, GPIO\_LEVELDETECT1, GPIO\_RISINGDETECT, and GPIO\_FALLINGDETECT registers, the detection starts after 5 clock cycles. This period is required to clean the synchronization edge/level detection pipeline.

The mechanism is independent of each clock group. If the clock has been started before a new setting is performed, the following is recommended: first, set the new detection required; second, disable the previous setting (if necessary). In this way, the corresponding clock is not gated and the detection starts immediately.

---

### 25.3.4.2 Set and Clear Instructions

The GPIO module implements the set-and-clear protocol register update for the data output and interrupt enable registers. This protocol is an alternative to the atomic test and set operations and consists of writing operations at dedicated addresses (one address for setting bit[s] and one address for clearing bit[s]). The data to write is 1 at bit position(s) to clear (or to set) and 0 at unaffected bit(s).

Registers can be accessed in two ways:

- Standard: Full register read and write operations at the primary register address
- Set and clear (recommended): Separate addresses are provided to set (and clear) bits in registers. Writing 1 at these addresses sets (or clears) the corresponding bit into the equivalent register; writing a 0 has no effect.

Therefore, for these registers, three addresses are defined for one unique physical register. Reading these addresses has the same effect and returns the register value.

### 25.3.4.2.1 Clear Instruction

#### 25.3.4.2.1.1 Clear Interrupt Enable Registers (*GPIO\_IRQSTATUS\_CLR\_0* and *GPIO\_IRQSTATUS\_CLR\_1*):

- A write operation in the clear interrupt enable1 (or enable2) register clears the corresponding bit in the interrupt enable1 (or enable2) register when the written bit is 1; a written bit at 0 has no effect.
- A read of the clear interrupt enable1 (or enable2) register returns the value of the interrupt enable1 (or enable2) register.

#### 25.3.4.2.1.2 Clear Data Output Register (*GPIO\_CLEARDATAOUT*):

- A write operation in the clear data output register clears the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.
- A read of the clear data output register returns the value of the data output register.

#### 25.3.4.2.1.3 Clear Instruction Example

Assume the data output register (or one of the interrupt enable registers) contains the binary value, 0000 0001 0000 0001h, and you want to clear bit 0.

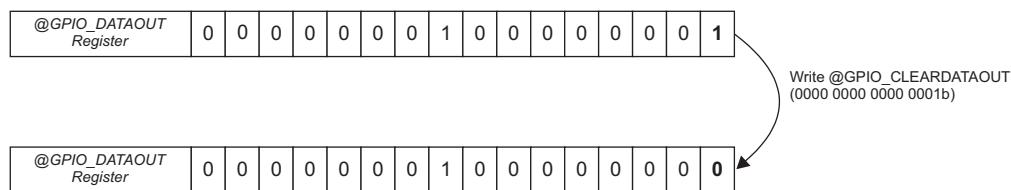
With the clear instruction feature, write 0000 0000 0000 0001h at the address of the clear data output register (or at the address of the clear interrupt enable register). After this write operation, a reading of the data output register (or the interrupt enable register) returns 0000 0001 0000 0000h; bit 0 is cleared.

---

**NOTE:** Although the general-purpose interface registers are 32-bits wide, only the 16 least-significant bits are represented in this example.

---

**Figure 25-4. Write @ *GPIO\_CLEARDATAOUT* Register Example**



### 25.3.4.2.2 Set Instruction

#### 25.3.4.2.2.1 Set Interrupt Enable Registers (*GPIO\_IRQSTATUS\_SET\_0* and *GPIO\_IRQSTATUS\_SET\_1*):

- A write operation in the set interrupt enable1 (or enable2) register sets the corresponding bit in the interrupt enable1 (or enable2) register when the written bit is 1; a written bit at 0 has no effect.
- A read of the set interrupt enable1 (or enable2) register returns the value of the interrupt enable1 (or enable2) register.

#### 25.3.4.2.2.2 Set Data Output Register (*GPIO\_SETDATAOUT*):

- A write operation in the set data output register sets the corresponding bit in the data output register when the written bit is 1; a written bit at 0 has no effect.
- A read of the set data output register returns the value of the data output register.

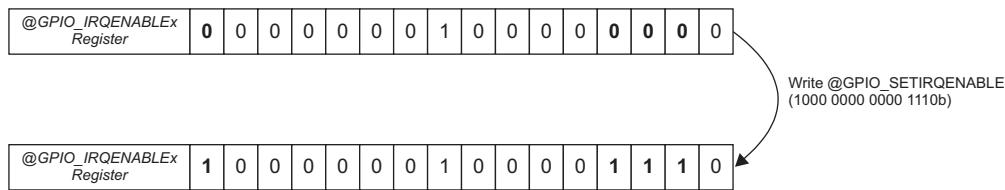
### 25.3.4.2.2.3 Set Instruction Example

Assume the interrupt enable1 (or enable2) register (or the data output register) contains the binary value, 0000 0001 0000 0000h, and you want to set bits 15, 3, 2, and 1.

With the set instruction feature, write 1000 0000 0000 1110h at the address of the set interrupt enable1 (or enable2) register (or at the address of the set data output register). After this write operation, a reading of the interrupt enable1 (or enable2) register (or the data output register) returns 1000 0001 0000 1110h; bits 15, 3, 2, and 1 are set.

**NOTE:** Although the general-purpose interface registers are 32-bits wide, only the 16 least-significant bits are represented in this example.

**Figure 25-5. Write @ GPIO\_SETIRQENABLEx Register Example**



### 25.3.4.3 Data Input (Capture)/Output (Drive)

The output enable register (GPIO\_OE) controls the output/input capability for each pin. At reset, all the GPIO-related pins are configured as input and output capabilities are disabled. This register is not used within the module; its only function is to carry the pads configuration.

When configured as an output (the desired bit reset in GPIO\_OE), the value of the corresponding bit in the GPIO\_DATAOUT register is driven on the corresponding GPIO pin. Data is written to the data output register synchronously with the interface clock. This register can be accessed with read/write operations or by using the alternate set and clear protocol register update feature. This feature lets you set or clear specific bits of this register with a single write access to the set data output register (GPIO\_SETDATAOUT) or to the clear data output register (GPIO\_CLEARDATAOUT) address. If the application uses a pin as an output and does not want interrupt generation from this pin, the application must properly configure the interrupt enable registers.

When configured as an input (the desired bit set to 1 in GPIO\_OE), the state of the input can be read from the corresponding bit in the GPIO\_DATAIN register. The input data is sampled synchronously with the interface clock and then captured in the data input register synchronously with the interface clock. When the GPIO pin levels change, they are captured into this register after two interface clock cycles (the required cycles to synchronize and to write data). If the application uses a pin as an input, the application must properly configure the interrupt enable registers to the interrupt as needed.

### 25.3.4.4 Debouncing Time

To enable the debounce feature for a pin, the GPIO configuration registers must be programmed as follows:

- The GPIO pin must be configured as input in the output enable register (write 1 to the corresponding bit of the GPIO\_OE register).
- The debouncing time must be set in the debouncing value register (GPIO\_DEBOUNCINGTIME). The GPIO\_DEBOUNCINGTIME register is used to set the debouncing time for all input lines in the GPIO module. The value is global for all the ports of one GPIO module, so up to six different debouncing values are possible. The debounce cell is running with the debounce clock (32 kHz). This register represents the number of the clock cycle(s) (one cycle is 31 microseconds long) to be used.

The following formula describes the required input stable time to be propagated to the debounced output:

$$\text{Debouncing time} = (\text{DEBOUNCETIME} + 1) \times 31 \mu\text{s}$$

Where the DEBOUNCETIME field value in the GPIO\_DEBOUNCINGTIME register is from 0 to 255.

- The debouncing feature must be enabled in the debouncing enable register (write 1 to the corresponding DEBOUNCEENABLE bit in the GPIO\_DEBOUNCENABLE register).

#### 25.3.4.5 GPIO as a Keyboard Interface

The general-purpose interface can be used as a keyboard interface (Figure 25-6). You can dedicate channels based on the keyboard matrix = \* c). Figure 25-6 shows row channels configured as inputs with the input debounce feature enabled. The row channels are driven high with an external pull-up. Column channels are configured as outputs and drive a low level.

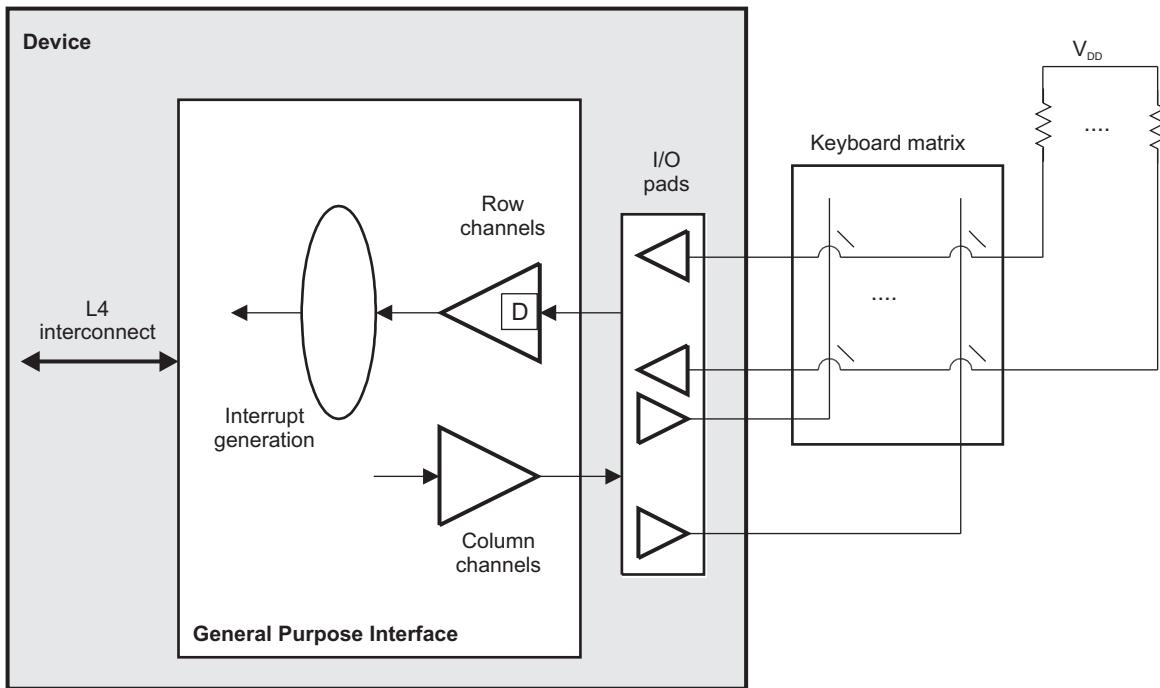
When a keyboard matrix key is pressed, the corresponding row and column lines are shorted together and a low level is driven on the corresponding row channel. This generates an interrupt based on the proper configuration (see Section 25.3.3).

When the keyboard interrupt is received, the processor can disable the keyboard interrupt and scan the column channels for the key coordinates.

- The scanning sequence has as many states as column channels: For each step in the sequence, the processor drives one column channel low and the others high.
- The processor reads the values of the row channels and thus detects which keys in the column are pressed.

At the end of the scanning sequence, the processor establishes which keys are pressed. The keyboard interface can then be reconfigured in the interrupt waiting state.

**Figure 25-6. General-Purpose Interface Used as a Keyboard Interface**



### 25.4 GPIO Registers

#### 25.4.1 GPIO Registers

Table 25-5 lists the memory-mapped registers for the GPIO. All register offset addresses not listed in Table 25-5 should be considered as reserved locations and the register contents should not be modified.

**Table 25-5. GPIO REGISTERS**

| <b>Offset</b> | <b>Acronym</b>       | <b>Register Name</b> | <b>Section</b>                    |
|---------------|----------------------|----------------------|-----------------------------------|
| 0h            | GPIO_REVISION        |                      | <a href="#">Section 25.4.1.1</a>  |
| 10h           | GPIO_SYSCONFIG       |                      | <a href="#">Section 25.4.1.2</a>  |
| 24h           | GPIO_IRQSTATUS_RAW_0 |                      | <a href="#">Section 25.4.1.3</a>  |
| 28h           | GPIO_IRQSTATUS_RAW_1 |                      | <a href="#">Section 25.4.1.4</a>  |
| 2Ch           | GPIO_IRQSTATUS_0     |                      | <a href="#">Section 25.4.1.5</a>  |
| 30h           | GPIO_IRQSTATUS_1     |                      | <a href="#">Section 25.4.1.6</a>  |
| 34h           | GPIO_IRQSTATUS_SET_0 |                      | <a href="#">Section 25.4.1.7</a>  |
| 38h           | GPIO_IRQSTATUS_SET_1 |                      | <a href="#">Section 25.4.1.8</a>  |
| 3Ch           | GPIO_IRQSTATUS_CLR_0 |                      | <a href="#">Section 25.4.1.9</a>  |
| 40h           | GPIO_IRQSTATUS_CLR_1 |                      | <a href="#">Section 25.4.1.10</a> |
| 44h           | GPIO_IRQWAKEN_0      |                      | <a href="#">Section 25.4.1.11</a> |
| 48h           | GPIO_IRQWAKEN_1      |                      | <a href="#">Section 25.4.1.12</a> |
| 114h          | GPIO_SYSSTATUS       |                      | <a href="#">Section 25.4.1.13</a> |
| 130h          | GPIO_CTRL            |                      | <a href="#">Section 25.4.1.14</a> |
| 134h          | GPIO_OE              |                      | <a href="#">Section 25.4.1.15</a> |
| 138h          | GPIO_DATAIN          |                      | <a href="#">Section 25.4.1.16</a> |
| 13Ch          | GPIO_DATAOUT         |                      | <a href="#">Section 25.4.1.17</a> |
| 140h          | GPIO_LEVELDETECT0    |                      | <a href="#">Section 25.4.1.18</a> |
| 144h          | GPIO_LEVELDETECT1    |                      | <a href="#">Section 25.4.1.19</a> |
| 148h          | GPIO_RISINGDETECT    |                      | <a href="#">Section 25.4.1.20</a> |
| 14Ch          | GPIO_FALLINGDETECT   |                      | <a href="#">Section 25.4.1.21</a> |
| 150h          | GPIO_DEBOUNCENABLE   |                      | <a href="#">Section 25.4.1.22</a> |
| 154h          | GPIO_DEBOUNCINGTIME  |                      | <a href="#">Section 25.4.1.23</a> |
| 190h          | GPIO_CLEARDATAOUT    |                      | <a href="#">Section 25.4.1.24</a> |
| 194h          | GPIO_SETDATAOUT      |                      | <a href="#">Section 25.4.1.25</a> |

### 25.4.1.1 GPIO\_REVISION Register (offset = 0h) [reset = 50600801h]

GPIO\_REVISION is shown in [Figure 25-7](#) and described in [Table 25-6](#).

The GPIO revision register is a read only register containing the revision number of the GPIO module. A write to this register has no effect, the same as the reset.

**Figure 25-7. GPIO\_REVISION Register**

| 31 | 30     | 29 | 28       | 27    | 26    | 25    | 24 |
|----|--------|----|----------|-------|-------|-------|----|
|    | SCHEME |    | Reserved |       |       | FUNC  |    |
|    | R-1h   |    | R-1h     |       |       | R-60h |    |
| 23 | 22     | 21 | 20       | 19    | 18    | 17    | 16 |
|    |        |    |          | FUNC  |       |       |    |
|    |        |    |          | R-60h |       |       |    |
| 15 | 14     | 13 | 12       | 11    | 10    | 9     | 8  |
|    | RTL    |    |          |       |       | MAJOR |    |
|    | R-1h   |    |          |       |       | R-0h  |    |
| 7  | 6      | 5  | 4        | 3     | 2     | 1     | 0  |
|    | CUSTOM |    |          |       | MINOR |       |    |
|    | R-0h   |    |          |       | R-1h  |       |    |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-6. GPIO\_REVISION Register Field Descriptions**

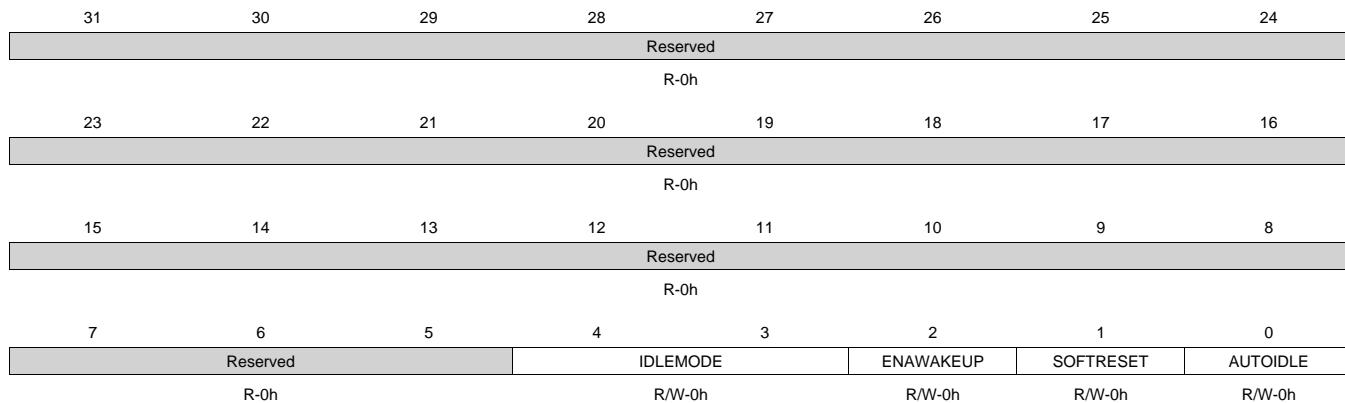
| Bit   | Field    | Type | Reset | Description  |
|-------|----------|------|-------|--|
| 31-30 | SCHEME   | R    | 1h    | Used to distinguish between old Scheme and current.  |
| 29-28 | Reserved | R    | 1h    |  |
| 27-16 | FUNC     | R    | 60h   | Indicates a software compatible module family.       |
| 15-11 | RTL      | R    | 1h    | RTL version  |
| 10-8  | MAJOR    | R    | 0h    | Major Revision                                       |
| 7-6   | CUSTOM   | R    | 0h    | Indicates a special version for a particular device. |
| 5-0   | MINOR    | R    | 1h    | Minor Revision                                       |

### 25.4.1.2 GPIO\_SYSConfig Register (offset = 10h) [reset = 0h]

GPIO\_SYSConfig is shown in [Figure 25-8](#) and described in [Table 25-7](#).

The GPIO\_SYSConfig register controls the various parameters of the L4 interconnect. When the AUTOIDLE bit is set, the GPIO\_DATAIN read command has a 3 OCP cycle latency due to the data in sample gating mechanism. When the AUTOIDLE bit is not set, the GPIO\_DATAIN read command has a 2 OCP cycle latency.

**Figure 25-8. GPIO\_SYSConfig Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-7. GPIO\_SYSConfig Register Field Descriptions**

| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-5 | Reserved  | R    | 0h    |  |
| 4-3  | IDLEMODE  | R/W  | 0h    | Power Management, Req/Ack control.<br>0x0 = Force-idle. An idle request is acknowledged unconditionally<br>0x1 = No-idle. An idle request is never acknowledged<br>0x2 = Smart-idle. Acknowledgment to an idle request is given based on the internal activity of the module<br>0x3 = Smart Idle Wakeup (GPIO0 only) |
| 2    | ENAWAKEUP | R/W  | 0h    | 0x0 = Wakeup generation is disabled.<br>0x1 = Wakeup capability is enabled upon expected transition on input GPIO pin.   |
| 1    | SOFTRESET | R/W  | 0h    | Software reset.<br>This bit is automatically reset by the hardware.<br>During reads, it always returns 0.<br>0x0 = Normal mode<br>0x1 = The module is reset  |
| 0    | AUTOIDLE  | R/W  | 0h    | Internal interface clock gating strategy<br>0x0 = Internal Interface OCP clock is free-running<br>0x1 = Automatic internal OCP clock gating, based on the OCP interface activity   |

### 25.4.1.3 GPIO\_IRQSTATUS\_RAW\_0 Register (offset = 24h) [reset = 0h]

GPIO\_IRQSTATUS\_RAW\_0 is shown in [Figure 25-9](#) and described in [Table 25-8](#).

The GPIO\_IRQSTATUS\_RAW\_0 register provides core status information for the interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit sets it to 1, that is, triggers the IRQ (mostly for debug). Writing a 0 has no effect, that is, the register value is not be modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

**Figure 25-9. GPIO\_IRQSTATUS\_RAW\_0 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-8. GPIO\_IRQSTATUS\_RAW\_0 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n status.<br>0x0 = No effect.<br>0x1 = IRQ is triggered. |

#### 25.4.1.4 GPIO\_IRQSTATUS\_RAW\_1 Register (offset = 28h) [reset = 0h]

GPIO\_IRQSTATUS\_RAW\_1 is shown in [Figure 25-10](#) and described in [Table 25-9](#).

The GPIO\_IRQSTATUS\_RAW\_1 register provides core status information for the interrupt handling, showing all active events (enabled and not enabled). The fields are read-write. Writing a 1 to a bit sets it to 1, that is, triggers the IRQ (mostly for debug). Writing a 0 has no effect, that is, the register value is not be modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

**Figure 25-10. GPIO\_IRQSTATUS\_RAW\_1 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-9. GPIO\_IRQSTATUS\_RAW\_1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n status.<br>0x0 = No effect.<br>0x1 = IRQ is triggered. |

### 25.4.1.5 GPIO\_IRQSTATUS\_0 Register (offset = 2Ch) [reset = 0h]

GPIO\_IRQSTATUS\_0 is shown in [Figure 25-11](#) and described in [Table 25-10](#).

The GPIO\_IRQSTATUS\_0 register provides core status information for the interrupt handling, showing all active events which have been enabled. The fields are read-write. Writing a 1 to a bit clears the bit to 0, that is, clears the IRQ. Writing a 0 has no effect, that is, the register value is not modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

**Figure 25-11. GPIO\_IRQSTATUS\_0 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W1C-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-10. GPIO\_IRQSTATUS\_0 Register Field Descriptions**

| Bit  | Field      | Type  | Reset | Description  |
|------|------------|-------|-------|--|
| 31-0 | INTLINE[n] | R/W1C | 0h    | Interrupt n status.<br>0x0 = No effect.<br>0x1 = IRQ is triggered. |

#### **25.4.1.6 GPIO\_IRQSTATUS\_1 Register (offset = 30h) [reset = 0h]**

GPIO\_IRQSTATUS\_1 is shown in [Figure 25-12](#) and described in [Table 25-11](#).

The GPIO\_IRQSTATUS\_1 register provides core status information for the interrupt handling, showing all active events which have been enabled. The fields are read-write. Writing a 1 to a bit clears the bit to 0, that is, clears the IRQ. Writing a 0 has no effect, that is, the register value is not modified. Only enabled, active events trigger an actual interrupt request on the IRQ output line.

**Figure 25-12. GPIO\_IRQSTATUS\_1 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W1C-0h   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-11. GPIO\_IRQSTATUS\_1 Register Field Descriptions**

| Bit  | Field      | Type  | Reset | Description  |
|------|------------|-------|-------|--|
| 31-0 | INTLINE[n] | R/W1C | 0h    | Interrupt n status.<br>0x0 = No effect.<br>0x1 = IRQ is triggered. |

### 25.4.1.7 GPIO\_IRQSTATUS\_SET\_0 Register (offset = 34h) [reset = 0h]

GPIO\_IRQSTATUS\_SET\_0 is shown in [Figure 25-13](#) and described in [Table 25-12](#).

All 1-bit fields in the GPIO\_IRQSTATUS\_SET\_0 register enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit enables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

**Figure 25-13. GPIO\_IRQSTATUS\_SET\_0 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-12. GPIO\_IRQSTATUS\_SET\_0 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n enable<br>0x0 = No effect.<br>0x1 = Enable IRQ generation. |

#### **25.4.1.8 GPIO\_IRQSTATUS\_SET\_1 Register (offset = 38h) [reset = 0h]**

GPIO\_IRQSTATUS\_SET\_1 is shown in [Figure 25-14](#) and described in [Table 25-13](#).

All 1-bit fields in the GPIO\_IRQSTATUS\_SET\_1 register enable a specific interrupt event to trigger an interrupt request. Writing a 1 to a bit enables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

**Figure 25-14. GPIO\_IRQSTATUS\_SET\_1 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-13. GPIO\_IRQSTATUS\_SET\_1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n enable<br>0x0 = No effect.<br>0x1 = Enable IRQ generation. |

### 25.4.1.9 GPIO\_IRQSTATUS\_CLR\_0 Register (offset = 3Ch) [reset = 0h]

GPIO\_IRQSTATUS\_CLR\_0 is shown in [Figure 25-15](#) and described in [Table 25-14](#).

All 1-bit fields in the GPIO\_IRQSTATUS\_CLR\_0 register clear a specific interrupt event. Writing a 1 to a bit disables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

**Figure 25-15. GPIO\_IRQSTATUS\_CLR\_0 Register**

| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-14. GPIO\_IRQSTATUS\_CLR\_0 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n enable<br>0x0 = No effect.<br>0x1 = Disable IRQ generation. |

#### **25.4.1.10 GPIO\_IRQSTATUS\_CLR\_1 Register (offset = 40h) [reset = 0h]**

GPIO\_IRQSTATUS\_CLR\_1 is shown in [Figure 25-16](#) and described in [Table 25-15](#).

All 1-bit fields in the GPIO\_IRQSTATUS\_CLR\_1 register clear a specific interrupt event. Writing a 1 to a bit disables the interrupt field. Writing a 0 has no effect, that is, the register value is not modified.

**Figure 25-16. GPIO\_IRQSTATUS\_CLR\_1 Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-15. GPIO\_IRQSTATUS\_CLR\_1 Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description   |
|------|------------|------|-------|---|
| 31-0 | INTLINE[n] | R/W  | 0h    | Interrupt n enable<br>0x0 = No effect.<br>0x1 = Disable IRQ generation. |

### 25.4.1.11 GPIO\_IRQWAKEN\_0 Register (offset = 44h) [reset = 0h]

GPIO\_IRQWAKEN\_0 is shown in [Figure 25-17](#) and described in [Table 25-16](#).

Per-event wakeup enable vector (corresponding to first line of interrupt). Every 1-bit field in the GPIO\_IRQWAKEN\_0 register enables a specific (synchronous) IRQ request source to generate an asynchronous wakeup (on the appropriate sakeup line). This register allows the user to mask the expected transition on input GPIO from generating a wakeup request. The GPIO\_IRQWAKEN\_0 is programmed synchronously with the interface clock before any Idle mode request coming from the host processor. Note: In Force-Idle mode, the module wake-up feature is totally inhibited. The wake-up generation can also be gated at module level using the EnaWakeUp bit from GPIO\_SYSConfig register.

**Figure 25-17. GPIO\_IRQWAKEN\_0 Register**

|         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31      | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-16. GPIO\_IRQWAKEN\_0 Register Field Descriptions**

| Bit  | Field   | Type | Reset | Description   |
|------|---------|------|-------|---|
| 31-0 | INTLINE | R/W  | 0h    | Wakeup Enable for Interrupt Line<br>0x0 = Disable wakeup generation.<br>0x1 = Enable wakeup generation. |

#### 25.4.1.12 GPIO\_IRQWAKEN\_1 Register (offset = 48h) [reset = 0h]

GPIO\_IRQWAKEN\_1 is shown in [Figure 25-18](#) and described in [Table 25-17](#).

Per-event wakeup enable vector (corresponding to second line of interrupt). Every 1-bit field in the GPIO\_IRQWAKEN\_1 register enables a specific (synchronous) IRQ request source to generate an asynchronous wakeup (on the appropriate sakeup line). This register allows the user to mask the expected transition on input GPIO from generating a wakeup request. The GPIO\_IRQWAKEN\_1 is programmed synchronously with the interface clock before any Idle mode request coming from the host processor. Note: In Force-Idle mode, the module wake-up feature is totally inhibited. The wake-up generation can also be gated at module level using the EnaWakeUp bit from GPIO\_SYSConfig register.

**Figure 25-18. GPIO\_IRQWAKEN\_1 Register**

| 31      | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| INTLINE |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-17. GPIO\_IRQWAKEN\_1 Register Field Descriptions**

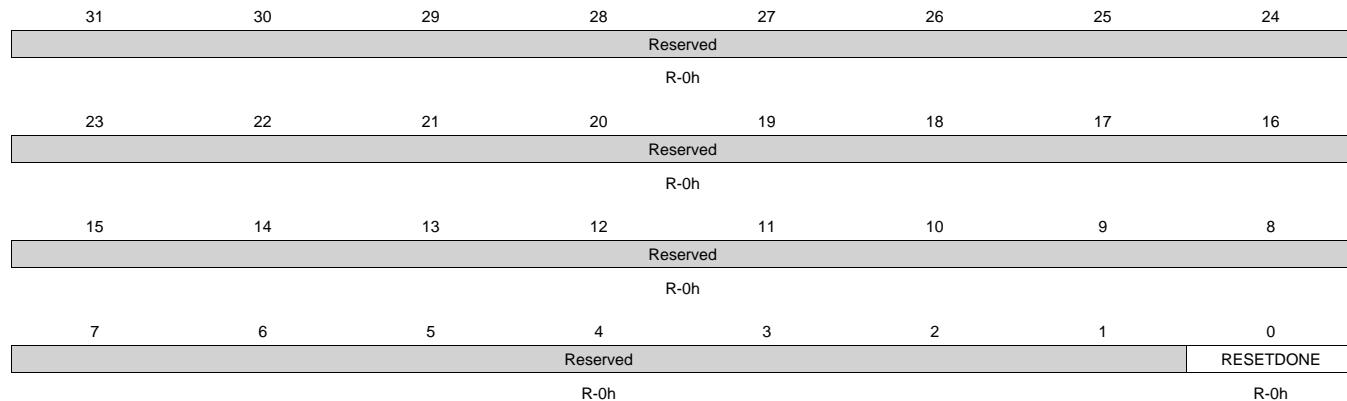
| Bit  | Field   | Type | Reset | Description   |
|------|---------|------|-------|---|
| 31-0 | INTLINE | R/W  | 0h    | Wakeup Enable for Interrupt Line<br>0x0 = Disable wakeup generation.<br>0x1 = Enable wakeup generation. |

### 25.4.1.13 GPIO\_SYSSTATUS Register (offset = 114h) [reset = 0h]

GPIO\_SYSSTATUS is shown in [Figure 25-19](#) and described in [Table 25-18](#).

The GPIO\_SYSSTATUS register provides the reset status information about the GPIO module. It is a read-only register; a write to this register has no effect.

**Figure 25-19. GPIO\_SYSSTATUS Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-18. GPIO\_SYSSTATUS Register Field Descriptions**

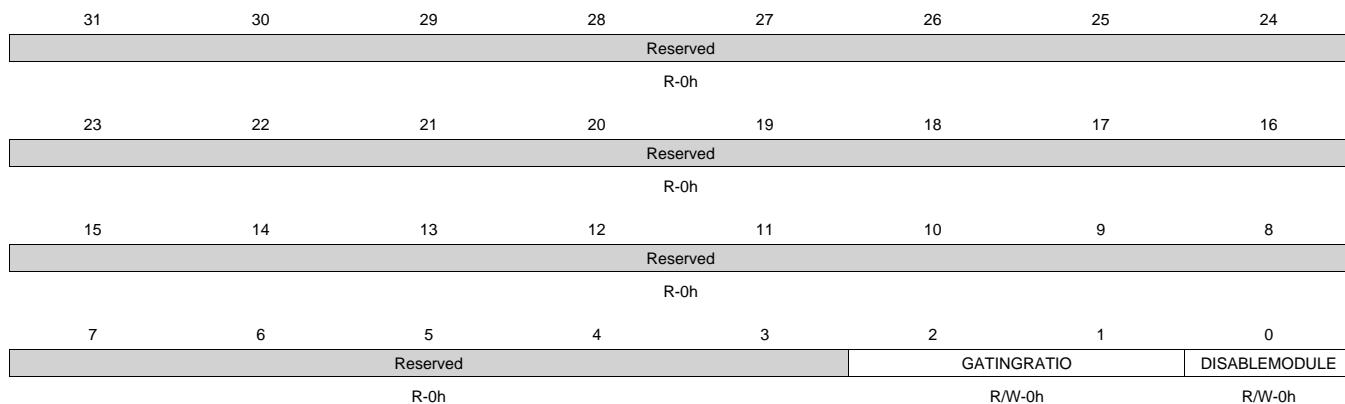
| Bit  | Field     | Type | Reset | Description  |
|------|-----------|------|-------|--|
| 31-1 | Reserved  | R    | 0h    |  |
| 0    | RESETDONE | R    | 0h    | Reset status information.<br>0x0 = Internal Reset is on-going.<br>0x1 = Reset completed. |

#### 25.4.1.14 GPIO\_CTRL Register (offset = 130h) [reset = 0h]

GPIO\_CTRL is shown in [Figure 25-20](#) and described in [Table 25-19](#).

The GPIO\_CTRL register controls the clock gating functionality. The DISABLEMODULE bit controls a clock gating feature at the module level. When set, this bit forces the clock gating for all internal clock paths. Module internal activity is suspended. System interface is not affected by this bit. System interface clock gating is controlled with the AUTOIDLE bit in the system configuration register (GPIO\_SYSCONFIG). This bit is to be used for power saving when the module is not used because of the multiplexing configuration selected at the chip level. This bit has precedence over all other internal configuration bits.

**Figure 25-20. GPIO\_CTRL Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-19. GPIO\_CTRL Register Field Descriptions**

| Bit  | Field         | Type | Reset | Description  |
|------|---------------|------|-------|--|
| 31-3 | Reserved      | R    | 0h    |  |
| 2-1  | GATINGRATIO   | R/W  | 0h    | Gating Ratio.<br>Controls the clock gating for the event detection logic.<br>0x0 = Functional clock is interface clock.<br>0x1 = Functional clock is interface clock divided by 2.<br>0x2 = Functional clock is interface clock divided by 4.<br>0x3 = Functional clock is interface clock divided by 8. |
| 0    | DISABLEMODULE | R/W  | 0h    | Module Disable<br>0x0 = Module is enabled, clocks are not gated.<br>0x1 = Module is disabled, clocks are gated.  |

### 25.4.1.15 GPIO\_OE Register (offset = 134h) [reset = FFFFFFFFh]

GPIO\_OE is shown in [Figure 25-21](#) and described in [Table 25-20](#).

The GPIO\_OE register is used to enable the pins output capabilities. At reset, all the GPIO related pins are configured as input and output capabilities are disabled. This register is not used within the module, its only function is to carry the pads configuration. When the application is using a pin as an output and does not want interrupt generation from this pin, the application can/has to properly configure the Interrupt Enable registers.

**Figure 25-21. GPIO\_OE Register**

| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| OUTPUTEN[n]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-FFFFFFFFFFh |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-20. GPIO\_OE Register Field Descriptions**

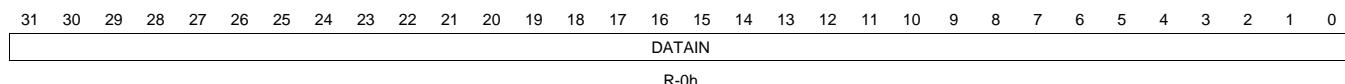
| Bit  | Field       | Type | Reset     | Description   |
|------|-------------|------|-----------|---|
| 31-0 | OUTPUTEN[n] | R/W  | FFFFFFFFh | Output Data Enable<br>0x0 = The corresponding GPIO port is configured as an output.<br>0x1 = The corresponding GPIO port is configured as an input. |

#### **25.4.1.16 GPIO\_DATAIN Register (offset = 138h) [reset = 0h]**

GPIO\_DATAIN is shown in [Figure 25-22](#) and described in [Table 25-21](#).

The GPIO\_DATAIN register is used to register the data that is read from the GPIO pins. The GPIO\_DATAIN register is a read-only register. The input data is sampled synchronously with the interface clock and then captured in the GPIO\_DATAIN register synchronously with the interface clock. So, after changing, GPIO pin levels are captured into this register after two interface clock cycles (the required cycles to synchronize and to write the data). When the AUTOIDLE bit in the system configuration register (GPIO\_SYS CONFIG) is set, the GPIO\_DATAIN read command has a 3 OCP cycle latency due to the data in sample gating mechanism. When the AUTOIDLE bit is not set, the GPIO\_DATAIN read command has a 2 OCP cycle latency.

**Figure 25-22. GPIO\_DATAIN Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-21. GPIO\_DATAIN Register Field Descriptions**

| Bit  | Field  | Type | Reset | Description        |
|------|--------|------|-------|--------------------|
| 31-0 | DATAIN | R    | 0h    | Sampled Input Data |

#### 25.4.1.17 GPIO\_DATAOUT Register (offset = 13Ch) [reset = 0h]

GPIO\_DATAOUT is shown in [Figure 25-23](#) and described in [Table 25-22](#).

The GPIO\_DATAOUT register is used for setting the value of the GPIO output pins. Data is written to the GPIO\_DATAOUT register synchronously with the interface clock. This register can be accessed with direct read/write operations or using the alternate Set/Clear feature. This feature enables to set or clear specific bits of this register with a single write access to the set data output register (GPIO\_SETDATAOUT) or to the clear data output register (GPIO\_CLEARDATAOUT) address.

**Figure 25-23. GPIO\_DATAOUT Register**

|         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31      | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAOUT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-22. GPIO\_DATAOUT Register Field Descriptions**

| Bit  | Field   | Type | Reset | Description                |
|------|---------|------|-------|----------------------------|
| 31-0 | DATAOUT | R/W  | 0h    | Data to set on output pins |

#### **25.4.1.18 GPIO\_LEVELDETECT0 Register (offset = 140h) [reset = 0h]**

GPIO\_LEVELDETECT0 is shown in [Figure 25-24](#) and described in [Table 25-23](#).

The GPIO\_LEVELDETECT0 register is used to enable/disable for each input lines the low-level (0) detection to be used for the interrupt request generation. Enabling at the same time high-level detection and low-level detection for one given pin makes a constant interrupt generator.

**Figure 25-24. GPIO\_LEVELDETECT0 Register**

|                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LEVELDETECT0[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-23. GPIO\_LEVELDETECT0 Register Field Descriptions**

| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-0 | LEVELDETECT0[n] | R/W  | 0h    | Low Level Interrupt Enable<br>0x0 = Disable the IRQ assertion on low-level detect.<br>0x1 = Enable the IRQ assertion on low-level detect. |

### 25.4.1.19 GPIO\_LEVELDETECT1 Register (offset = 144h) [reset = 0h]

GPIO\_LEVELDETECT1 is shown in [Figure 25-25](#) and described in [Table 25-24](#).

The GPIO\_LEVELDETECT1 register is used to enable/disable for each input lines the high-level (1) detection to be used for the interrupt request generation. Enabling at the same time high-level detection and low-level detection for one given pin makes a constant interrupt generator.

**Figure 25-25. GPIO\_LEVELDETECT1 Register**

|                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LEVELDETECT1[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-24. GPIO\_LEVELDETECT1 Register Field Descriptions**

| Bit  | Field           | Type | Reset | Description  |
|------|-----------------|------|-------|--|
| 31-0 | LEVELDETECT1[n] | R/W  | 0h    | High Level Interrupt Enable<br>0x0 = Disable the IRQ assertion on high-level detect.<br>0x1 = Enable the IRQ assertion on high-level detect. |

### **25.4.1.20 GPIO\_RISINGDETECT Register (offset = 148h) [reset = 0h]**

GPIO\_RISINGDETECT is shown in [Figure 25-26](#) and described in [Table 25-25](#).

The GPIO\_RISINGDETECT register is used to enable/disable for each input lines the rising-edge (transition 0 to 1) detection to be used for the interrupt request generation.

**Figure 25-26. GPIO\_RISINGDETECT Register**

|                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31              | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RISINGDETECT[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-25. GPIO\_RISINGDETECT Register Field Descriptions**

| Bit  | Field           | Type | Reset | Description   |
|------|-----------------|------|-------|---|
| 31-0 | RISINGDETECT[n] | R/W  | 0h    | Rising Edge Interrupt Enable<br>0x0 = Disable IRQ on rising-edge detect.<br>0x1 = Enable IRQ on rising-edge detect. |

### 25.4.1.21 GPIO\_FALLINGDETECT Register (offset = 14Ch) [reset = 0h]

GPIO\_FALLINGDETECT is shown in [Figure 25-27](#) and described in [Table 25-26](#).

The GPIO\_FALLINGDETECT register is used to enable/disable for each input lines the falling-edge (transition 1 to 0) detection to be used for the interrupt request generation.

**Figure 25-27. GPIO\_FALLINGDETECT Register**

|                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31               | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FALLINGDETECT[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-26. GPIO\_FALLINGDETECT Register Field Descriptions**

| Bit  | Field            | Type | Reset | Description  |
|------|------------------|------|-------|--|
| 31-0 | FALLINGDETECT[n] | R/W  | 0h    | Falling Edge Interrupt Enable<br>0x0 = Disable IRQ on falling-edge detect.<br>0x1 = Enable IRQ on falling-edge detect. |

### **25.4.1.22 GPIO\_DEBOUNCENABLE Register (offset = 150h) [reset = 0h]**

GPIO\_DEBOUNCENABLE is shown in [Figure 25-28](#) and described in [Table 25-27](#).

The GPIO\_DEBOUNCENABLE register is used to enable/disable the debouncing feature for each input line.

**Figure 25-28. GPIO\_DEBOUNCENABLE Register**

|                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31                | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEBOUNCEENABLE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-27. GPIO\_DEBOUNCENABLE Register Field Descriptions**

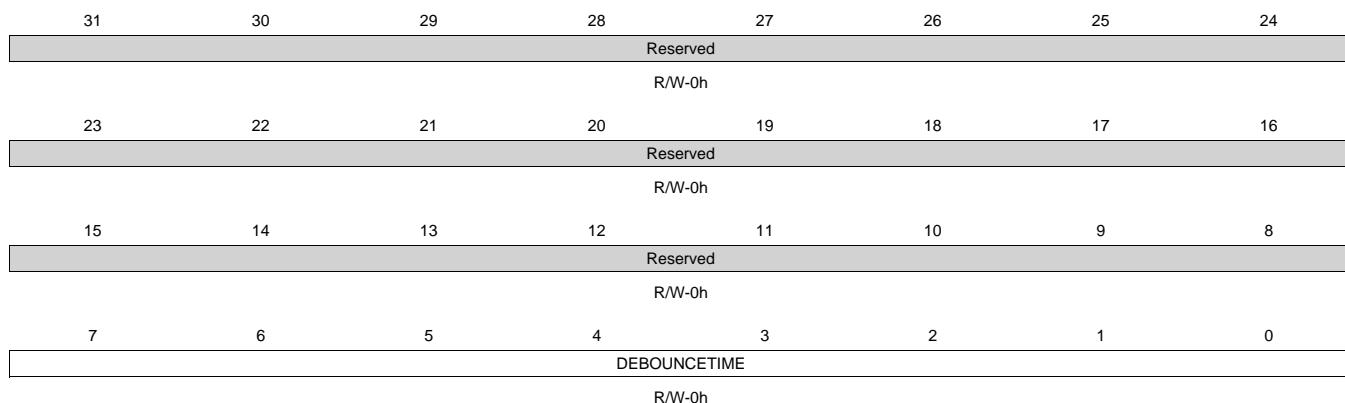
| Bit  | Field             | Type | Reset | Description  |
|------|-------------------|------|-------|--|
| 31-0 | DEBOUNCEENABLE[n] | R/W  | 0h    | Input Debounce Enable<br>0x0 = Disable debouncing feature on the corresponding input port.<br>0x1 = Enable debouncing feature on the corresponding input port. |

### 25.4.1.23 GPIO\_DEBOUNCINGTIME Register (offset = 154h) [reset = 0h]

GPIO\_DEBOUNCINGTIME is shown in [Figure 25-29](#) and described in [Table 25-28](#).

The GPIO\_DEBOUNCINGTIME register controls debouncing time (the value is global for all ports). The debouncing cell is running with the debouncing clock (32 kHz), this register represents the number of the clock cycle(s) (31 s long) to be used.

**Figure 25-29. GPIO\_DEBOUNCINGTIME Register**



LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-28. GPIO\_DEBOUNCINGTIME Register Field Descriptions**

| Bit  | Field        | Type | Reset | Description   |
|------|--------------|------|-------|---|
| 31-8 | Reserved     | R/W  | 0h    |   |
| 7-0  | DEBOUNCETIME | R/W  | 0h    | Input Debouncing Value in 31 microsecond steps.<br>Debouncing time = (DEBOUNCETIME + 1) - 31 microseconds |

#### **25.4.1.24 GPIO\_CLEARDATAOUT Register (offset = 190h) [reset = 0h]**

GPIO\_CLEARDATAOUT is shown in [Figure 25-30](#) and described in [Table 25-29](#).

Writing a 1 to a bit in the GPIO\_CLEARDATAOUT register clears to 0 the corresponding bit in the GPIO\_DATAOUT register; writing a 0 has no effect. A read of the GPIO\_CLEARDATAOUT register returns the value of the data output register (GPIO\_DATAOUT).

**Figure 25-30. GPIO\_CLEARDATAOUT Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-29. GPIO\_CLEARDATAOUT Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Clear Data Output Register<br>0x0 = No effect<br>0x1 = Clear the corresponding bit in the GPIO_DATAOUT register. |

### 25.4.1.25 GPIO\_SETDATAOUT Register (offset = 194h) [reset = 0h]

GPIO\_SETDATAOUT is shown in [Figure 25-31](#) and described in [Table 25-30](#).

Writing a 1 to a bit in the GPIO\_SETDATAOUT register sets to 1 the corresponding bit in the GPIO\_DATAOUT register; writing a 0 has no effect. A read of the GPIO\_SETDATAOUT register returns the value of the data output register (GPIO\_DATAOUT).

**Figure 25-31. GPIO\_SETDATAOUT Register**

|            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 31         | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTLINE[n] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |
| R/W-0h     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

**Table 25-30. GPIO\_SETDATAOUT Register Field Descriptions**

| Bit  | Field      | Type | Reset | Description  |
|------|------------|------|-------|--|
| 31-0 | INTLINE[n] | R/W  | 0h    | Set Data Output Register<br>0x0 = No effect<br>0x1 = Set the corresponding bit in the GPIO_DATAOUT register. |

**Initialization**

This chapter describes the initialization of the device.

| Topic                                    | Page        |
|--|-------------|
| <b>26.1 Functional Description .....</b> | <b>4048</b> |

## 26.1 Functional Description

This chapter describes the booting functionality of the device, referred hereafter as ROM Code. The booting functionality covers the following features:

**Memory Booting:** Booting the device by starting code stored on permanent memories like flash-memory or memory cards. This process is usually performed after either device cold or warm reset.

**Peripheral Booting:** Booting the device by downloading the executable code over a communication interface like UART, USB or Ethernet. This process is intended for flashing a device.

The device always starts up in secure mode. The ROM Code takes care of early initialization. The ROM code switches the device into public mode. Hence the Public ROM Code provides run-time services for cache maintenance.

### 26.1.1 Architecture

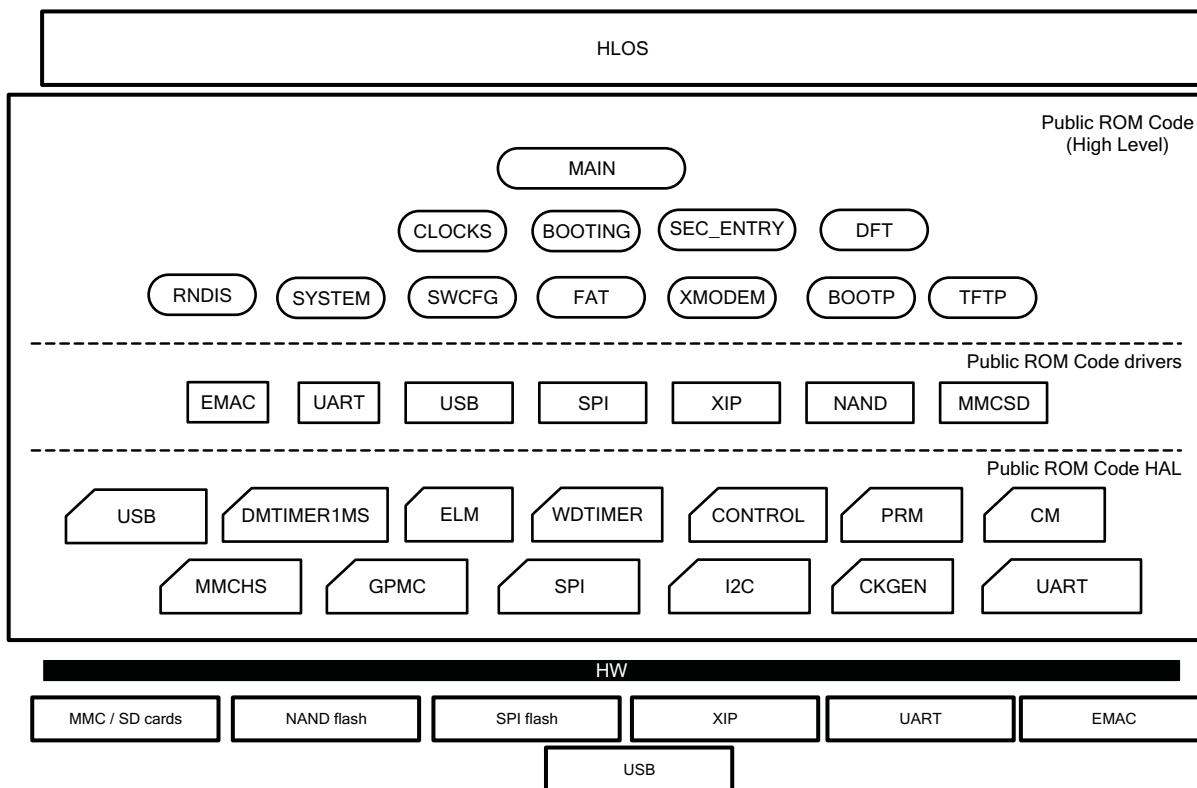
The architecture of the Public ROM Code is shown in [Figure 26-1](#). It is split into three main layers with a top-down approach: high-level, drivers, and hardware abstraction layer (HAL). One layer communicates with a lower level layer through a unified interface.

The high level layer is in charge of the main tasks of the Public ROM Code: watchdog and clocks configuration and main booting routine.

The drivers layer implements the logical and communication protocols for any booting device in accordance with the interface specification.

Finally the HAL implements the lowest level code for interacting with the hardware infrastructure IPs. End booting devices are attached to device IO pads.

**Figure 26-1. Public ROM Code Architecture**



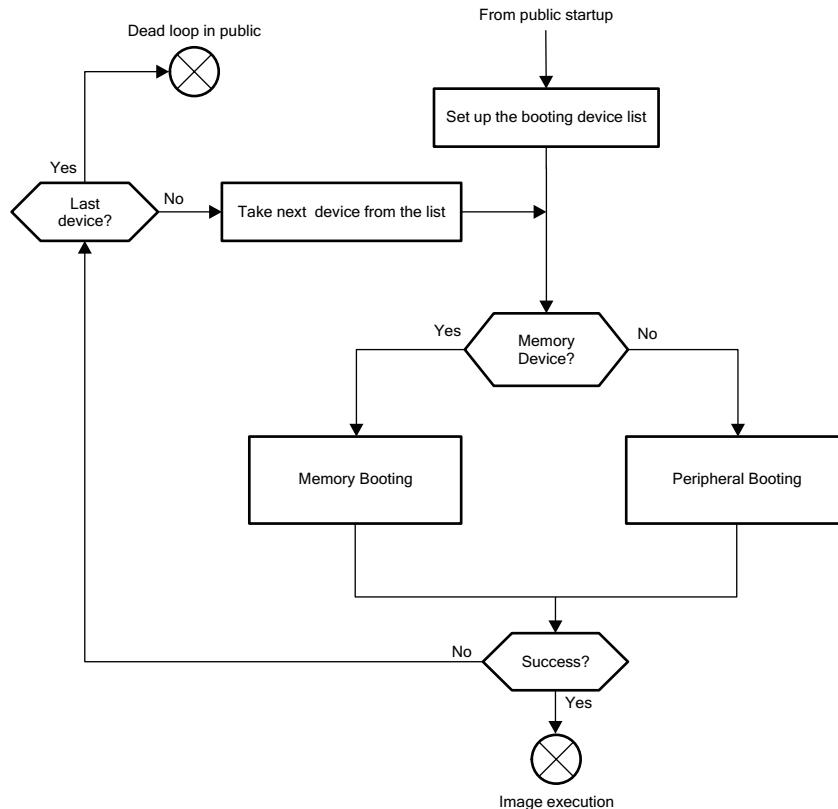
### 26.1.2 Functionality

[Figure 26-2](#) illustrates the high level flow for the Public ROM Code booting procedure. The ROM Code performs platform configuration and initialization as part of the public start-up procedure.

The booting device list is created based on the **SYSBOOT** pins. A booting device can be a **memory booting device** (soldered flash memory or temporarily booting device like memory card) or a **peripheral interface** connected to a host.

The main loop of the booting procedure goes through the booting device list and tries to search for an image from the currently selected booting device. This loop is exited if a valid booting image is found and successfully executed or upon watchdog expiration.

**Figure 26-2. Public ROM Code Boot Procedure**

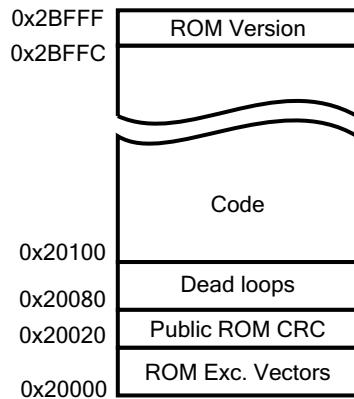


### 26.1.3 Memory Map

#### 26.1.3.1 Public ROM Memory Map

The on-chip ROM memory map is shown in [Figure 26-3](#). The Public ROM Code mapping consists in the following:

- Exception vectors
- CRC
- Dead loops collection
- Code and const data sections
- ROM Version

**Figure 26-3. ROM Memory Map**


### **Public ROM Exception Vectors**

**Table 26-1** lists the Public ROM exception vectors. The reset exception is redirected to the Public ROM Code startup. Other exceptions are redirected to their RAM handlers by loading appropriate addresses into the PC register.

**Table 26-1. ROM Exception Vectors**

| Address | Exception       | Content                               |
|---------|-----------------|---------------------------------------|
| 20000h  | Reset           | Branch to the Public ROM Code startup |
| 20004h  | Undefined       | PC = 4030CE04h                        |
| 20008h  | SWI             | PC = 4030CE08h                        |
| 2000Ch  | Pre-fetch abort | PC = 4030CE0Ch                        |
| 20010h  | Data abort      | PC = 4030CE10h                        |
| 20014h  | Unused          | PC = 4030CE14h                        |
| 20018h  | IRQ             | PC = 4030CE18h                        |
| 2001Ch  | FIQ             | PC = 4030CE1Ch                        |

### Public ROM Code CRC

The Public ROM Code CRC is calculated as 32 bit CRC code (CRC-32-IEEE 802.3) for the address range 20000h–2BFFFh. The four bytes CRC code is stored at location 20020h.

### Dead Loops

Built-in dead loops are used for different purposes as shown in below [Table 26-2](#). All dead loops are branch instructions coded in ARM mode. The fixed location of these dead loops facilitates debugging and testing. The first seven dead loops are default exception handlers linked with RAM exception vectors. The dead loops might be called directly from the user code. However there exists a special function which can be called from ROM Code in order to execute a dead loop. The function is an assembly code in ARM mode which takes the dead loop address from R0 register. The main purpose of the function is to go in a dead loop until the watchdog expires and resets the device. The function is located at address 200C0h. In addition the function clears global cold reset status upon issuing the global SW reset.

**Table 26-2. Dead Loops**

| Address | Purpose                                   |
|---------|---|
| 20080h  | Undefined exception default handler       |
| 20084h  | SWI exception default handler             |
| 20088h  | Pre-fetch abort exception default handler |
| 2008Ch  | Data abort exception default handler      |
| 20090h  | Unused exception default handler          |
| 20094h  | IRQ exception default handler             |
| 20098h  | FIQ exception default handler             |
| 2009Ch  | Validation tests PASS                     |
| 200A0h  | Validation tests FAIL                     |
| 200A4h  | Reserved                                  |
| 200A8h  | Image not executed or returned.           |
| 200ACh  | Reserved                                  |
| 200B0h  | Reserved                                  |
| 200B4h  | Reserved                                  |
| 200B8h  | Reserved                                  |
| 200BCh  | Reserved                                  |

### Code

This space is used to hold code and constant data.

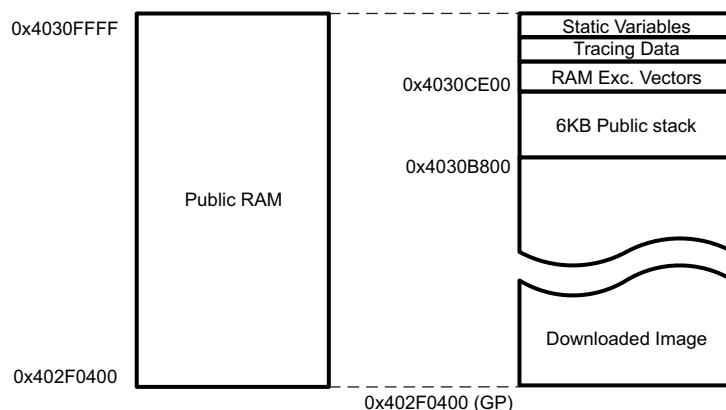
### Public ROM Code Version

The ROM Code version consists of two decimal numbers: major and minor. It can be used to identify the ROM Code release version in a given IC (e.g., useful to identify an IC version). The ROM Code version is a 32bits hexadecimal value located at address 2BFFCh.

### 26.1.3.2 Public RAM Memory Map

The Public ROM Code makes use of the on chip RAM module connected to the L3 interconnect (further referred to as **L3 RAM**). Its usage is shown in [Figure 26-4](#). The Public RAM memory map ranges from address 402F0400h to 4030FFFFh on a **GP Device**.

**Figure 26-4. Public RAM Memory Map**



#### Download Image

This area is used by the Public ROM Code to store the downloaded boot image. It can be up to 109KB.

#### Public Stack

Space reserved for stack.

#### RAM Exception Vectors

The RAM exception vectors enable a simple means for redirecting exceptions to custom handlers. [Table 26-3](#) shows content of the RAM space reserved for RAM vectors. The first seven addresses are ARM instructions which load the value located in the subsequent seven addresses into the PC register. These instructions are executed when an exception occurs since they are called from the ROM exception vectors. Undefined, SWI, Unused and FIQ exceptions are redirected to a hardcoded dead loop. Pre-fetch abort, data abort, and IRQ exception are redirected to pre-defined ROM handlers. User code can redirect any exception to a custom handler either by writing its address to the appropriate location from 4030CE24h to 4030CE3Ch or by overriding the branch (load into PC) instruction between addresses from 4030CE04h to 4030CE1Ch.

**Table 26-3. RAM Exception Vectors**

| Address   | Exception       | Content          |
|-----------|-----------------|------------------|
| 4030CE00h | Reserved        | Reserved         |
| 4030CE04h | Undefined       | PC = [4030CE24h] |
| 4030CE08h | SWI             | PC = [4030CE28h] |
| 4030CE0Ch | Pre-fetch abort | PC = [4030CE2Ch] |
| 4030CE10h | Data abort      | PC = [4030CE30h] |
| 4030CE14h | Unused          | PC = [4030CE34h] |
| 4030CE18h | IRQ             | PC = [4030CE38h] |
| 4030CE1Ch | FIQ             | PC = [4030CE3Ch] |
| 4030CE20h | Reserved        | 20090h           |
| 4030CE24h | Undefined       | 20080h           |
| 4030CE28h | SWI             | 20084h           |

**Table 26-3. RAM Exception Vectors (continued)**

| Address   | Exception       | Content   |
|-----------|-----------------|---|
| 4030CE2Ch | Pre-fetch abort | Address of default pre-fetch abort handler <sup>(1)</sup> |
| 4030CE30h | Data abort      | Address of default data abort handler <sup>(1)</sup>      |
| 4030CE34h | Unused          | 20090h  |
| 4030CE38h | IRQ             | Address of default IRQ handler                            |
| 4030CE3Ch | FIQ             | 20098h  |

<sup>(1)</sup> The default handlers for pre-fetch and data abort are performing reads from CP15 debug registers to retrieve the reason of the abort:

- In case of pre-fetch abort: the IFAR register is read from CP15 and stored into R0. The IFSR register is read and stored into the R1 register. Then the ROM Code jumps to the pre-fetch abort dead loop (20088h).
- In case of data abort: the DFAR register is read from CP15 and stored into R0. The DFSR register is read and stored into the R1 register. Then the ROM Code jumps to the data abort dead loop (2008Ch).

### Tracing Data

This area contains trace vectors reflecting the execution path of the public boot. [Section 26.1.12](#) describes the usage of the different trace vectors and lists all the possible trace codes.

**Table 26-4. Tracing Data**

| Address   | Size[bytes] | Description  |
|-----------|-------------|--|
| 4030CE40h | 4           | Current tracing vector, word 1                         |
| 4030CE44h | 4           | Current tracing vector, word 2                         |
| 4030CE48h | 4           | Current tracing vector, word 3                         |
| 4030CE4Ch | 4           | Current copy of the PRM_RSTST register (reset reasons) |
| 4030CE50h | 4           | Cold reset run tracing vector, word 1                  |
| 4030CE54h | 4           | Cold reset run tracing vector, word 2                  |
| 4030CE58h | 4           | Cold reset run tracing vector, word 3                  |
| 4030CE5Ch | 4           | Reserved   |
| 4030CE60h | 4           | Reserved   |
| 4030CE64h | 4           | Reserved   |

### Static Variables

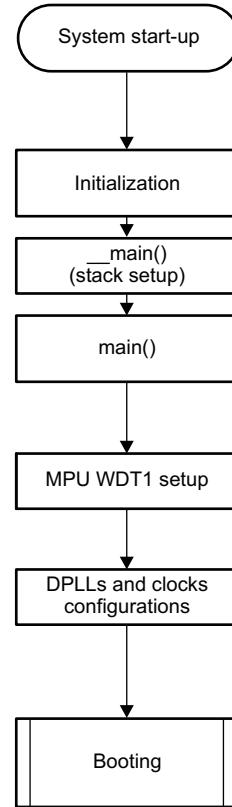
This area contains the ROM Code static variables used during boot time.

#### 26.1.4 Start-up and Configuration

##### 26.1.4.1 ROM Code Start-up

The Public ROM Code is physically located at the address 20000h.

**Figure 26-5. ROM Code Startup Sequence**



Upon system startup, the CPU performs the public-side initialization and stack setup (compiler auto generated C- initialization or “scatter loading”). Then it configures the watchdog timer 1 (set to three minutes), performs system clocks configuration. Finally it jumps to the booting routine.

#### 26.1.4.2 CPU State at Public Startup

The CPU **L1 instruction cache** and **branch prediction** mechanisms are not activated as part of the public boot process. The public vector base address is configured to the reset vector of Public ROM Code (20000h). MMU is left switched off during the public boot (hence L1 data cache off).

#### 26.1.4.3 Clocking Configuration

The device supports the following frequencies based on SYSBOOT[15:14]

**Table 26-5. Crystal Frequencies Supported**

| SYSBOOT[15:14] | Crystal Frequency |
|----------------|-------------------|
| 00b            | 19.2 MHz          |
| 01b            | 24 MHz            |
| 10b            | 25 MHz            |
| 11b            | 26 MHz            |

The ROM Code configures the clocks and DPLLs which are necessary for ROM Code execution:

- L3 ADPLLS locked to provide 200MHz clocks for peripheral blocks.
- DDR DPLL locked to provide 400MHz.
- MPU ADPLLS is locked to provide 500 MHz for the A8.
- PER ADPLLLJ is locked to provide 960MHz and 192MHz for peripheral blocks.

**Table 26-6** summarizes the ROM Code default settings for clocks. This default configuration enables all the ROM Code functionalities with minimized needs on power during boot.

**Table 26-6. ROM Code Default Clock Settings**

| Clock       | Frequency [MHz] | Source        |
|-------------|-----------------|---------------|
| L3F_CLK     | 200             | CORE_CLKOUTM4 |
| SPI_CLK     | 48              | PER_CLKOUTM2  |
| MMC_CLK     | 96              | PER_CLKOUTM2  |
| UART_CLK    | 48              | PER_CLKOUTM2  |
| I2C_CLK     | 48              | PER_CLKOUTM2  |
| MPU_CLK     | 500             | MPU_PLL       |
| USB_PHY_CLK | 960 MHz         | PER_CLKDCOLDO |

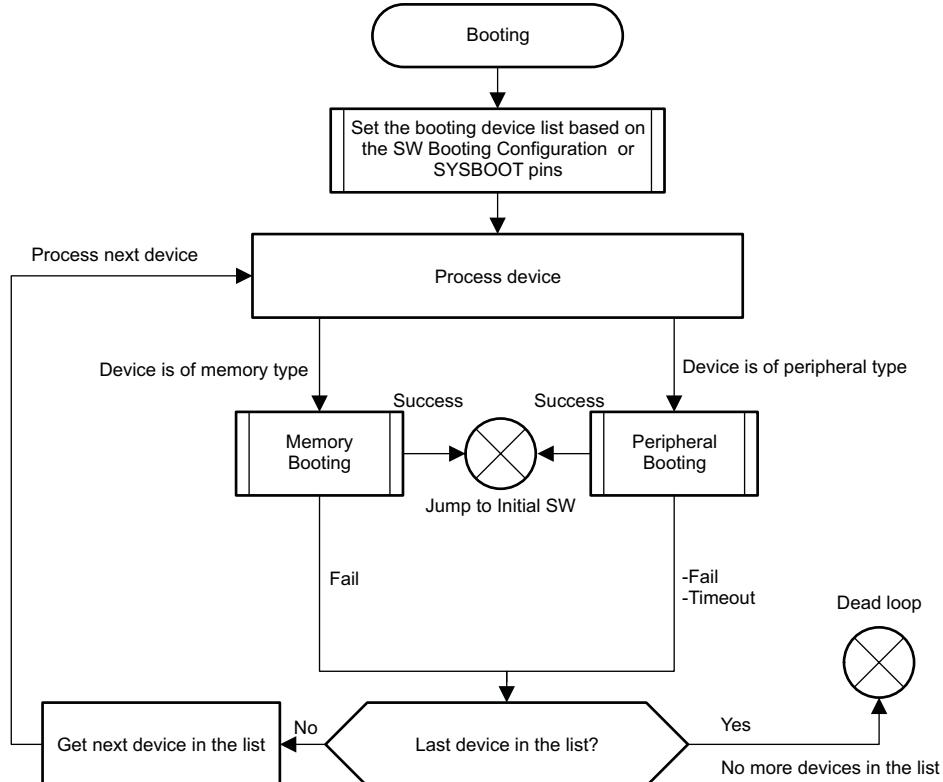
The DPLLs and PRCM clock dividers are configured with the ROM Code default values after cold or warm reset in order to give the same working conditions to the Public ROM Code sequence.

## 26.1.5 Booting

### 26.1.5.1 Overview

Figure 26-6 shows the booting procedure. First a **booting device list** is created. The list consists of all devices which will be searched for a booting image. The list is filled in based on the **SYSBOOT**.

**Figure 26-6. ROM Code Booting Procedure**



Once the booting device list is set up, the booting routine examines the devices enumerated in the list sequentially and either executes the memory booting or peripheral booting procedure depending on the booting device type. The memory booting procedure is executed when the booting device type is one of NOR, NAND, MMC or SPI-EEPROM. The peripheral booting is executed when the booting device type is Ethernet, USB or UART.

The memory booting procedure reads data from a memory type device. If a valid booting image is found and successfully read from the external memory device, the code begins to execute.

The peripheral booting procedure downloads data from a host (commonly a PC) to the device device by means of Ethernet, USB or UART links. The ROM Code uses a host-slave logical protocol for synchronization. Upon successful UART, USB or Ethernet connection the host sends the image binary contents. The peripheral booting procedure is described in detail in [Section 26.1.8](#).

If the memory or peripheral booting fails for all devices enumerated in the device list then the ROM Code gets into a loop, waiting for the watchdog to reset the system.

## 26.1.5.2 Device List

The ROM Code creates the **device list** based on information gathered from the **SYSBOOT configuration pins** sensed in the control module. The pins are used to index the device table from which the list of devices is extracted.

### 26.1.5.2.1 SYSBOOT Configuration Pins

Table 26-7 contains the SYSBOOT configuration pins.

Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup>

| SYSBOOT[15:14]   | SYSBOOT[13:12]   | SYSBOOT[11:10]  | SYSBOOT[9]  | SYSBOOT[8]                            | SYSBOOT[7:6]                   | SYSBOOT[5]  | SYSBOOT[4:0]        | Boot Sequence |   |                                 |         |
|--|--|---|---|---------------------------------------|--------------------------------|---|---------------------|---------------|---|---------------------------------|---------|
| <b>For all boot modes:</b> Crystal Frequency               | <b>For all boot modes:</b> Set to 00b for normal operation | <b>For XIP boot:</b> Muxed or non-muxed device<br><b>For NAND boot:</b> must be 00b   | <b>For NAND and NANDI2C boot:</b> NAND ECC<br><b>For Fast External Boot:</b> WAIT enable <sup>[1]</sup> | <b>For XIP boot:</b> Bus width        | <b>For EMAC boot:</b> PHY mode | <b>For all boot modes:</b> CLKOUT1 output enabled/disabled on XDMA_EVEN_T_INTR0 |                     |               |   |                                 |         |
| CONTROL_STATUS[23:22]                                      | CONTROL_STATUS[21:20]                                      | CONTROL_STATUS[19:18]   | CONTROL_STATUS[17]  | CONTROL_STATUS[16]                    | CONTROL_STATUS[7:6]            | CONTROL_STATUS[5]   | CONTROL_STATUS[4:0] |               |   |                                 |         |
|  |  |   |   |                                       |                                |   |                     | 1st           | 2nd   | 3rd                             | 4th     |
|  |  |   |   |                                       |                                |   | 00000b              | Reserved      |   |                                 |         |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b (all other values reserved)                            | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved                                      | Don't care for ROM code <sup>[3]</sup>  | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code        | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled                                     | 00001b              | UART0         | XIP w/<br>WAIT <sup>[1]</sup><br>(MUX2)<br><sup>[2]</sup> | MMC0                            | SPI0    |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b (all other values reserved)                            | <b>For NAND boot:</b> must be 00b   | 0 = ECC done by ROM<br>1 = ECC handled by NAND  | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code        | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled                                     | 00010b              | UART0         | SPI0  | NAND                            | NANDI2C |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b (all other values reserved)                            | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved                                      | Don't care for ROM code   | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code        | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled                                     | 00011b              | UART0         | SPI0  | XIP<br>(MUX2)<br><sup>[2]</sup> | MMC0    |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b (all other values reserved)                            | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b> must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND  | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code        | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled                                     | 00100b              | UART0         | XIP w/<br>WAIT <sup>[1]</sup><br>(MUX1)                   | MMC0                            | NAND    |

**Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup> (continued)**

| SYSBOOT[15:14]   | SYSBOOT[13:12]                     | SYSBOOT[11:10]   | SYSBOOT[9]                                     | SYSBOOT[8]                            | SYSBOOT[7:6]  | SYSBOOT[5]                                  | SYSBOOT[4:0] | Boot Sequence |   |   |         |
|--|------------------------------------|--|--|---------------------------------------|---|---|--------------|---------------|---|---|---------|
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 00101b       | UART0         | XIP (MUX1)<br><sup>[2]</sup>                        | SPI0  | NANDI2C |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b   | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 00110b       | EMAC1         | SPI0  | NAND  | NANDI2C |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 00111b       | EMAC1         | MMC0  | XIP w/ WAIT <sup>[1]</sup> (MUX2)<br><sup>[2]</sup> | NAND    |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 0   | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01000b       | EMAC1         | MMC0  | XIP (MUX2)<br><sup>[2]</sup>                        | NANDI2C |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01001b       | EMAC1         | XIP w/ WAIT <sup>[1]</sup> (MUX1)<br><sup>[2]</sup> | NAND  | MMC0    |

Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup> (continued)

| SYSBOOT[15:14]   | SYSBOOT[13:12]                     | SYSBOOT[11:10]   | SYSBOOT[9]   | SYSBOOT[8]                            | SYSBOOT[7:6]  | SYSBOOT[5]                                  | SYSBOOT[4:0] | Boot Sequence      |                |                |          |
|--|------------------------------------|--|--|---------------------------------------|---|---|--------------|--------------------|----------------|----------------|----------|
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND   | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01010b       | EMAC1              | XIP (MUX1) [2] | SPI0           | NANDI2C  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b   | 0 = ECC done by ROM<br>1 = ECC handled by NAND   | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01011b       | USB0               | NAND           | SPI0           | MMC0     |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND   | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01100b       | USB0               | NAND           | XIP (MUX2) [2] | NANDI2C  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved<br><b>For NAND boot:</b><br>must be 00b | 0 = ECC done by ROM<br>1 = ECC handled by NAND   | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01101b       | USB0               | NAND           | XIP (MUX1) [2] | SPI0     |
|  |                                    |  |  |                                       |   |   | 01110b       | Reserved           |                |                |          |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For Fast External Boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved                               | <b>For Fast External Boot:</b><br>0 = WAIT signal not monitored during boot<br>1 = WAIT signal monitored during boot | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 01111b       | Fast External Boot | UART0          | EMAC1          | Reserved |

**Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup> (continued)**

| SYSBOOT[15:14]   | SYSBOOT[13:12]                     | SYSBOOT[11:10]   | SYSBOOT[9]                                     | SYSBOOT[8]                            | SYSBOOT[7:6]  | SYSBOOT[5]                                  | SYSBOOT[4:0] | Boot Sequence   |          |       |       |
|--|------------------------------------|--|--|---------------------------------------|---|---|--------------|---|----------|-------|-------|
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | 00b = non-muxed device<br>10b = muxed device<br>x1b = reserved | Don't care for ROM code                        | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10000b       | XIP (MUX1) <sup>[2]</sup><br><sup>[1]</sup>                           | UART0    | EMAC1 | MMC0  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | 00b = non-muxed device<br>10b = muxed device<br>x1b = reserved | Don't care for ROM code                        | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10001b       | XIP w/ WAIT <sup>[1]</sup><br>(MUX1) <sup>[2]</sup><br><sup>[1]</sup> | UART0    | EMAC1 | MMC0  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b                           | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10010b       | NAND  | NANDI 2C | USB0  | UART0 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b                           | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10011b       | NAND  | NANDI 2C | MMC0  | UART0 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b                           | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10100b       | NAND  | NANDI 2C | SPI0  | EMAC1 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For NAND boot:</b><br>must be 00b                           | 0 = ECC done by ROM<br>1 = ECC handled by NAND | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10101b       | NANDI2C   | MMC0     | EMAC1 | UART0 |

Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup> (continued)

| SYSBOOT[15:14]   | SYSBOOT[13:12]                     | SYSBOOT[11:10]   | SYSBOOT[9]              | SYSBOOT[8]                            | SYSBOOT[7:6]  | SYSBOOT[5]                                  | SYSBOOT[4:0] | Boot Sequence  |       |       |       |
|--|------------------------------------|--|-------------------------|---------------------------------------|---|---|--------------|--|-------|-------|-------|
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | Don't care for ROM code  | Don't care for ROM code | Don't care for ROM code               | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10110b       | SPI0   | MMC0  | UART0 | EMAC1 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | Don't care for ROM code  | Don't care for ROM code | Don't care for ROM code               | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 10111b       | MMC0   | SPI0  | UART0 | USB0  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | Don't care for ROM code  | Don't care for ROM code | Don't care for ROM code               | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11000b       | SPI0   | MMC0  | USB0  | UART0 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | Don't care for ROM code  | Don't care for ROM code | Don't care for ROM code               | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11001b       | SPI0   | MMC0  | EMAC1 | UART0 |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved | Don't care for ROM code | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11010b       | XIP (MUX2) <sup>[2]</sup><br><sup>[1]</sup>                              | UART0 | SPI0  | MMC0  |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For XIP boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved | Don't care for ROM code | 0 = 8-bit device<br>1 = 16-bit device | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11011b       | XIP w/<br>WAIT <sup>[1]</sup><br>(MUX2) <sup>[2]</sup><br><sup>[1]</sup> | UART0 | SPI0  | MMC0  |

**Table 26-7. SYSBOOT Configuration Pins<sup>[4]</sup> (continued)**

| SYSBOOT[15:14]   | SYSBOOT[13:12]                     | SYSBOOT[11:10]   | SYSBOOT[9]   | SYSBOOT[8]                            | SYSBOOT[7:6]  | SYSBOOT[5]                                  | SYSBOOT[4:0] | Boot Sequence      |       |       |          |
|--|------------------------------------|--|--|---------------------------------------|---|---|--------------|--------------------|-------|-------|----------|
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | Don't care for ROM code  | Don't care for ROM code  | Don't care for ROM code               | Don't care for ROM code   | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11100b       | MMC1               | MMC0  | UART0 | USB0     |
|  |                                    |  |  |                                       |   |   | 11101b       | Reserved           |       |       |          |
|  |                                    |  |  |                                       |   |   | 11110b       | Reserved           |       |       |          |
| 00b = 19.2MHz<br>01b = 24MHz<br>10b = 25MHz<br>11b = 26MHz | 00b<br>(all other values reserved) | <b>For Fast External Boot:</b><br>00b = non-muxed device<br>10b = muxed device<br>x1b = reserved | <b>For Fast External Boot:</b><br>0 = WAIT signal not monitored during boot<br>1 = WAIT signal monitored during boot | 0 = 8-bit device<br>1 = 16-bit device | 00b = MII<br>01b = RMII<br>10b = Reserved<br>11b = RGMII w/o internal delay | 0 = CLKOUT1 disabled<br>1 = CLKOUT1 enabled | 11111b       | Fast External Boot | EMAC1 | UART0 | Reserved |

**SYSBOOT Configuration Pins Notes:**

1. WAIT is monitored on GPMC\_WAIT0.
2. MUX1 and MUX2 designate which group of XIP signals are used. Each group is defined in [Table 26-9](#).
3. Note that even though some bits may be a "don't care" for ROM code, all SYSBOOT values are latched into the CONTROL\_STATUS register and may be used by software after ROM execution has completed.
4. SYSBOOT[15:0] terminals are respectively LCD\_DATA[15:0] inputs, latched on the rising edge of PWRONRSTn.

The ROM Code uses the row pointed by the SYSBOOT pins value. The device list is filled in with the 1<sup>st</sup> to 4<sup>th</sup> devices.

[Table 26-7](#) is the decoding table for SYSBOOT pin configuration. The following shortcuts are used in the table:

- MMC1: MMC or SD card (MMC port 1)
- MMC0: MMC or SD card (MMC port 0)
- NAND / NANDI2C: NAND flash memory / read geometry from EEPROM on I2C0
- XIP: NOR or other XIP device without wait monitoring
- XIP w/ WAIT: NOR or other XIP device with wait monitoring
- MUX1: Boot with XIP\_MUX1 signals detailed in [Table 26-9](#)
- MUX2: Boot with XIP\_MUX2 signals detailed in [Table 26-9](#)
- UART0: UART interface (UART port 0)
- EMAC1: Ethernet interface (EMAC port 1)
- SPI0: SPI EEPROM (SPI 0, CS0)
- USB0: USB interface (USB0)

**Note:** For any SYSBOOT value that is selected, please be aware of the pin muxing implications. For example, if the boot mode selected is EMAC1, NAND, SPI0, NANDI2C, the device will drive EMAC, GPMC, SPI and I2C pins, in that order, depending on which boot device finally succeeds. Ensure that if a specific boot mode in the sequence chosen is not used that the components using those particular signals do not conflict with the ROM driving those signals (or external components are not in contention with the ROM using these signals). For specific details of the pins driven by each device, please refer the description of that boot device later in this chapter.

To extend the boot flow to boot from devices that are not natively supported by the ROM, SPI boot can be used. For example, to be able to boot from a USB stick, the system can be configured to boot from a SPI flash, and the code for configuring the USB and booting from a USB stick can be loaded into the SPI flash. This is known as a secondary boot.

The values corresponding to SYSBOOT[4:0]= x1111 provide a bypass mode booting feature.

The fast external boot feature consists of minimal execution by the ROM Code for configuring the GPMC interface and then directly jump to the code contained in the connected NOR flash device.

## 26.1.6 Fast External Booting

### 26.1.6.1 Overview

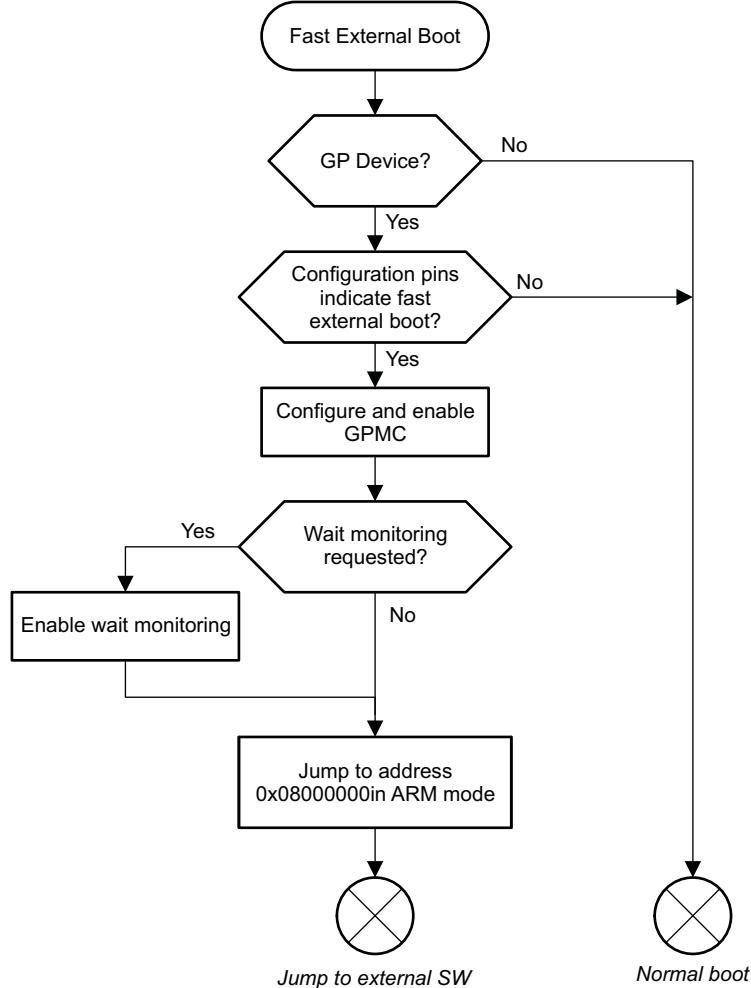
The Fast External boot feature:

- Consists of a blind jump in ARM mode to address 0x00000000 in an external XIP device connected to CS0
- The jump is performed with minimum on-chip ROM Code execution (only configures GPMC interface), without configuring any PLL
- Allows the customer to create its own booting code
- Is set up by means of the configuration pins, see [Table 26-7](#).
- Addr/Data muxed device or a non-muxed (selected using SYSBOOT[11:10]) device in connected in XIP\_MUX1 configuration
- Bus width selected by SYSBOOT[8]
- CS0 chip select
- Device wait signal connected to the WAIT0 GPMC signal (if used)
- The wait monitoring enable/disable is based on the value of SYSBOOT[9]

### 26.1.6.2 External Booting

[Figure 26-7](#) shows the Fast External Boot procedure. The code does not make use of RAM and is designed for fast execution.

**Figure 26-7. Fast External Boot**

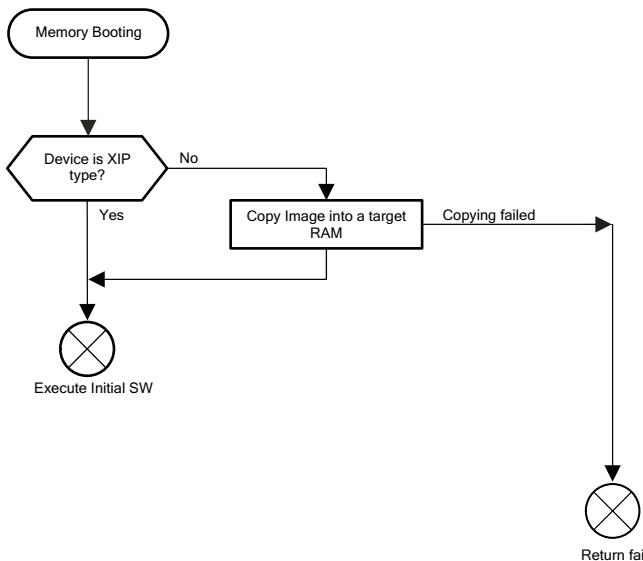


## 26.1.7 Memory Booting

### 26.1.7.1 Overview

The memory booting procedure takes care of starting an external code located in memory device types.

**Figure 26-8. Memory Booting**



There are two groups of memory booting devices distinguished by the need of **code shadowing**. The code shadowing means copying a code from a non-directly addressable device into a location (typically a RAM area) from where the code can be executed. Devices which are directly addressable are called eXecute In Place (XIP) devices.

The Memory Booting flowchart is shown in [Figure 26-8](#). The second step is about performing the shadowing of the image that is copying the image from external mass storage (non-XIP) into internal RAM. Failure in image copy results in Memory Booting returning to the main booting procedure which will select the next device for booting. The next sections detail procedures for device initialization and detection in addition to the description of the sector read routine for each supported device type. A sector is a logical unit of 512 bytes.

The detection of whether an image is present or not on a selected device depends on the first few bytes. On a **GP Device** type a booting image is considered to be present when the first four bytes of the sector is not equal to 0000 0000h or FFFF FFFFh.

During the first read sector call, sectors are copied to a temporary RAM buffer. Once the image is found and destination address is known, the content of the temporary buffer is moved to the target RAM location so it is needed to re-read the first image sector. On a **GP Device** the GP header is discarded, therefore only executable code is located in RAM with the first executable instruction located at the destination address.

The Image authentication and execution is detailed in [Section 26.1.10](#). For more information about image formats and contents, see [Section 26.1.9](#)

MMC/SD cards and NAND devices can hold up to four copies of the booting image. Therefore the ROM Code searches for one valid image out of the four if present by walking over the first four blocks of the mass storage space. Other XIP devices (NOR) use only one copy of the booting image.

## 26.1.7.2 XIP Memory

The ROM Code can boot directly from XIP devices. A typical XIP device is a NOR flash memory. Support for XIP devices is performed under the following assumptions:

- Uses GPMC as the communication interface
- Up to 1Gbit (128Mbytes) memories can be connected
- Both x8 and x16 data bus width
- Asynchronous protocol
- Supports address/data multiplexed mode and non-muxed mode
- GPMC clock is 50-MHz
- Device connected to CS0 mapped to address 0x08000000.
- Wait pin signal WAIT0 monitored depending on the SYSBOOT pin configuration (XIP / XIP w/ WAIT).
- Flexible muxing options for gpmc\_a0-gpmc\_a11 for non-muxed XIP devices

Depending on the SYSBOOT pins, the GPMC is configured to use the WAIT signal connected on the WAIT0 pin or not. Wait pin polarity is set to stall accessing memory when the WAIT0 pin is low. The wait monitoring is intended to be used with memories which require long time for initialization after reset or need to pause while reading data. The boot procedure from XIP device can be described as such:

- Configure GPMC for XIP device access.
- Set the image location to 0x08000000.
- Verify if bootable image is present at the image location.
- If the image has been found, start the execution.
- If the image has not been found, return from XIP booting to the main booting loop.

### 26.1.7.2.1 XIP Initialization and Detection

- GPMC initialization

[Figure 26-9](#) and [Table 26-8](#) describe the GPMC timing settings set for XIP boot and other address-data accessible devices.

Figure 26-9. GPMC XIP Timings

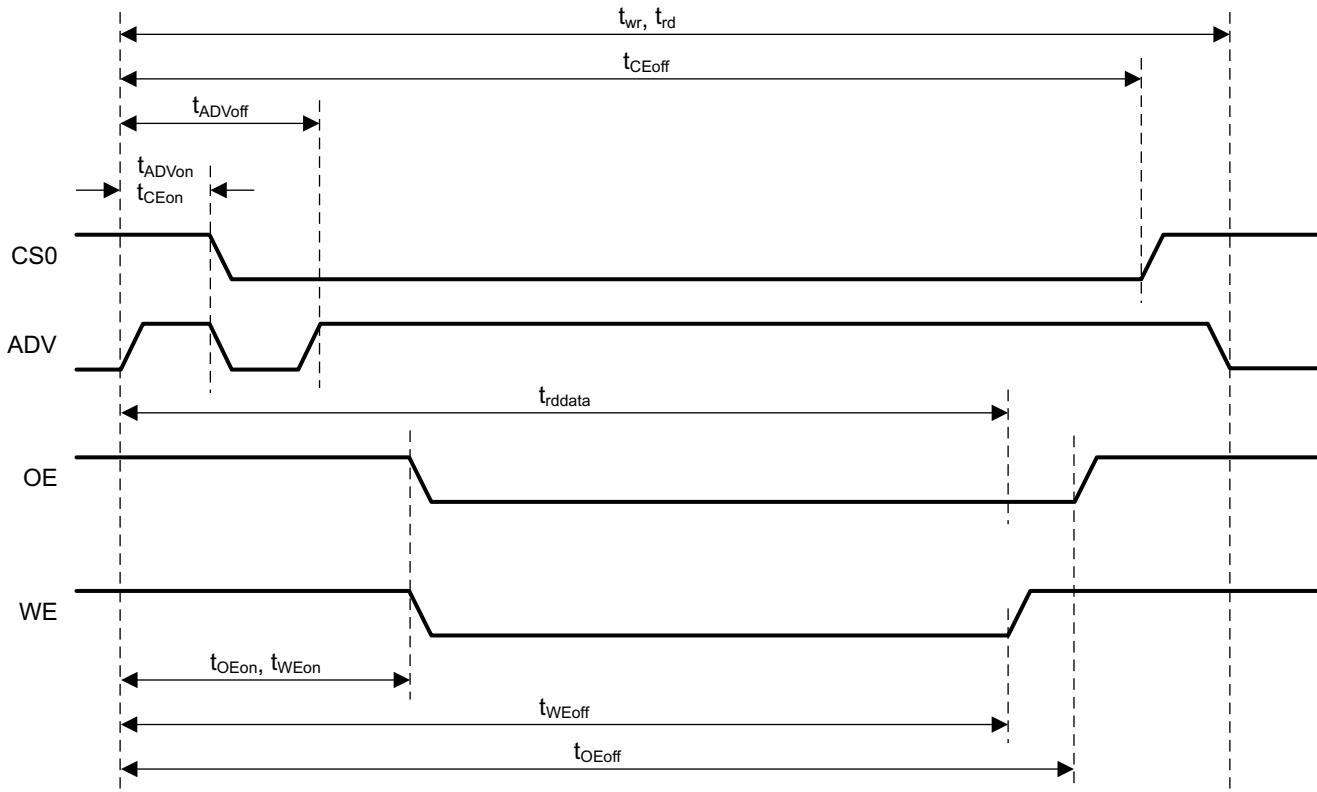


Table 26-8. XIP Timings Parameters

| Parameter           | Description        | Value [clock cycles] |
|---------------------|--------------------|----------------------|
| t <sub>wr</sub>     | write cycle period | 17                   |
| t <sub>rd</sub>     | read cycle period  | 17                   |
| t <sub>CEon</sub>   | CE low time        | 1                    |
| t <sub>CEoff</sub>  | CE high time       | 16                   |
| t <sub>ADVon</sub>  | ADV low time       | 1                    |
| t <sub>ADVoff</sub> | ADV high time      | 2                    |
| t <sub>OEon</sub>   | OE low time        | 3                    |
| t <sub>WEon</sub>   | WE low time        | 3                    |
| t <sub>rddata</sub> | data latch time    | 15                   |
| t <sub>OEoff</sub>  | OE high time       | 16                   |
| t <sub>WEoff</sub>  | WE high time       | 15                   |

The one clock cycle is 20ns, which corresponds to 50-MHz frequency.

- Device detection

There is no specific identification routine executed prior to booting from an XIP device.

#### 26.1.7.2.2 Pins Used

The list of device pins that are configured by the ROM in the case of NOR boot mode are as follows. Please note that all the pins might not be driven at boot time. The decision as to which pins need to be driven is done based on the type of NOR flash selected. The pins that are not listed below are not configured by the ROM code and are left at power-on defaults. Specifically, external logic is needed to isolate the upper address lines (A12–A27) of the NOR flash from the device pins and drive them low during boot. Once the initial software starts running, it can appropriately configure the pinmux setting for the lines and remove the isolation to allow GPMC to drive all the address lines.

**Table 26-9. Pins Used for NOR Boot**

| Signal name | Pin used in XIP_MUX1 <sup>(1)</sup> mode | Pin used in XIP_MUX2 <sup>(1)</sup> mode |
|-------------|--|--|
| CS0         | GPMC_CSNO                                | GPMC_CSNO                                |
| ADVN_ALE    | GPMC_ADVN_ALE                            | GPMC_ADVN_ALE                            |
| OEN_REN     | GPMC_OEN_REN                             | GPMC_OEN_REN                             |
| BE0N_CLE    | GPMC_BEN0_CLE                            | GPMC_BEN0_CLE                            |
| WEN         | GPMC_WEN                                 | GPMC_WEN                                 |
| WAIT        | GPMC_WAIT0                               | GPMC_WAIT0                               |
| AD0 - AD15  | GPMC_AD0 - GPMC_AD15                     | GPMC_AD0 - GPMC_AD15                     |
| A0          | GPMC_A0                                  | LCD_DATA0                                |
| A1          | GPMC_A1                                  | LCD_DATA1                                |
| A2          | GPMC_A2                                  | LCD_DATA2                                |
| A3          | GPMC_A3                                  | LCD_DATA3                                |
| A4          | GPMC_A4                                  | LCD_DATA4                                |
| A5          | GPMC_A5                                  | LCD_DATA5                                |
| A6          | GPMC_A6                                  | LCD_DATA6                                |
| A7          | GPMC_A7                                  | LCD_DATA7                                |
| A8          | GPMC_A8                                  | LCD_VSYNC                                |
| A9          | GPMC_A9                                  | LCD_HSYNC                                |
| A10         | GPMC_A10                                 | LCD_PCLK                                 |
| A11         | GPMC_A11                                 | LCD_AC_BIAS_EN                           |

<sup>(1)</sup> XIP\_MUX1 and XIP\_MUX2 do not correspond to the pin mux modes that are defined for each terminal. This table identifies which pins are used in each boot mode.

#### 26.1.7.2.3 Sysboot Pins

Some of the SYSBOOT pins have special meanings when NOR boot is selected.

**Table 26-10. Special SYSBOOT Pins for NOR Boot**

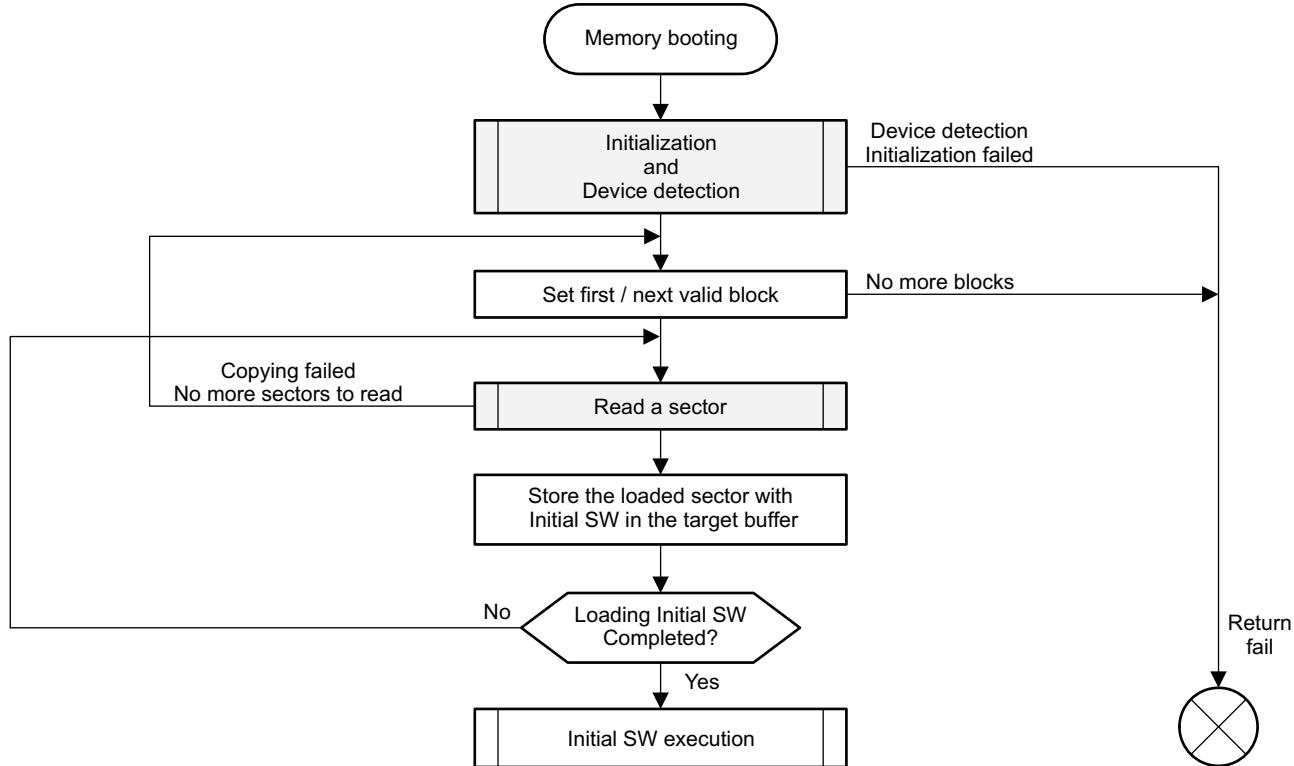
| SYSBOOT[n] | Description  |
|------------|--|
| [8]        | 0 = 8-bit device<br>1 = 16-bit device                          |
| [11:10]    | 00b = Non-muxed device<br>10b = Muxed device<br>x1b = Reserved |

### 26.1.7.3 Image Shadowing for Non-XIP Memories

#### 26.1.7.3.1 Shadowing on GP Device

The **GP Device** shadowing uses the following approach.

**Figure 26-10. Image Shadowing on GP Device**



### 26.1.7.4 NAND

The NAND flash memory is not XIP and requires shadowing before the code can be executed.

#### 26.1.7.4.1 Features

- Uses GPMC as the communication interface
- Device from 512Mbit (64MByte)
- x8 and x16 bus width
- Support for large page size (2048 bytes + 64 spare bytes) or very large page size 4096 bytes + 128 / 218 spare bytes)
- Only supports devices where Chip Select can be de-asserted during read, program or erase cycles, without interrupting the operation
- Device Identification based on ONFI or ROM table
- ECC correction : 8 bits/sector for most devices (16b/sector for devices with large spare area)
- Support for disabling ECC correction, so than the in-built ECC correction mechanisms on some NANDs can be used.
- GPMC timings adjusted for NAND access
- GPMC clock is 50MHz
- Device connected to GPMC\_CS0
- Wait pin signal GPMC\_WAIT0 connected to NAND BUSY output
- Four physical blocks are searched for an image. The block size depends on device.

#### 26.1.7.4.2 Initialization and Detection

The initialization routine for NAND devices consists in three parts: GPMC initialization, device detection with parameters determination and finally bad block detection.

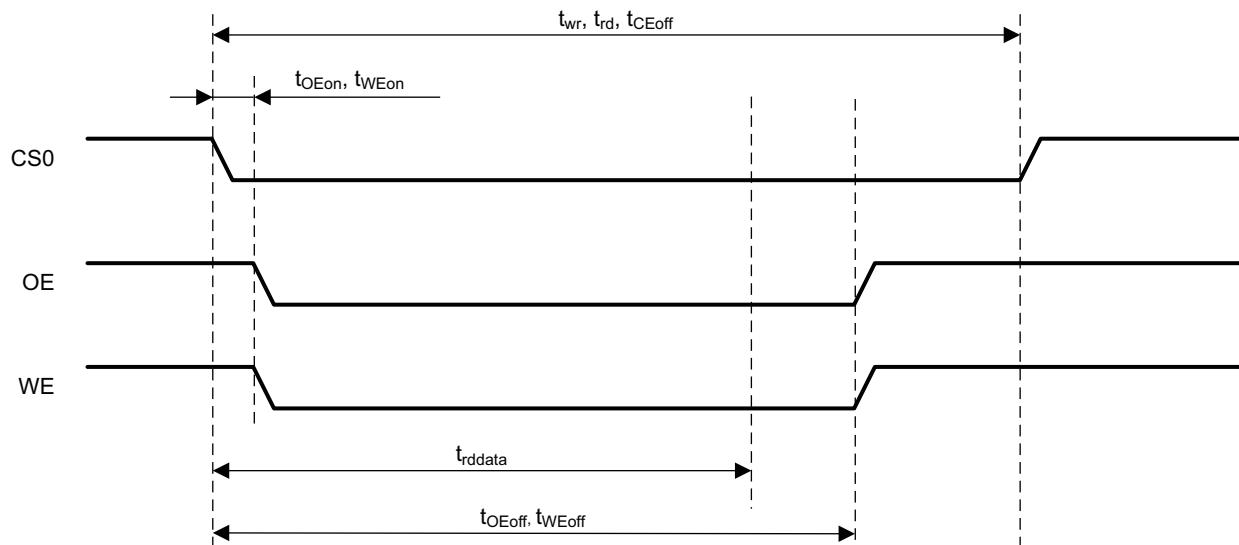
##### ONFI Support

The NAND identification starts with ONFI detection. For more information on ONFI standard, see the Open NAND Flash Interface Specification (<http://www.onfi.org>).

##### GPMC Initialization

The GPMC interface is configured as such it can be used for accessing NAND devices. The address bus is released since a NAND device does not use it. The data bus width is initially set to 16 bits; and changed to 8 bits if needed after device parameters determination. The following scheme is applied since NAND devices require different timings when compared to regular NOR devices:

**Figure 26-11. GPMC NAND Timings**



**Table 26-11. NAND Timings Parameters**

| Parameter    | Description                       | Value [clock cycles] |
|--------------|-----------------------------------|----------------------|
| $t_{wr}$     | write cycle period                | 30                   |
| $t_{rd}$     | read cycle period                 | 30                   |
| $t_{CEon}$   | CE low (not marked on the figure) | 0                    |
| $t_{OEon}$   | CE low to OE low time             | 7                    |
| $t_{WEon}$   | CE low to WE low time             | 5                    |
| $t_{rddata}$ | CE low to data latch time         | 21                   |
| $t_{OEoff}$  | CE low to OE high time            | 24                   |
| $t_{WEoff}$  | CE low to WE high time            | 22                   |

Figure 26-11 and Table 26-11 describes the timings configured for NAND device access. The one clock cycle is 20 ns, which correspond to 50-MHz frequency.

### Device Detection and Parameters

The ROM Code first performs an initial wait for device auto initialization (with 250ms timeout) with polling of the ready information. Then, it needs to identify the NAND type connected to the GPMC interface. The GPMC is initialized using 8 bits, asynchronous mode. The NAND device is reset (command FFh) and its status is polled until ready for operation (with 200ms timeout). The ONFI Read ID (command 90h / address 20h) is sent to the NAND device. If it replies with the ONFI signature (4 bytes) then a Read parameters page (command ECh) is sent. If the parameters page does not have the ONFI signature, then the ONFI identification fails. If the ONFI identification passes, the information shown in Table 26-12 is then extracted: page size, spare area size, number of pages per block, and the addressing mode. The remaining data bytes from the parameters page stream are simply ignored.

**Table 26-12. ONFI Parameters Page Description**

| Offset | Description                    | Size (bytes) |
|--------|--------------------------------|--------------|
| 6      | Features supported             | 2            |
| 80     | Number of data bytes per page  | 4            |
| 84     | Number of spare bytes per page | 2            |
| 92     | Number of pages per block      | 4            |
| 101    | Number of address cycles       | 1            |

If the ONFI Read ID command fails (it will be the case with any device not supporting ONFI) then the device is reset again with polling for device to be ready (with 200ms timeout). Then, the standard Read ID (command 90h / address 00h) is sent. If the Device ID (2<sup>nd</sup> byte of the ID byte stream) is recognized as being a supported device then the device parameters are extracted from an internal ROM Code table. The list of supported devices is shown in Table 26-13.

**Table 26-13. Supported NAND Devices**

| Capacity | Device ID | Bus Width | Page size |
|----------|-----------|-----------|-----------|
| 512 Mb   | F0        | x8        | 2048      |
| 512 Mb   | C0        | x16       | 2048      |
| 512 Mb   | A0        | x8        | 2048      |
| 512 Mb   | B0        | x16       | 2048      |
| 512 Mb   | F2        | x8        | 2048      |
| 512 Mb   | C2        | x16       | 2048      |
| 512 Mb   | A2        | x8        | 2048      |
| 512 Mb   | B2        | x16       | 2048      |
| 1 Gb     | F1        | x8        | 2048      |
| 1 Gb     | C1        | x16       | 2048      |
| 1 Gb     | A1        | x8        | 2048      |
| 1 Gb     | B1        | x16       | 2048      |
| 2 Gb     | DA        | x8        | 2048      |
| 2 Gb     | CA        | x16       | 2048      |
| 2 Gb     | AA        | x8        | 2048      |
| 2 Gb     | BA        | x16       | 2048      |
| 2 Gb     | 83        | x8        | 2048      |
| 2 Gb     | 93        | x16       | 2048      |
| 4 Gb     | DC        | x8        | 2048      |
| 4 Gb     | CC        | x16       | 2048      |
| 4 Gb     | AC        | x8        | 2048/4096 |
| 4 Gb     | BC        | x16       | 2048/4096 |
| 4 Gb     | 84        | x8        | 2048      |

**Table 26-13. Supported NAND Devices (continued)**

| Capacity | Device ID | Bus Width | Page size |
|----------|-----------|-----------|-----------|
| 4 Gb     | 94        | x16       | 2048      |
| 8 Gb     | D3        | x8        | 2048/4096 |
| 8 Gb     | C3        | x16       | 2048/4096 |
| 8 Gb     | A3        | x8        | 2048/4096 |
| 8 Gb     | B3        | x16       | 2048/4096 |
| 8 Gb     | 85        | x8        | 2048      |
| 8 Gb     | 95        | x16       | 2048      |
| 16 Gb    | D5        | x8        | 2048/4096 |
| 16 Gb    | C5        | x16       | 2048/4096 |
| 16 Gb    | A5        | x8        | 2048/4096 |
| 16 Gb    | B5        | x16       | 2048/4096 |
| 16 Gb    | 86        | x8        | 2048      |
| 16 Gb    | 96        | x16       | 2048      |
| 32 Gb    | D7        | x8        | 2048/4096 |
| 32 Gb    | C7        | x16       | 2048/4096 |
| 32 Gb    | A7        | x8        | 2048/4096 |
| 32 Gb    | B7        | x16       | 2048/4096 |
| 32 Gb    | 87        | x8        | 2048      |
| 32 Gb    | 97        | x16       | 2048      |
| 64 Gb    | DE        | x8        | 2048/4096 |
| 64 Gb    | CE        | x16       | 2048/4096 |
| 64 Gb    | AE        | x8        | 2048/4096 |
| 64 Gb    | BE        | x16       | 2048/4096 |

When the parameters are retrieved from the ROM table: page size and block size is updated based on 4<sup>th</sup> byte of NAND ID data. Due to inconsistency amongst different manufacturers, only devices which has been recognized to be at least 2Gb (included) have these parameters updated. Therefore, the ROM Code supports 4kB page devices but only if their size, according to the table, is at least 2Gb. Devices which are smaller than 2Gb have the block size parameter fixed to 128kB. [Table 26-14](#) shows the 4<sup>th</sup> ID Data byte encoding used in ROM Code.

**Table 26-14. 4th NAND ID Data Byte**

| Item              | Description | I/O # |   |   |   |   |   |          |          |
|-------------------|-------------|-------|---|---|---|---|---|----------|----------|
|                   |             | 7     | 6 | 5 | 4 | 3 | 2 | 1        | 0        |
| <b>Page Size</b>  | <b>1kB</b>  |       |   |   |   |   |   | <b>0</b> | <b>0</b> |
|                   | 2kB         |       |   |   |   |   |   | 0        | 1        |
|                   | 4kB         |       |   |   |   |   |   | 1        | 0        |
|                   | 8kB         |       |   |   |   |   |   | 1        | 1        |
| <b>Cell type</b>  | 2 levels    |       |   |   |   | 0 | 0 |          |          |
|                   | 4 levels    |       |   |   |   | 0 | 1 |          |          |
|                   | 8 levels    |       |   |   |   | 1 | 0 |          |          |
|                   | 16 levels   |       |   |   |   | 1 | 1 |          |          |
| <b>Block Size</b> | 64kB        |       |   | 0 | 0 |   |   |          |          |
|                   | 128kB       |       |   | 0 | 1 |   |   |          |          |
|                   | 256kB       |       |   | 1 | 0 |   |   |          |          |
|                   | 512kB       |       |   | 1 | 1 |   |   |          |          |

### Reading NAND Geometry From I2C EEPROM

ROM supports a special boot mode called NANDI2C to support NAND devices whose geometry cannot be detected by the ROM automatically using methods described in the previous section. If this boot mode is selected, the ROM code tries to read NAND geometry from an I2C EEPROM. If the read is successful, ROM code then proceeds to next steps of NAND boot, beginning with reading bad blocks information.

The list of device pins that are configured by the ROM in the case of NANDI2C boot mode. (This is in addition to the NAND boot pins described in the previous sections.)

**Table 26-15. Pins Used for NANDI2C Boot for I2C EEPROM Access**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| I2C SCL     | i2c0_scl           |
| I2C SDA     | i2c0_sda           |

ROM accesses the I2C EEPROM at I2C slave address **0x50** and reads 7 bytes starting from address offset **0x80**. The format of this (NAND geometry information) is as follows:

**Table 26-16. NAND Geometry Information on I2C EEPROM**

| Byte address | Information  |   |
|--------------|--|---|
|              | Upper nibble   | Lower nibble  |
| 0x80         |  | Magic Number – 0x10   |
| 0x81         |  | Magic Number – 0xb3   |
| 0x82         |  | Magic Number – 0x57   |
| 0x83         |  | Magic Number – 0xa6   |
| 0x84         | NAND column address (word/byte offset within a page) size in bytes, Example: 2 | NAND row address (page offset) size in bytes. Example: 3                    |
| 0x85         | Page size ( $2^N$ ) exponent “N”. Example (for page size of 2048): 11          | Pages per block ( $2^N$ ) exponent “N” Example (for number of blocks 64): 6 |
| 0x86         | NAND bus width<br>0 = 8-bit device<br>1 = 16-bit device                        | ECC Type<br>00b = No ECC<br>01b = BCH8<br>10b = BCH16<br>11b = Reserved     |

- ECC correction

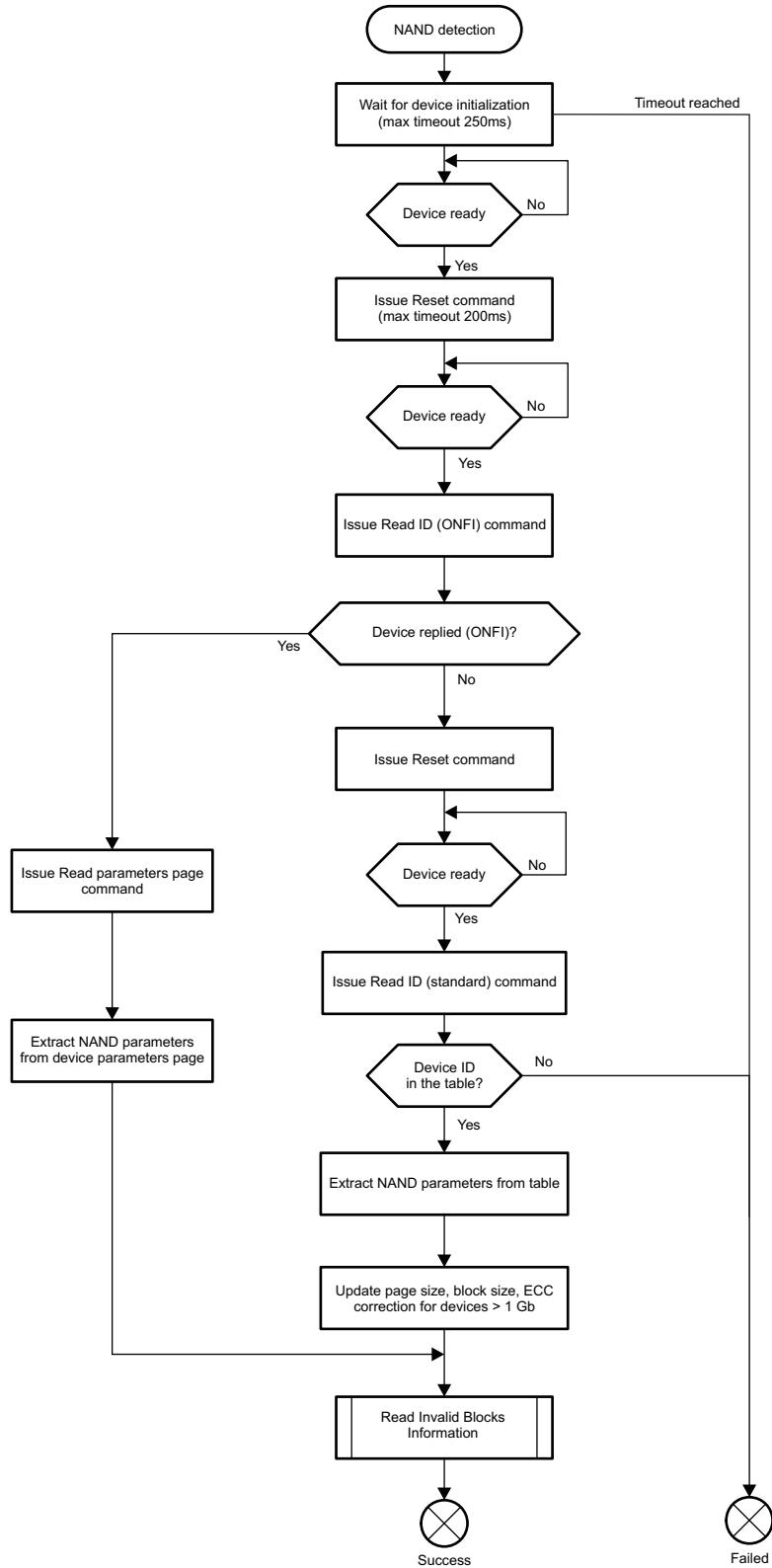
The default ECC correction applied is BCH 8b/sector using the GPMC and ELM hardware.

For device ID codes D3h, C3h, D5h, C5h, D7h, C7h, DEh, CEh when manufacturer code (first ID byte) is 98h the Cell type information is checked in the 4th byte of ID data. If it is equal to 10b then the ECC correction applied is BCH 16b/sector.

In addition ECC computation done by the ROM can be turned off completely by using SYSBOOT[9]. This is particularly useful when interfacing with NAND devices that have built in ECC engines.

**Table 26-17. ECC Configuration for NAND Boot**

| SYSBOOT[9] | ECC Computation     |
|------------|---------------------|
| 0          | ECC done by ROM     |
| 1          | ECC handled by NAND |

**Figure 26-12. NAND Device Detection**


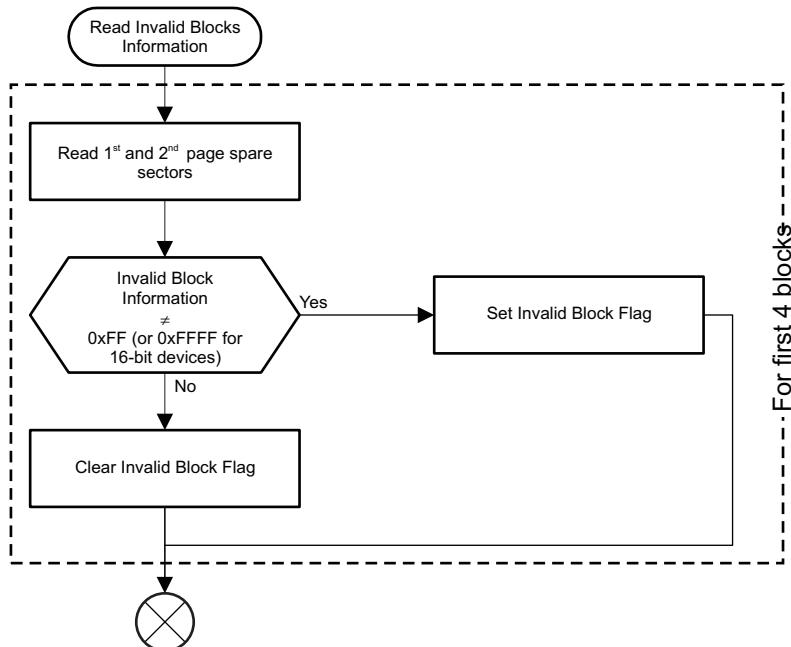
The detection procedure is described in [Figure 26-12](#). Once the device has been successfully detected, the ROM Code changes GPMC to 16-bit bus width if necessary.

### Bad Block Verification

Invalid blocks are blocks which contain invalid bits whose reliability cannot be guaranteed by the manufacturer. Those bits are identified in the factory or during the programming and reported in the initial invalid block information located in the spare area on the 1<sup>st</sup> and 2<sup>nd</sup> page of each block. Since the ROM Code is looking for an image in the first four blocks, it must detect block validity status of these blocks. Blocks which are detected as invalid are not accessed later on. Blocks validity status is coded in the spare areas of the first two pages of a block (first byte equal to FFh in 1<sup>st</sup> and 2<sup>nd</sup> pages for an 8 bits device / first word equal to FFFFh in 1<sup>st</sup> and 2<sup>nd</sup> page for a 16bits device).

[Figure 26-13](#) depicts the invalid block detection routine. The routine consists in reading spare areas and checking validity data pattern.

**Figure 26-13. NAND Invalid Blocks Detection**



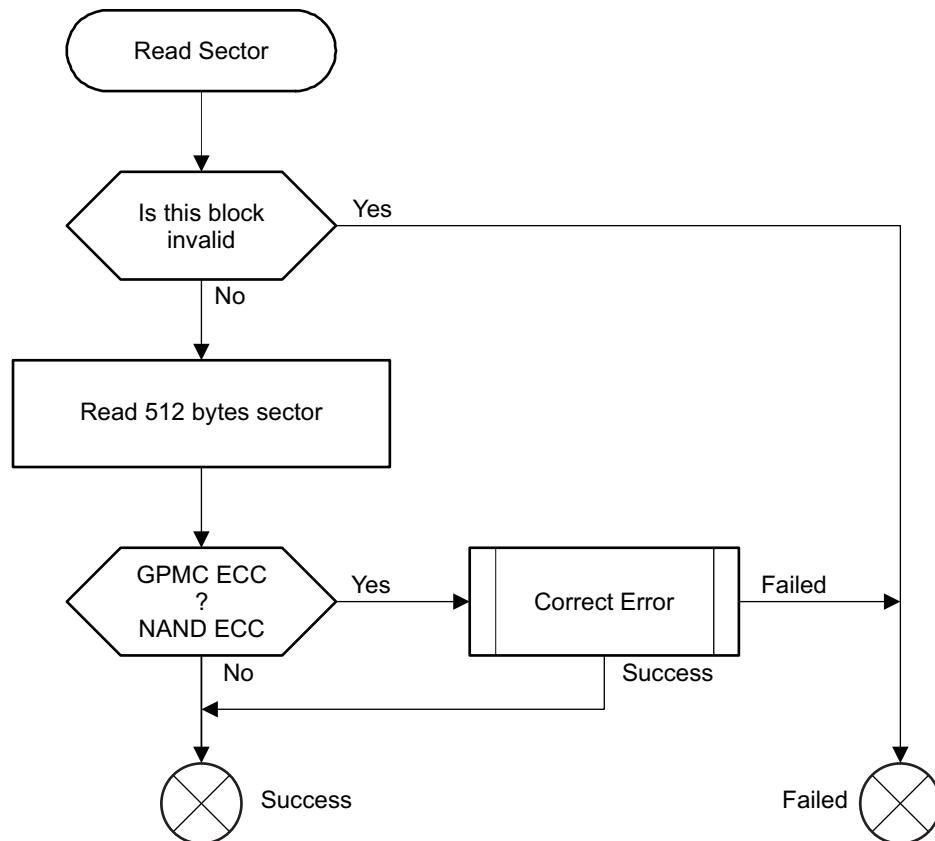
For first 4 blocks

#### 26.1.7.4.2.1 NAND Read Sector Procedure

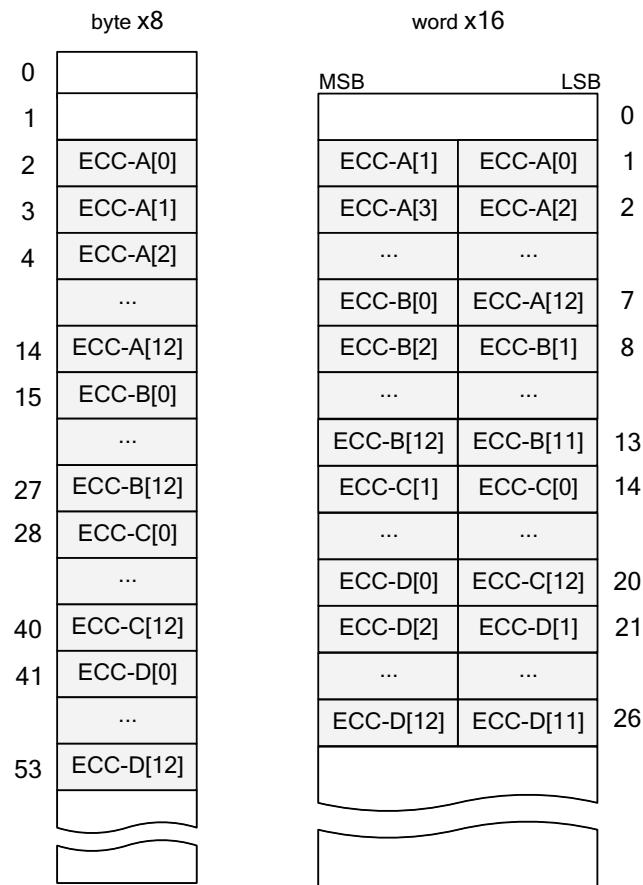
The ROM Code reads data from NAND devices in 512 bytes sectors. The read function fails in two cases:

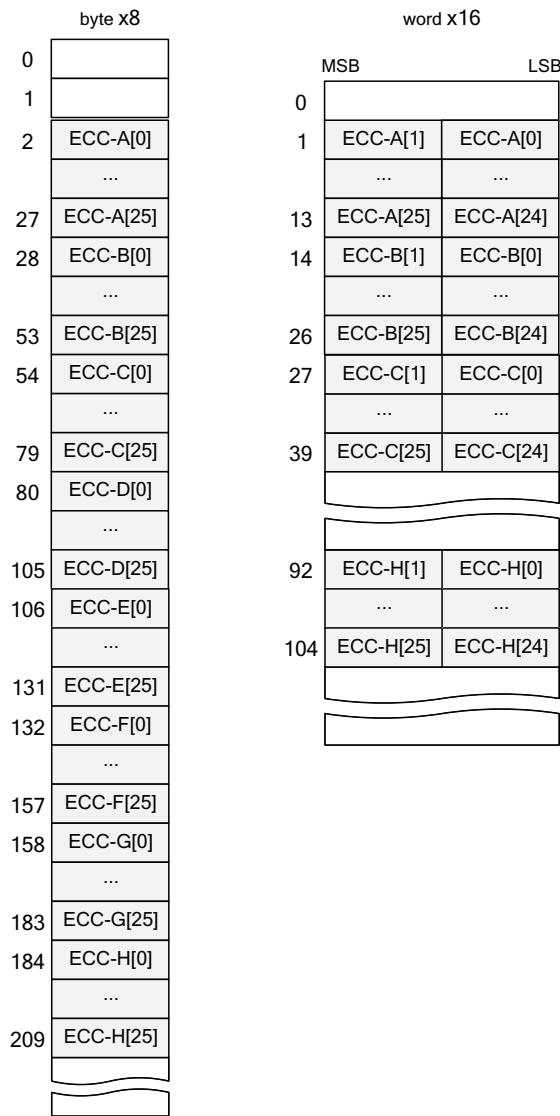
- The accessed sector is within a block marked as invalid
- The accessed sector contains an error which cannot be corrected with ECC

[Figure 26-14](#) shows the read sector routine for NAND devices. The ROM Code uses normal read (command 00h 30h) for reading NAND page data.

**Figure 26-14. NAND Read Sector Procedure**


Page data can contain errors due to memory alteration. The ROM Code uses an ECC correction algorithm to detect and possibly correct those errors. The ECC algorithm used is BCH with capability for correcting 8b or 16b errors per sector. The BCH data is automatically calculated by the GPMC on reading each 512 bytes sector. The computed ECC is compared against ECC stored in the spare area for the corresponding page. Depending on the page size, the amount of ECC data bytes stored in the corresponding spare area is different. [Figure 26-15](#) and [Figure 26-16](#) show the mapping of ECC data inside the spare area for respectively 2KB-page and 4KB- page devices. If both ECC data are equal then the Read Sector function returns the read 512 bytes sector without error. Otherwise the ROM Code tries to correct error(s) in the corresponding sector (this procedure is assisted by the ELM hardware) and returns the data if successful. If errors are uncorrectable, the function returns with FAIL.

**Figure 26-15. ECC Data Mapping for 2 KB Page and 8b BCH Encoding**


**Figure 26-16. ECC Data Mapping for 4 KB Page and 16b BCH Encoding**


#### 26.1.7.4.2.2 Pins Used

The list of device pins that are configured by the ROM in the case of NAND boot mode are as follows. Please note that all the pins might not be driven at boot time.

---

**NOTE:** Caution must be taken when using an 8-bit NAND. The ROM initially configures all address and data signals (AD0-AD15) the GPMC uses when attempting to read configuration values from the NAND. If you use an 8-bit NAND, and connect AD15-AD8 to other functions (GPIOs, for example), there may be contention on these signals during the boot phase. AD15-AD8 are configured as outputs and will be driven by the ROM if NAND boot is selected. Ensure external circuits will not be in contention with these driven outputs.

---

**Table 26-18. Pins Used for NAND Boot**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| CS0         | GPMC_CS0           |
| ADVN_ALE    | GPMC_ADVN_ALE      |

**Table 26-18. Pins Used for NAND Boot (continued)**

| Signal name | Pin Used in Device   |
|-------------|----------------------|
| OEN_REN     | GPMC_OEN_REN         |
| BEON_CLE    | GPMC_BEN0_CLE        |
| WEN         | GPMC_WEN             |
| WAIT        | GPMC_WAIT0           |
| CLK         | GPMC_CLK             |
| AD0 - AD15  | GPMC_AD0 - GPMC_AD15 |

#### 26.1.7.4.2.3 SYSBOOT Pins

SYSBOOT[11:10] has a special meaning when NAND boot is selected. SYSBOOT[11:10] must be 0.

#### 26.1.7.5 MMC / SD Cards

##### 26.1.7.5.1 Overview

The ROM Code supports booting from MMC / SD cards in the following conditions:

- MMC/SD Cards compliant to the Multimedia Card System Specification and Secure Digital I/O Card Specification of low and high capacities.
- MMC/SD cards connected to MMC0 or MMC1.
- Support for 3.3/1.8 V on MMC0 and MMC1.
- Initial 1-bit MMC Mode, optional 4-bit mode, if device supports it.
- Clock Frequency: identification mode: 400 KHz; data transfer mode up to 10 MHz.
- Only one card connected to the bus.
- Raw mode, image data read directly from sectors in the user area.
- File system mode (FAT12/16/32 supported with or without Master Boot Record), image data is read from a booting file.

##### 26.1.7.5.2 System Interconnection

Each interface has booting restrictions on which type of memory it supports: -

- MMC0 supports booting from the MMC/SD card cage and also supports booting from eMMC/eSD/managed NAND memory devices with less than 4GB capacity.
- MMC1 supports booting from eMMC/eSD/managed NAND memory device with 4GB capacity or greater.

The restriction is a result of many eMMC devices not being compliant with the eMMC v4.41 specification. If you have the need to boot from two different card cages, many MMC/SD cards will boot from MMC1, but for maximum compatibility only MMC0 should be used to boot from the card cage. Similarly for maximum compatibility, booting from eMMC/eSD/managed NAND should only be performed on MMC1.

Note that the above restrictions only apply to booting from each port. Drivers can be written for either port to support any desired interface.

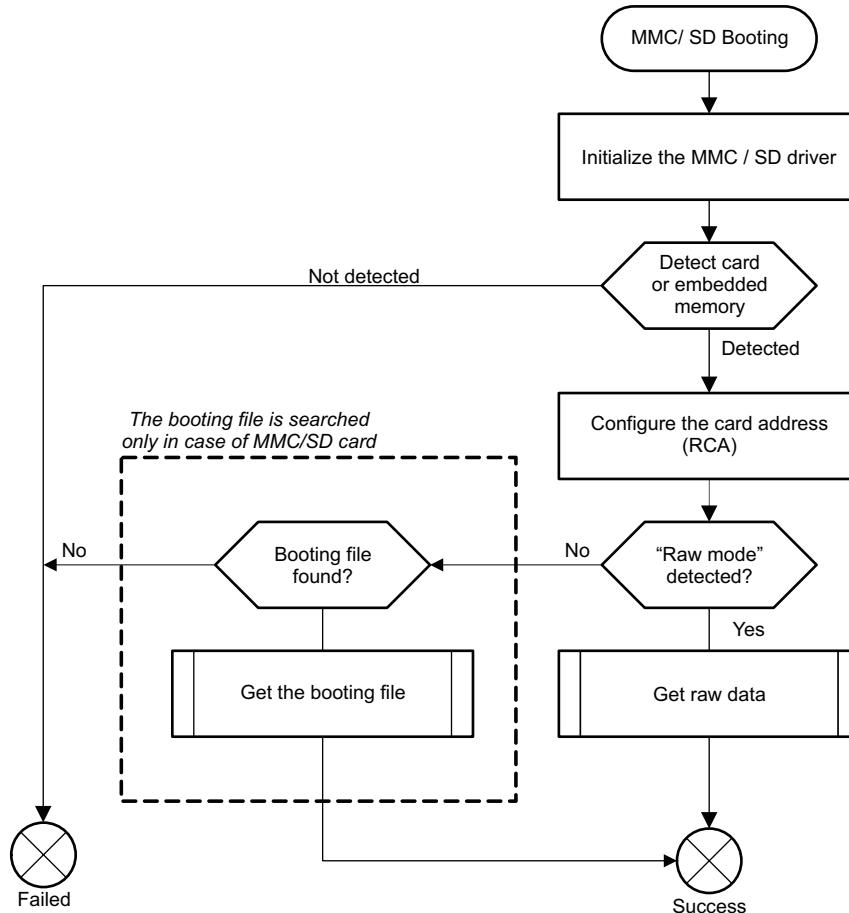
##### Note:

- The ROM Code does not handle the card detection feature on card cage.
- If MMC1 is used the GPMC interface is not usable, due to pin muxing options.
- MMC1 supports sector mode without querying the card.

### 26.1.7.5.3 Booting Procedure

The high level flowchart of the eMMC / eSD and MMC/SD booting procedure is depicted in [Figure 26-17](#). The booting file is searched only in case of booting from a card. eMMC/eSD embedded memories only support raw mode.

**Figure 26-17. MMC/SD Booting**



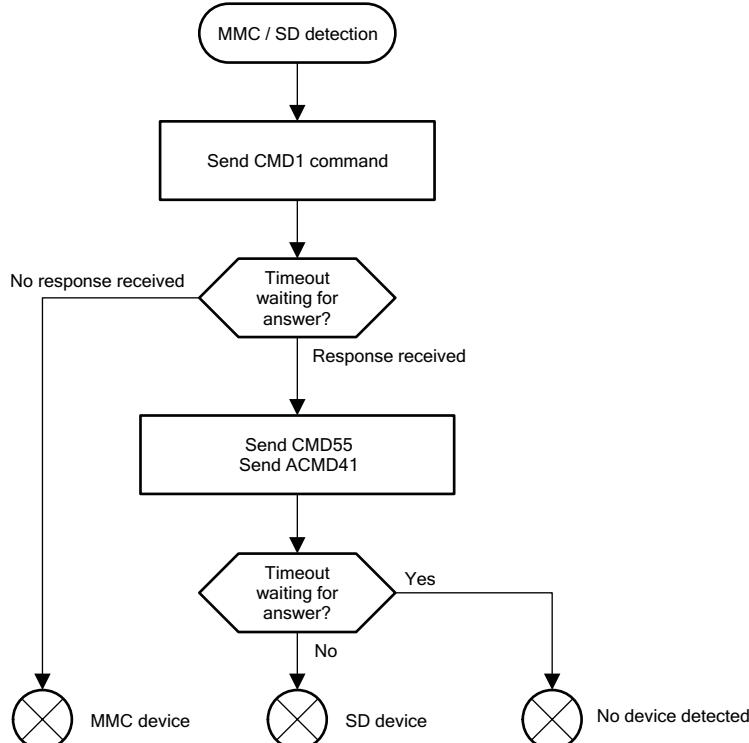
### 26.1.7.5.4 Initialization and Detection

The ROM Code initializes the memory device or card connected on MMC interface using the standard High-Voltage range (3.3 V). If neither memory device nor card is detected then the ROM Code carries on to the next booting device. The standard identification process and **Relative Card Address (RCA)** assignment are used. However, the ROM Code assumes that only one memory or card is present on the bus. This first sequence is done using the CMD signal which is common to SD and MMC devices.

MMC and SD standards detail this phase as **initialization phase**. Both standards differ in the first commands involved: **CMD1** and **ACMD41**. The ROM Code uses this difference in command set to discriminate between MMC and SD devices: CMD1 is supported only by the MMC standard whereas ACMD41 is only supported by SD standard. The ROM Code first sends a CMD1 to the device and gets a response only if an MMC device is connected. If no response is received then ACMD41 (ACMD41 is made out of CMD55 and ACMD41) is sent and a response is expected from an SD device. If no response is received then it is assumed that no device is connected and the ROM Code exits the MMC/SD Booting procedure with FAIL. This is detection procedure shown in [Figure 26-18](#).

Another point to note is the difference in the arguments to CMD1 between MMC0 and MMC1. At first the ROM queries the card with CMD1, ARG = 0, to get the OCR from the card. On MMC0, the response from the card is simply reflected back to the card as the argument for all subsequent CMD1, till the card is in the READY state. On MMC1, Bit30 of the response received from the card is set to 1 by the ROM, and this modified value is used as the argument for subsequent CMD1. This is done to indicate to the card that the ROM supports sector addressing. This mode might not be compatible with older (older than v4.4) versions of cards.

**Figure 26-18. MMC/SD Detection Procedure**



As previously mentioned the contents of an MMC/SD card may be formatted as raw binary or within a FAT filesystem. eMMC / eSD devices only support raw mode. The ROM Code reads out raw sectors from image or the booting file within the file system and boots from it.

#### 26.1.7.5.5 MMC/SD Read Sector Procedure in Raw Mode

In raw mode the booting image can be located at one of the four consecutive locations in the main area: offset 0x0 / 0x20000 (128 KB) / 0x40000 (256 KB) / 0x60000 (384 KB). For this reason, a booting image shall not exceed 128KB in size. However it is possible to flash a device with an image greater than 128KB starting at one of the aforementioned locations. Therefore the ROM Code does not check the image size. The only drawback is that the image will cross the subsequent image boundary.

The raw mode is detected by reading sectors #0, #256, #512, #768. The content of these sectors is then verified for presence of a TOC structure as described in [Section 26.1.9](#). In the case of a **GP Device**, a Configuration Header (CH) **must** be located in the first sector followed by a **GP header**. The CH might be void (only containing a CHSETTINGS item for which the Valid field is zero).

##### 26.1.7.5.5.1 Configuration Header

For Raw Mode, the Configuration Header should be formatted as follows:

**Table 26-19. Configuration Header TOC Item**

| Offset | Field        | Size (Bytes) | Value                                  | Description   |
|--------|--------------|--------------|--|---|
| 0x0000 | Start        | 4            | 0x000000a0                             | Offset from the start address of TOC to the actual address of a section.  |
| 0x0004 | Size         | 4            | 0x00000050                             | Size of a section   |
| 0x0008 | Reserved     | 4            | 0x00000000                             | Unused  |
| 0x000C | Reserved     | 4            |  | Unused  |
| 0x0010 | Reserved     | 4            |  | Unused  |
| 0x0014 | Filename     | 12           | 0x45534843<br>0x4e495454<br>0x00005347 | 12-character name of a section, including the zero (\0) terminator (these are the ASCII characters for 'CHSETTINGS'). |
| 0x0020 | Closing item | 4            | 0xFFFFFFFF                             | 32 bytes of 0xFF to signify the end of the TOC item.  |
| 0x0024 | Closing item | 4            |  |   |
| 0x0028 | Closing item | 4            |  |   |
| 0x002C | Closing item | 4            |  |   |
| 0x0030 | Closing item | 4            |  |   |
| 0x0034 | Closing item | 4            |  |   |
| 0x0038 | Closing item | 4            |  |   |
| 0x003C | Closing item | 4            |  |   |

**Table 26-20. Configuration Header Settings**

| Offset | Field       | Value      | Description   |
|--------|-------------|------------|---|
| 0000h  | Section key | 0xC0C0C0C1 | Key used for section verification                                   |
| 0004h  | Valid       | 0x00       | Enables or disables the section.<br>00h: Disable.<br>Other: Enable. |
| 0005h  | Version     | 0x001      | Configuration header version.<br>Others: For future use             |
| 0006h  | Reserved    | 0x00       | Set the reset of the header (offset 0x6 to 0xe0) to 0x00.           |
| ...    |             |            |   |
| 0xe0   |             |            |   |

#### 26.1.7.5.6 MMC/SD Read Sector Procedure in FAT Mode

MMC/SD Cards may hold a FAT file system which ROM Code is able to read and process. The image used by the booting procedure is taken from a specific booting file named "MLO". This file has to be located in the root directory on an active primary partition of type FAT12/16 or FAT32.

An MMC/SD card can be configured either as floppy-like or hard-drive-like.

- When acting as floppy-like, the content of the card is a single file system without any Master Boot Record (MBR) holding a partition table
- When acting as hard-drive-like, an MBR is present in the first sector of the card. This MBR holds a table of partitions, one of which must be FAT12/16/32, primary and active.

The card should always hold an MBR except for MMC cards using floppy-like file system (please refer to the CSD internal Register fields FILE\_FORMAT\_GRP and FILE\_FORMAT in the MultiMedia Card System Specification). However, depending on the used operating system the MMC/SD card will be formatted either with partition(s) (using an MBR) or without. The ROM Code supports both types; this is described in the following section.

The ROM Code retrieves a map of the booting file from the FAT table. The booting file map is a collection of all FAT table entries related to the booting file (a FAT entry points to a cluster holding part of the file). The booting procedure uses this map to access any 512 byte sector within the booting file without involving ROM Code FAT module.

The sector read procedure utilizes standard MMC/SD raw data read function. The sector address is generated based on the booting memory file map collected during the initialization. Hence the ROM Code can address sectors freely within the booting file space.

#### 26.1.7.5.7 FAT File system

This paragraph describes functions which are used by the ROM Code, it is not intended to fully describe the Master Boot Record and the FAT file system:

- How to recognize if a sector is the 1<sup>st</sup> sector of an MBR
- How to recognize if a sector is the 1<sup>st</sup> sector of a FAT12/16/32
- How to find the 1<sup>st</sup> cluster of the booting file
- How to buffer the booting file FAT entries.

Some memory devices which support file systems can be formatted with or without MBR, therefore the first task of the ROM Code is to detect whether or not the device is holding an MBR in the first sector.

If this is the case, an active FAT12/16/32 partition is searched in all 4 MBR partition entries, based on the Type field. If the MBR entries are not valid or if no useable partition is found then the ROM Code returns to the Booting procedure with FAIL. The Extended partitions are not checked, the booting file must reside in a primary partition.

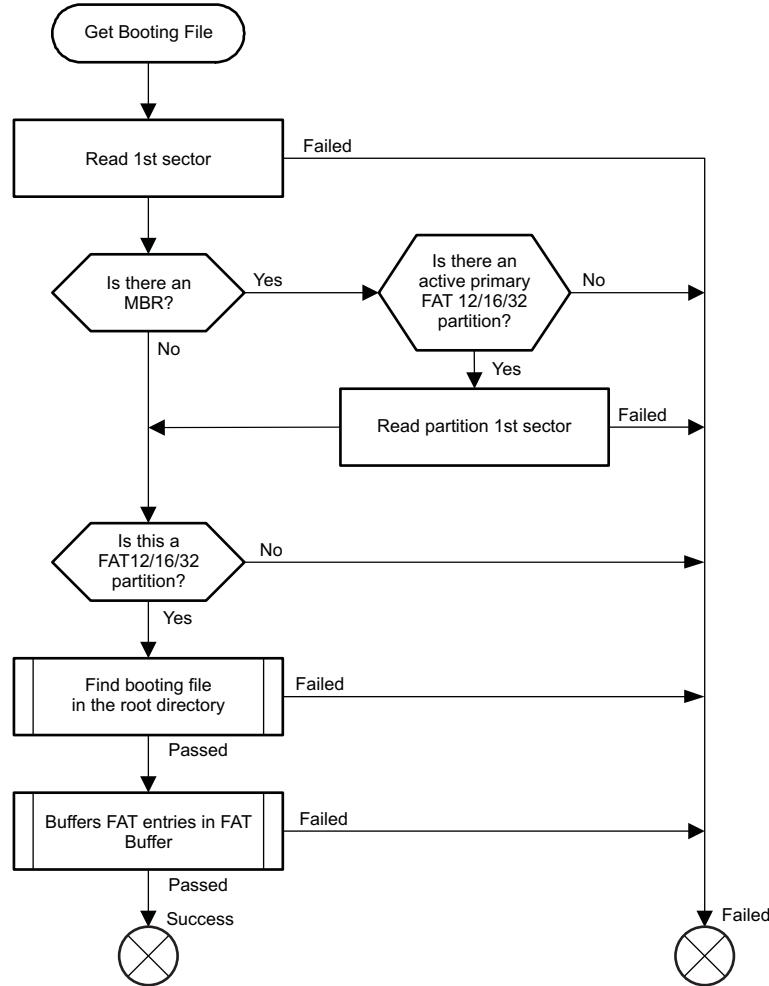
If a partition is found then its first sector is read and used further on. If no MBR is present (in case of a floppy-like system), the first sector of the device is read and used further on.

The read sector is checked to be a valid FAT12/16 or FAT32 partition. If this fails, in case another partition type is used (i.e. Linux FS or any other) or if the partition is not valid, the ROM Code returns with FAIL.

Otherwise, the Root Directory entries are searched for a file named depending on the booting device. The Long File Names (LFN) format is not used and **only File Names in 8.3 Format are searched for**. If no valid file is found, the ROM Code returns with FAIL.

Once the file has been found, the ROM Code reads the File Allocation Table (FAT) and buffers the singly-linked chain of clusters in a FAT Buffer which will be used by the Booting Procedure to access the file directly sector by sector. For FAT12/16 and for FAT32 (valid if a specific flag has been set in the FAT32 Boot Sector), there exist multiples copies of the FAT (ROM Code supports only 2 copies). When buffering FAT entries, the 2 FATs are compared. If they are not the same, only entries from the last FAT are used. The FAT Buffer holds sector numbers and not cluster numbers. The ROM Code converts each cluster entry to one or several sector entries if applicable.

The whole process is described in [Figure 26-19](#). Every part related to MBR or FAT12/16/32 is described in the next paragraph.

**Figure 26-19. MMC/SD Booting, Get Booting File**


#### 26.1.7.5.7.1 Master Boot Record (MBR)

The Master Boot Record is the 1<sup>st</sup> sector of a memory device. It is made out of some executable code and 4 partition entries. The aim of such a structure is to be able to divide the hard disk in partitions mostly used to boot different systems (i.e. Microsoft Windows™, Linux, ...). Its structure is described in [Table 26-21](#) and [Table 26-22](#). The valid partition types searched by the ROM Code are described in [Table 26-23](#).

**Table 26-21. Master Boot Record Structure**

| Offset | Length[bytes] | Entry Description     | Value                              |
|--------|---------------|-----------------------|------------------------------------|
| 0000h  | 446           | Optional Code         |                                    |
| 01BEh  | 16            | Partition Table Entry | (see <a href="#">Table 26-22</a> ) |
| 01CEh  | 16            | Partition Table Entry | (see <a href="#">Table 26-22</a> ) |
| 01DEh  | 16            | Partition Table Entry | (see <a href="#">Table 26-22</a> ) |
| 01EEh  | 16            | Partition Table Entry | (see <a href="#">Table 26-22</a> ) |
| 01FEh  | 2             | Signature             | AA55h                              |

**Table 26-22. Partition Entry**

| Offset | Length[bytes] | Entry Description  | Value   |
|--------|---------------|--|---|
| 0000h  | 1             | Partition State  | 00h: Inactive 80h: Active   |
| 0001h  | 1             | Partition Start Head                                     | H <sub>s</sub>  |
| 0002h  | 2             | Partition Start Cylinder and Sector                      | C <sub>s</sub> [7:0]-C <sub>s</sub> [9:8]-S <sub>s</sub> [5:0]  |
| 0004h  | 1             | Partition Type   | See Table 26-23 for partial partition types   |
| 0005h  | 16            | Partition End Head                                       | H <sub>e</sub>  |
| 0006h  | 2             | Partition End Cylinder and Sector                        | C <sub>e</sub> [7:0]-C <sub>e</sub> [9:8]-S <sub>e</sub> [5:0]  |
| 0008h  | 4             | First sector position relative to the beginning of media | LBA <sub>s</sub> =C <sub>s</sub> .H.S+ H <sub>s</sub> .S+ S <sub>s</sub> -1   |
| 000Ch  | 4             | Number of sectors in partition                           | LBA <sub>e</sub> =C <sub>e</sub> .H.S+ H <sub>e</sub> .S+ S <sub>e</sub> -1Nbs= LBA <sub>s</sub> -LBA <sub>e</sub> +1 |

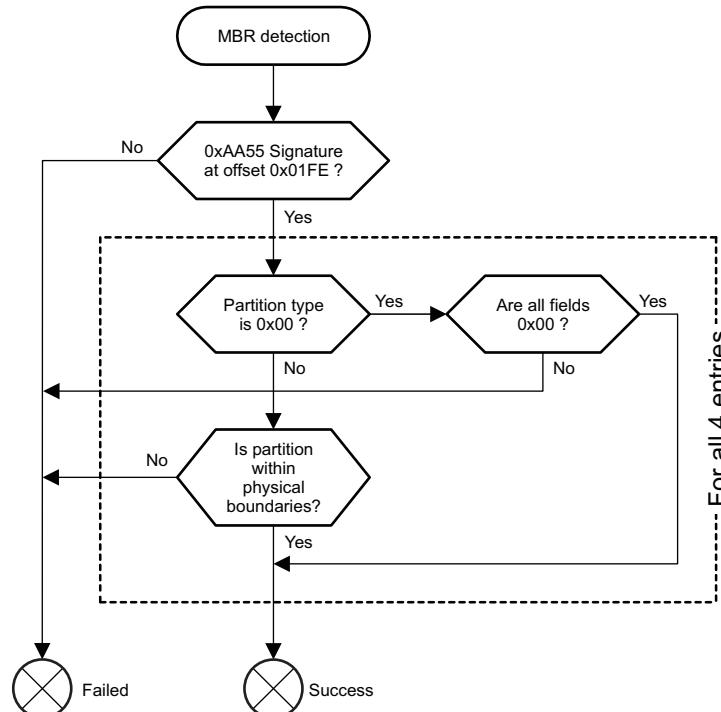
**Table 26-23. Partition Types**

| Partition Type | Description |
|----------------|-------------|
| 01h            | FAT12       |
| 04h, 06h, 0Eh  | FAT16       |
| 0Bh, 0Ch, 0Fh  | FAT32       |

The way the ROM Code detects whether a sector is the 1<sup>st</sup> sector of an MBR or not is described in [Figure 26-20](#).

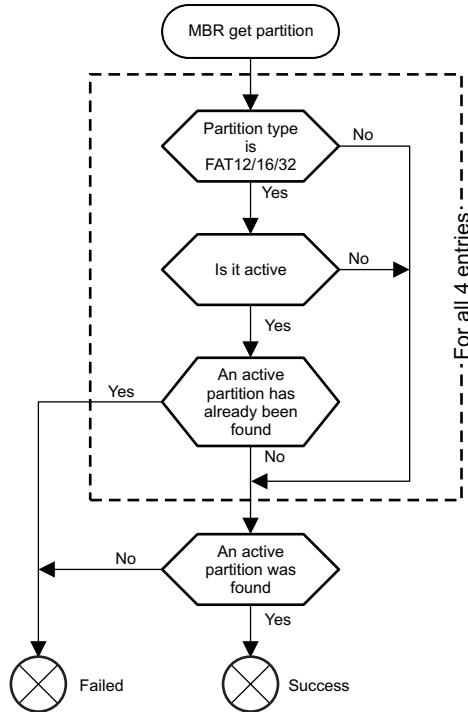
The ROM Code first checks if the signature is present. Each partition entry is checked:

- If its type is set to 00h then all fields in the entry must be 00h
- The partition is checked to be within physical boundaries, i.e. the partition is located inside and it's size fits the total physical sectors.

**Figure 26-20. MBR Detection Procedure**

Once identified, the ROM Code gets the partition using the procedure described in [Figure 26-21](#). The partition type is checked to be FAT12/16 or FAT32. Its state must be 00h or 80h (if there are more than one active partition the test fails). The ROM Code returns with FAIL if no active primary FAT12/16/32 could be found.

**Figure 26-21. MBR, Get Partition**



#### 26.1.7.5.7.2 FAT12/16/32 Boot Sector

The FAT file system is made out of several parts:

- Boot Sector which holds the BIOS Parameter Block (BPB)
- File Allocation Table (FAT) which describes the use of each cluster of the partition
- Data Area which holds the Files, Directories and Root Directory (for FAT12/16, the Root Directory has a specific fixed location).

The Boot Sector is described in [Table 26-24](#).

**Note:** In the following description, all the fields whose names start with BPB\_ are part of the BPB. All the fields whose names start with BS\_ are part of the Boot Sector and not really part of the BPB (not mandatory), they are not used at all by the ROM Code.

**Table 26-24. FAT Boot Sector**

| Offset | Length[bytes] | Name           | Description  |
|--------|---------------|----------------|--|
| 0000h  | 3             | BS_jmpBoot     | Jump Instruction to Boot Code (not used)   |
| 0003h  | 8             | BS_OEMName     | Name of the System which created the partition   |
| 000Bh  | 2             | BPB_BytsPerSec | Counts of Bytes per sector (usually 512)   |
| 000Dh  | 1             | BPB_SecPerClus | Number of sectors per allocation unit  |
| 000Eh  | 2             | BPB_RsvdSecCnt | Number of reserved sectors for the Boot SectorFor FAT12/16 is 1, for FAT32, usually 32 |
| 0010h  | 1             | BPB_NumFATs    | Number of copies of FAT, usually 2   |

**Table 26-24. FAT Boot Sector (continued)**

| Offset   | Length[bytes] | Name           | Description  |
|----------|---------------|----------------|--|
| 0011h    | 2             | BPB_RootEntCnt | For FAT12/16, number of 32 bytes entries in the Root Directory (multiple of BPB_BytsPerSec/32) For FAT32 this value is 0                             |
| 0013h    | 2             | BPB_TotSec16   | Total Count of sectors on the volume. If the size is bigger than 10000h or for FAT32, this field is 0 and BPB_TotSec32 holds the value               |
| 0015h    | 1             | BPB_Media      | Media Type, usually F8h: fixed, non-removable  |
| 0016h    | 2             | BPB_FATSz16    | For FAT12/16, size in sectors of one FAT For FAT32, holds 0  |
| 0018h    | 2             | BPB_SecPerTrk  | Number of sectors per track, 63 for SD/MMC   |
| 001Ah    | 2             | BPB_NumHeads   | Number of heads, 255 for SD/MMC  |
| 001Ch    | 4             | BPB_HiddSec    | Number of sectors preceding the partition  |
| 0020h    | 4             | BPB_TotSec32   | Total Count of sectors on the volume. If the size is smaller than 10000h (for FAT12/16), this field is 0 and BPB_TotSec16 is valid                   |
| FAT12/16 | 0024h         | BS_DrvNum      | Drive Number   |
|          | 0025h         | BS_Reserved1   | 00h  |
|          | 0026h         | BS_BootSig     | Extended Boot Signature 29h. Indicates that the following 3 fields are present   |
|          | 0027h         | BS_VolID       | Volume Serial Number   |
|          | 002Bh         | BS_VolLab      | Volume Label   |
|          | 0036h         | BS_FilSysType  | File system Type: "FAT12", "FAT16", "FAT32". Note: This field is not mandatory (i.e BS_) therefore it cannot be used to identify the partition type. |

**Table 26-24. FAT Boot Sector (continued)**

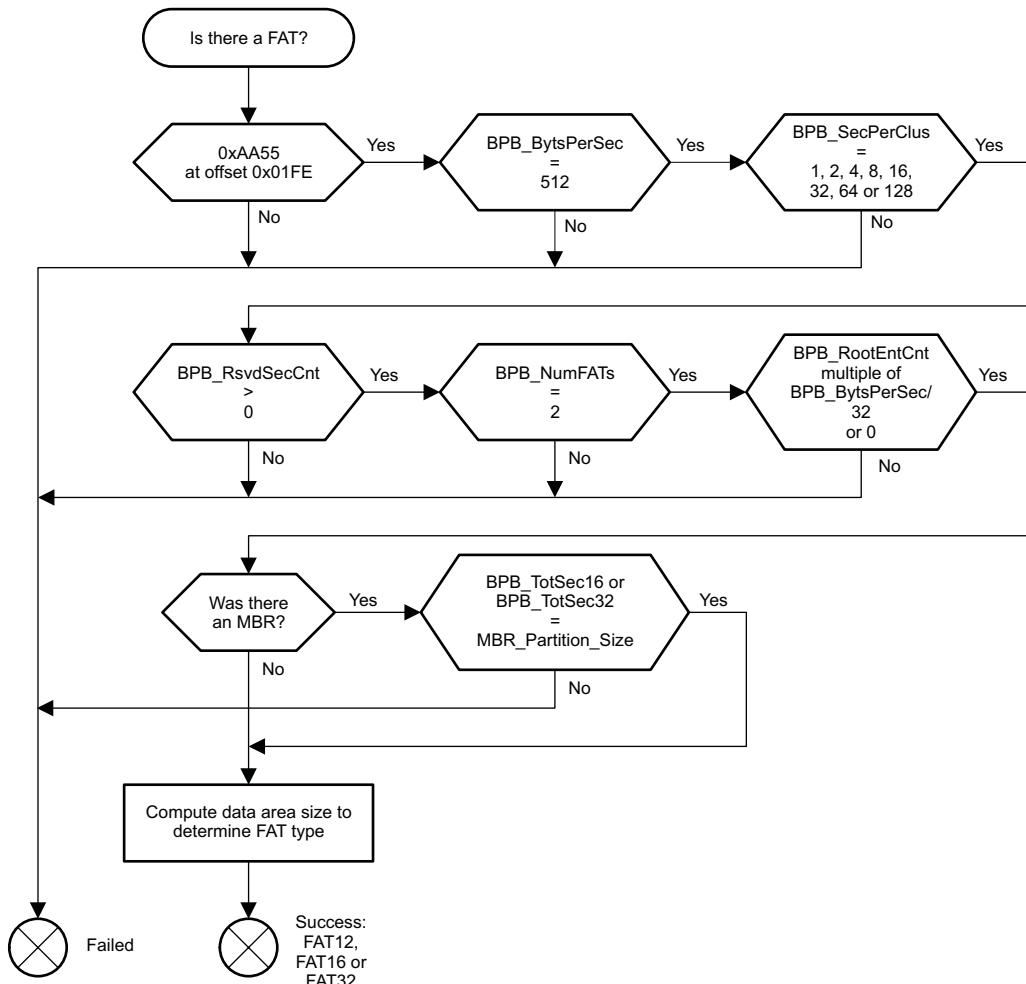
| Offset | Length[bytes] | Name          | Description   |  |
|--------|---------------|---------------|---------------|--|
| FAT32  | 0024h         | 4             | BPB_FATSz32   | Size in sectors of one FAT. Field BPB_FATSz16 must be 0  |
|        | 0028h         | 2             | BPB_ExtFlags  | FAT Flags:   |
|        |               |               | [7]:          | 0=FAT is mirrored;<br>1=Only one FAT is used   |
|        |               |               | [3:0]:        | Number of used FAT if no mirroring used  |
|        | 002Ah         | 2             | BPB_FSVer     | File system Version Number   |
|        | 002Ch         | 4             | BPB_RootClus  | First Cluster number of the Root Directory   |
|        | 0030h         | 2             | BPB_FSIInfo   | Sector number of FSINFO Structure in the reserved-area, usually 1  |
|        | 0032h         | 2             | BPB_BkBootSec | If non-zero, indicates the sector number in the reserved-area of a copy of the Boot Sector   |
|        | 0034h         | 12            | BPB_Reserved  | Reserved, set to 00h   |
|        | 0040h         | 1             | BS_DrvNum     | Drive Number   |
|        | 0041h         | 1             | BS_Reserved1  | 00h  |
|        | 0042h         | 1             | BS_BootSig    | Extended Boot Signature 29h. Indicates that the following 3 fields are present   |
|        | 0043h         | 4             | BS_VolID      | Volume Serial Number   |
|        | 0047h         | 11            | BS_VolLab     | Volume Label   |
|        | 0052h         | 8             | BS_FilSysType | File system Type: "FAT12", "FAT16", "FAT32". Note: This field is not mandatory (i.e BS_) therefore it cannot be used to identify the partition type. |
| 01FEh  | 2             | BPB_Signature |               | AA55h  |

To check whether or not a sector holds a valid FAT12/16/32 partition, only fields starting with BPB can be checked as they are mandatory. The fields starting from offset 0024h to 01FDh cannot be used for the check as they will differ if using FAT12/16 or FAT32. The procedure is described in Figure 30. First the ROM Code checks if the *BPB\_Signature* is equal to AA55h. Then it checks some fields which must have some specific values (*BPB\_BytsPerSec*, *BPB\_SecPerClus*, *BPB\_RsvdSecCnt*, *BPB\_NumFATs*, *BPB\_RootEntCnt*). If the geometry of the device is known (valid CHS for device size < 4Gbytes) then it is compared against *BPB\_SecPerTrk* and *BPB\_NumHeads* fields. If an MBR was found before, the partition size is also checked:

The field *BPB\_ToSec16* is used if the total number of sectors is below 65518 (in this case *BPB\_TotSec32*=0), otherwise *BPB\_TotSec32* is used (*BPB\_TotSec16*=0). The partition sector offset is also checked: *BPB\_HiddSec* = *MBR\_Partition\_Offset* (if this value is not 0 as some operating systems do not update this field correctly). The last step is to decide which type of FAT file system it is. The ROM Code computes the number of clusters in the Data Area part of the partition:

Where *Nb clusters* is given by the size of the Data Area:

Figure 26-22. FAT Detection Procedure



#### 26.1.7.5.7.3 FAT12/16/32 Root Directory

The next task for the ROM Code is to find the booting file named “MLO” inside the Root Directory of the FAT12/16/32 file system. The file is not searched in any other location.

For a FAT12/16 file system, the Root Directory has a fixed location which is cluster 0. For a FAT32 file system, its cluster location is given by *BPB\_RootClus*. The general formulae to find the sector number (relative to device sector 0, not partition sector 0) of a cluster is given by:

$$\text{Cluster}_{\text{sector}} = \text{BPB_RsvdSecCnt} + \text{BPB_NumFATs} \times \text{BPB_FatSz} + \text{Cluster} \times \text{BPB_SecPerClus}$$

Note: *BPB\_FatSz* is *BPB\_FatSz16* for FAT12/16 or *BPB\_FatSz32* for FAT32

Note: the *BPB\_HiddSec* field can contain 0 even though the FAT file system is located somewhere other than on sector 0 (floppy-like). The ROM Code actually uses the partition offset taken from the MBR instead of this field which can be wrong. If no MBR was found (floppy-like) the value 0 is used.

Each entry in the Root Directory is 32 bytes long and hold information about the file, i.e. filename, date of creation, rights, cluster location etc. This is described in [Table 26-25](#).

The ROM Code checks each entry in the Root Directory until either the booting file is found or the entry is empty (first byte is 00h) or when the end of the Root Directory has been reached. Entries with ATTR\_LONG\_NAME attribute (LFN) and with first byte at E5h (erased file) are ignored. When found, the first cluster offset of the file is read from the DIR\_FstClusHi/DIR\_FstClusLo fields.

There is a slight difference between FAT12/16 and FAT32 when handling the Root Directory. On FAT12/16, this directory has a fixed location (see above) and length fixed by *BPB\_RootEntCnt* which is the total number of 32 bytes entries. Handling this directory is therefore straight forward. On FAT32, the Root Directory is like a standard file, the File Allocation Table (FAT) has to be used in order to retrieve each sector of the Directory. The way the FAT is handled is described in the next paragraph.

**Table 26-25. FAT Directory Entry**

| Offset | Length[bytes] | Name             | Description                                    |   |
|--------|---------------|------------------|--|---|
| 0000h  | 11            | DIR_Name         | Short Name (8+3)                               |   |
| 000Bh  | 1             | DIR_Attr         | File Attributes:                               |   |
|        |               |                  | ATTR_READ_ONLY                                 | 01h   |
|        |               |                  | ATTR_HIDDEN                                    | 02h   |
|        |               |                  | ATTR_SYSTEM                                    | 04h   |
|        |               |                  | ATTR_VOLUME_ID                                 | 08h   |
|        |               |                  | ATTR_DIRECTORY                                 | 10h   |
|        |               |                  | ATTR_ARCHIVE                                   | 20h   |
|        |               |                  | ATTR_LONG_NAME                                 | ATTR_READ_ONLY   ATTR_HIDDEN   ATTR_SYSTEM   ATTR_VOLUME_ID |
| 000Ch  | 1             | DIR_NTRes        | Reserved, set to 00h                           |   |
| 000Dh  | 1             | DIR_CrtTimeTenth | Millisecond stamp at file creation             |   |
| 000Eh  | 2             | DIR_CrtTime      | Time file was created                          |   |
| 0010h  | 2             | DIR_CrtDate      | Date file was created                          |   |
| 0012h  | 2             | DIR_LstAccDate   | Last Access date                               |   |
| 0014h  | 2             | DIR_FstClusHi    | High word of this entry's first cluster number |   |
| 0016h  | 2             | DIR_WrtTime      | Time of last write                             |   |
| 0018h  | 2             | DIR_WrtDate      | Date of last write                             |   |
| 001Ah  | 2             | DIR_FstClusLo    | Low word of this entry's first cluster number  |   |
| 001Ch  | 4             | DIR_FileSize     | File size in bytes                             |   |

#### 26.1.7.5.7.4 FAT12/16/32 File Allocation Table

The ROM Code needs to read the FAT in order to retrieve sectors either for the booting file or for the Root Directory (in case the file system is FAT32).

There can be multiple copies of the FAT inside the file system (ROM Code supports only 2) located after the Boot Sector:

$$FATn_{sector} = BPB_HiddSec + BPB_RsvdSecCnt + BPB_FatSz \times n$$

Its size is given by *BPB\_FATSz16* or *BPB\_FATSz32*. The ROM Code checks each copy of the FAT if identical. In case the values are different, the ROM Code uses the value from the last FAT copy.

With FAT32 file system, the copy system can be disabled according to a flag located in *BPB\_ExtFlags[7]*. If this flag is set then FAT *BPB\_ExtFlags[3:0]* is used. In this case no verification is made by the ROM Code with other copies of FAT.

The FAT is a simple array of values each referring to a cluster located in the Data Area. One entry of the array is 12, 16 or 32 bits depending on the file system in use.

The value inside an entry defines whether the cluster is being used or not and if another cluster has to be taken into account. This creates a singly-linked chain of clusters defining the file. The meaning of an entry is described in [Table 26-26](#).

**Note:** For compatibility reasons, clusters 0 and 1 are not used for files and those entries must contain FF8h and FFFh (for FAT12); FFF8h and FFFFh (for FAT16); ?FFFFFF8h and ?FFFFFFFh (for FAT32).

**Table 26-26. FAT Entry Description**

| FAT12     | FAT16       | FAT32               | Description                                |
|-----------|-------------|---------------------|--|
| 000h      | 0000h       | ?0000000h           | Free Cluster                               |
| 001h      | 0001h       | ?00000001h          | Reserved Cluster                           |
| 002h-FEFh | 0002h-FFEFh | 00000002h-?FFFFFFEh | Used Cluster; value points to next cluster |
| FF0h-FF6h | FFF0h-FFF6h | ?FFFFFF0h-?FFFFFF6h | Reserved values                            |
| FF7h      | FFF7h       | ?FFFFFF7h           | Bad Cluster                                |
| FF8h-FFFh | FFF8h-FFFFh | ?FFFFFF8h-?FFFFFFFh | Last Cluster in File                       |

**Note:** FAT32 uses only bits [27:0], the upper 4 bits are usually 0 and should be left untouched.

When accessing the Root Directory for FAT32, the ROM Code just starts from the Root Directory Cluster entry and follows the linked chain to retrieve the clusters.

When the booting file has been found, the ROM Code buffers each FAT entry corresponding to the file in a sector way. This means each cluster is translated to one or several sectors depending on how many sectors are in a cluster (*BPB\_SecPerClus*). This buffer is used later on by the booting procedure to access the file.

#### 26.1.7.5.8 Pins Used

The list of device pins that are configured by the ROM in the case of MMC boot mode are as follows. Please note that all the pins might not be driven at boot time.

**Table 26-27. Pins Used for MMC0 Boot**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| clk         | mmc0_clk           |
| cmd         | mmc0_cmd           |
| dat0        | mmc0_dat0          |
| dat1        | mmc0_dat1          |
| dat2        | mmc0_dat2          |
| dat3        | mmc0_dat3          |

**Table 26-28. Pins Used for MMC1 Boot**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| clk         | gpmc_csn1          |
| cmd         | gpmc_csn2          |
| dat0        | gpmc_ad0           |
| dat1        | gpmc_ad1           |
| dat2        | gpmc_ad2           |
| dat3        | gpmc_ad3           |

#### 26.1.7.6 SPI

SPI EEPROMs or SPI flashes have an EEPROM or NOR flash backend and they connect to the device using the serial SPI protocol.

These devices typically operate in three stages: the command stage, the address stage and the data transfer stage. The command is usually an 8 bit value followed by the address (depending on the size of the device) followed by the data to be read or written.

Because of the need for fewer pins, these devices are comparatively inexpensive, easy for board layout, and are the devices of choice when cost, complexity and form factor are critical considerations.

#### 26.1.7.6.1 Features

- Supports 12-MHz clock (50% duty cycle).
- Supports only SPI Mode 3 (clock polarity = 1, clock phase = 1).
- Supports only 24-bit addressable EEPROMs.
- Supports only 4-pin SPI mode (CS, CLK, Serial Input, Serial Output).
- The boot devices must be connected to chip select 0 and must support the read command (03h).
- The boot image is copied into internal memory and then executed.

#### 26.1.7.6.2 Initialization and Detection

The ROM Code initializes the SPI controller, pin muxing and clocks for communicating with the SPI device. The controller is initialized in Mode 3 and the clock is setup to operate at 12 MHz. There is no specific device identification routine that is executed by the ROM code to identify whether a boot device is present or not. If no SPI device is present, the sector read will return only 0xFFFFFFFF and the SPI boot will be treated as failed.

#### 26.1.7.6.3 SPI Read Sector Procedure

The ROM Code reads SPI data from the boot device in 512 byte sectors. For each call to the SPI Read Sector routine, the SPI Read Command (0x03) is sent along with the 24 bit start address of the data to be read.

From the next iteration onwards, a dummy value is transmitted on the master out line and the data is received on the master in line. This needs to be done because SPI protocol always operated in full duplex mode. The dummy data transmitted by the ROM is the Read Command appended to the start address. The data from the boot device is received MSB first.

As the A8 is a little-endian processor, and SPI operates in a big-endian format, this means that while writing to the flash, care needs to be taken to write the image in a big-endian format. This way we can avoid doing the endian conversion at boot time, thus improving boot performance.

#### 26.1.7.6.4 Pins Used

The list of device pins that are configured by the ROM in the case of SPI boot mode are as follows. Please note that all the pins might not be driven at boot time.

**Table 26-29. Pins Used for SPI Boot**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| cs          | spi0_cs0           |
| miso        | spi0_d0            |
| mosi        | spi0_d1            |
| clk         | spi0_sclk          |

#### 26.1.7.7 Blocks and Sectors Search Summary

[Table 26-30](#) summarizes numbers of blocks and sectors which are searched during the memory booting from devices requiring image shadowing. NAND is organized with blocks, which are erasable units.

**Table 26-30. Blocks and Sectors Searched on Non-XIP Memories**

| Memory                                    | Maximum Number of Blocks Checked | Number of Sectors Searched                    |
|---|----------------------------------|---|
| NAND                                      | first 4                          | Number of sectors in one block <sup>(1)</sup> |
| SPI, eMMC/eSD and MMC/SD cards (raw mode) | first 4                          | 1   |

<sup>(1)</sup> Depends on NAND geometry

Regarding MMC/SD card booting in FAT mode, the file system area is searched for one file.

### 26.1.8 Peripheral Booting

#### 26.1.8.1 Overview

The ROM Code can boot from four different peripheral interfaces:

- EMAC: 1000/100/10 Mbps Ethernet, using standard TCP/IP network boot protocols BOOTP and TFTP
- USB: Full speed, client mode
- UART: 115.2Kbps, 8 bits, even parity, 1 stop bit, no flow control

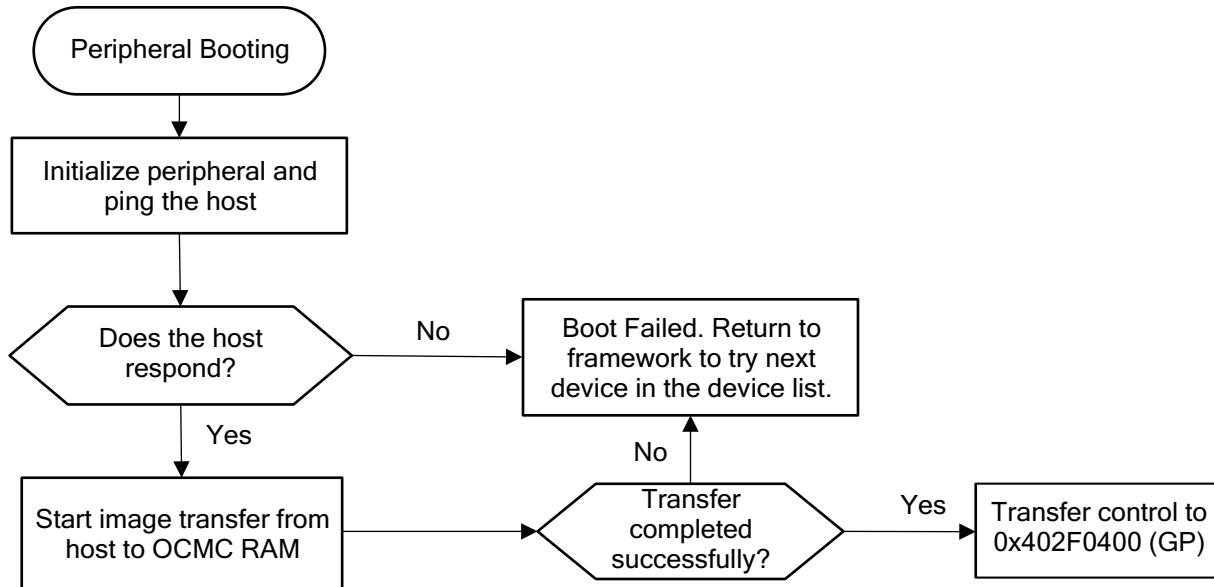
The purpose of booting from a **peripheral interface** is to download a **boot image** from an external host (typically a PC). This booting method is mostly used for programming flash memories connected to the device (e.g. in the case of initial flashing, firmware update or servicing).

#### 26.1.8.2 Boot Image Location and Size

The boot image is downloaded directly into internal RAM at the location 0x402F0000 on GP devices. The maximum size of downloaded image is 109 KB.

#### 26.1.8.3 Peripheral Boot Procedure Overview

Figure 26-23. Peripheral Booting Procedure



#### 26.1.8.4 EMAC Boot Procedure

##### 26.1.8.4.1 Device Initialization

- EMAC boot uses the CPGMAC port 1 of the device.
- Supports connection to external Ethernet PHY using the MII, RMII, RGMII and MDIO pins, based on SYSBOOT pin settings.
- Device uses EFUSE registers `mac_id0_lo` and `mac_id0_hi` in the control module for Ethernet MAC address of the device.
- Device detects if the PHY is alive on the MDIO interface and
  - Reads the STATUS register to check if Ethernet link is active
  - Reads the CONTROL register to detect the auto-negotiated mode of operation

- Is the mode full-duplex or half duplex
- Speed of operation, 1000/100/10 Mbps.
- Waits for five seconds for auto negotiation to complete before timing out.
- ROM expects an external 50-MHz reference clock requirement when using RMII.

#### 26.1.8.4.2 BOOTP (RFC 951)

The device then proceeds to obtain the IP and Boot information using BOOTP protocol. The device prepares and broadcasts the BOOTP message that has the following information:

- Device MAC address in “chaddr” field – to uniquely identify the device to the server.
- “vender-class-identifier” option number 60 (RFC 1497, RFC 1533). Servers could use this information to identify the device type. The value present is "**AM335x ROM v1.0**". (Note: Version may change in future devices.)
- “Client-identifier” option number 61 (RFC 1497, RFC 1533). This has the ASIC-ID structure which contains additional info for the device.

The device then expects a BOOTP response that provides the following information for the booting to proceed

- Device IP address from “yiaddr” field
- Subnetmask from extended option 1 (RFC 1497, RFC 1533)
- Gateway IP from extended option number 3 (RFC 1497, RFC 1533) or from “giaddr” field of BOOTP response.
- Boot image filename from “file” field
- TFTP server IP address from the “siaddr” field

Timeouts and retries:

- Exponentially increasing timeouts starting from 4s, doubling for each retry.
- 5 retries

#### 26.1.8.4.3 TFTP (RFC 1350)

After a successful BOOTP completion, the device initiates the TFTP download of the boot image into SRAM. The device has the capability to reach a TFTP server within the local subnet or outside, through the gateway.

Timeouts and retries:

- Timeout of 1s to receive a response for the READ request
- 5 retries for the READ request
- Retries are managed by server once data transfer starts (server re-sends a data packet if the ACK was not received within a timeout value)
- Device has a 60s timeout to complete the data transfer, to handle the scenario if the server dies in the middle of a data transfer

#### 26.1.8.4.4 Pins Used

The list of device pins that are configured by the ROM in the case of EMAC boot mode are as follows. Please note that all the pins might not be driven at boot time.

**Table 26-31. Pins Used for EMAC Boot in MII Mode**

| Signal Name    | Pin Used in Device | Pin Mux Mode |
|----------------|--------------------|--------------|
| gmii1_col      | MII1_COL           | 0            |
| gmii1_crs      | MII1_CRS           | 0            |
| gmii1_rxer     | MII1_RX_ER         | 0            |
| gmii1_txen     | MII1_TX_EN         | 0            |
| gmii1_rxdv     | MII1_RX_DV         | 0            |
| gmii1_txd[3:0] | MII1_TXD[3:0]      | 0            |
| gmii1_txclk    | MII1_TX_CLK        | 0            |
| gmii1_rxclk    | MII1_RX_CLK        | 0            |
| gmii1_rxd[3:0] | MII1_RXD[3:0]      | 0            |
| mdio_data      | MDIO               | 0            |
| mdio_clk       | MDC                | 0            |

**Table 26-32. Pins Used for EMAC Boot in RGMII Mode**

| Signal Name    | Pin Used in Device | Pin Mux Mode |
|----------------|--------------------|--------------|
| rgmii1_tctl    | MII1_TX_EN         | 2            |
| rgmii1_rctl    | MII1_RX_DV         | 2            |
| rgmii1_td[3:0] | MII1_TXD[3:0]      | 2            |
| rgmii1_tclk    | MII1_TX_CLK        | 2            |
| rgmii1_rclk    | MII1_RX_CLK        | 2            |
| rgmii1_rd[3:0] | MII1_RXD[3:0]      | 2            |
| mdio_data      | MDIO               | 0            |
| mdio_clk       | MDC                | 0            |

**Table 26-33. Pins Used for EMAC Boot in RMII Mode**

| Signal Name    | Pin Used in Device                               | Pin Mux Mode |
|----------------|--|--------------|
| rmii1_crs_dv   | MII1_CRS   | 1            |
| rmii1_rxer     | MII1_RX_ER                                       | 1            |
| rmii1_txen     | MII1_TX_EN                                       | 1            |
| rmii1_txd[1:0] | MII1_TXD[1:0]                                    | 1            |
| rmii1_rxd[1:0] | MII1_RXD[1:0]                                    | 1            |
| rmii1_refclk   | RMII1_REF_CLK (Driven by External 50-MHz Source) | 0            |
| mdio_data      | MDIO   | 0            |
| mdio_clk       | MDC  | 0            |

#### 26.1.8.4.5 SYSBOOT Pins

Some of the SYSBOOT pins have special meanings when EMAC boot is selected.

**Table 26-34. Ethernet PHY Mode Selection**

| SYSBOOT[7:6] | PHY Mode |
|--------------|----------|
| 00b          | MII      |
| 01b          | RMII     |

**Table 26-34. Ethernet PHY Mode Selection (continued)**

| SYSBOOT[7:6] | PHY Mode                     |
|--------------|------------------------------|
| 10b          | Reserved                     |
| 11b          | RGMII without internal delay |

### 26.1.8.5 UART Boot Procedure

#### 26.1.8.5.1 Device Initialization

- UART boot uses UART0.
- UART0 is configured to run at 115200 baud, 8-bits, no parity, 1 stop bit and no flow control.

#### 26.1.8.5.2 Boot Image Download

- UART boot uses x-modem client protocol to receive the boot image.
- Utilities like hyperterm, teraterm, minicom can be used on the PC side to download the boot image to the board
- With x-modem packet size of 1K throughout is roughly about 4KBytes/Sec.
- The ROM code will ping the host 10 times in 3s to start x-modem transfer. If host does not respond, UART boot will timeout.
- Once the transfer has started, if the host does not send any packet for 3s, UART boot will time out
- If the delay between two consecutive bytes of the same packet is more than 2ms, the host is requested to re-transmit the entire packet again
- Error checking using the CRC-16 support in x-modem. If an error is detected, the host is requested to re-transmit the packet again.

#### 26.1.8.5.3 Pins Used

The list of device pins that are configured by the ROM in the case of UART boot mode are as follows.

**Note:** All the pins might not be driven at boot time.

**Table 26-35. Pins Used for UART Boot**

| Signal name | Pin Used in Device |
|-------------|--------------------|
| rx          | uart0_rxd          |
| tx          | uart0_txd          |

### 26.1.8.6 USB Boot Procedure

#### 26.1.8.6.1 Device Initialization

The ROM code supports booting from the USB interface under the following conditions:

- When the high-speed USB OTG (USBOTGHS) IP is used through USB0 interface.
- USB operates in full-speed, client mode.
- USB will operate only in device-powered mode.
- Integrated transceiver (through UTMI).
- The enumeration default timeout is 3s (USB timeout).
- ROM code uses the default value of DATAPOLARITY.

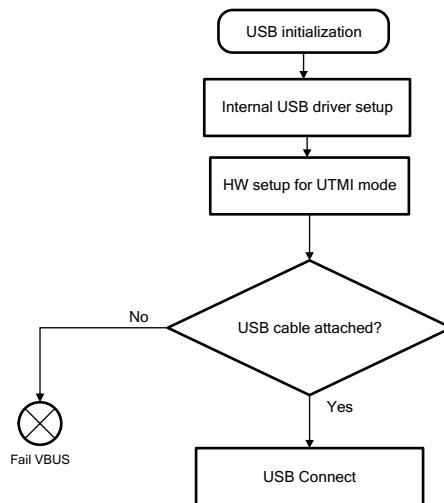
Even though using an OTG capable hardware, the ROM Code does not handle any OTG specific feature.

### 26.1.8.6.1.1 Overview

In case of boot from USB is chosen by the SYSBOOT pin configuration:

- The USBOTGHS hardware and PRCM clocks are configured for UTMI mode.
- The ROM Code continues with the USB procedure only if the USB cable is detected present (i.e. VBUS is detected at transceiver level and communicated as such through the UTPI traffic). If not, the initialization procedure is aborted.
- The ROM code implements the RNDIS class driver.
- From a user's perspective, USB boot is indistinguishable from Ethernet boot.
- The USB initialization procedure is shown in [Figure 26-24](#).

**Figure 26-24. USB Initialization Procedure**



### 26.1.8.6.1.2 Enumeration Descriptors

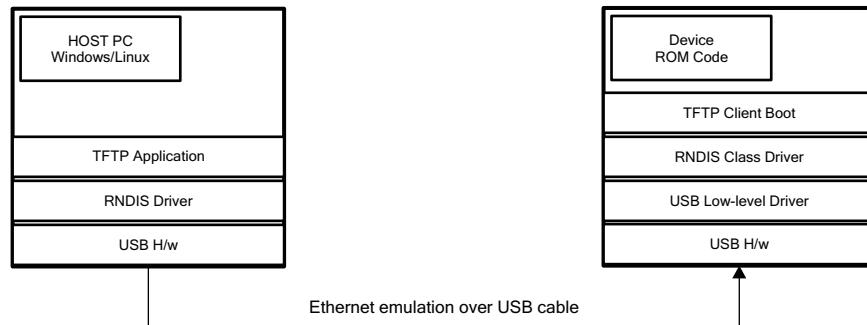
The device descriptor parameters which are used during enumeration are listed in [Table 26-36](#). The default Vendor ID and Product ID can be automatically overridden by the customer by programming the Efuses that are used to store these values.

**Table 26-36. Customized Descriptor Parameters**

| Parameter        | Size [bytes] | TI Default Values   |
|------------------|--------------|---------------------|
| Device ID code   | 2            | 0000h               |
| Device Class     | 1            | 02h                 |
| Device Sub-Class | 1            | FFh                 |
| Device Protocol  | 1            | FFh                 |
| Manufacturer     | String       | "Texas Instruments" |
| Product          | String       | "USBRNDIS"          |
| Serial number    | String       | "N/A"               |

### 26.1.8.6.2 Image Download Procedure

- The device supports USB client boot only.
- The ROM implements as RNDIS class driver, so the device enumerates as an ethernet port.
- Standard RNDIS drivers present on Linux and Windows are picked up during the enumeration. No special drivers need to be installed.
- Once the enumeration is complete, the customer can download the boot image using any standard TFTP server application.

**Figure 26-25. Image Transfer for USB Boot**


#### 26.1.8.6.3 Pins Used

The list of the device pins that are configured by the ROM in the case of USB boot mode are as follows. Please note that all the pins might not be driven at boot time.

**Table 26-37. Pins Used for USB Boot**

| Signal Name | Pin Used in Device |
|-------------|--------------------|
| USB0_DM     | USB0_DM            |
| USB0_DP     | USB0_DP            |
| USB0_ID     | USB0_ID            |
| USB0_VBUS   | USB0_VBUS          |

#### 26.1.8.7 ASIC ID structure

The ASIC ID size is 58 bytes for UART and 81 bytes for others. The fields of this structure are unused. This structure is included purely for legacy purposes.

#### 26.1.9 Image Format

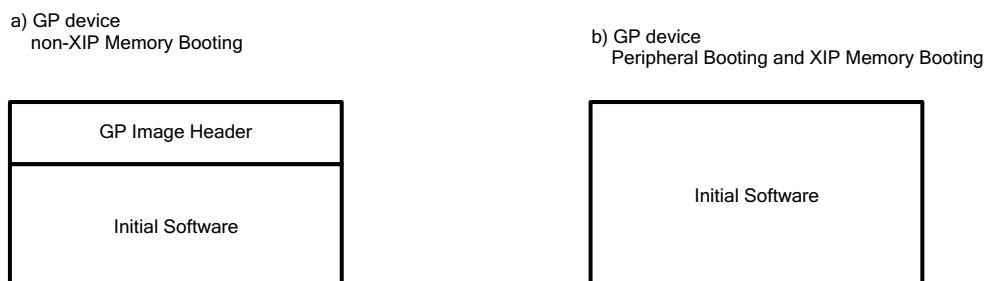
##### 26.1.9.1 Overview

All preceding sections describe how the ROM Code searches and detects a boot image from a memory or a peripheral device type. This section describes the format of the boot image.

A boot image is basically made out of two major parts:

- The software to execute
- A header containing the destination address and size of the image for non XIP memory devices

The mandatory section of a boot image contains the software which will be loaded into the memory and executed. An overview of the image formats is shown in [Figure 26-26](#):

**Figure 26-26. Image Formats on GP Devices**


### a) GP Non-XIP Memory Booting

Used for memories which require shadowing (e.g. MMC). Image must begin with a GP header which contains information on image size and destination address.

### b) GP Peripheral Booting and XIP Memory Booting

When memory device is of XIP type (e.g. NOR) the GP header is not needed and the image can contain code for direct execution. The same image format is used for peripheral booting (where the code is transferred to internal RAM).

#### 26.1.9.2 Image Format for GP Device

When the booting memory device is non-XIP (e.g. MMC) the image must contain a small header (referred to as GP header) with the size of the software to load and the destination address where to store it.

The GP header is not needed when booting from an XIP memory device (e.g. NOR) or in case of peripheral booting. In this case, the peripheral or memory booting image starts directly with executable code.

**Table 26-38. GP Device Image Format**

| Field       | Non-XIP Device Offset | XIP Device Offset | Size[bytes] | Description   |
|-------------|-----------------------|-------------------|-------------|---|
| Size        | 0000h                 | -                 | 4           | Size of the image                                   |
| Destination | 0004h                 | -                 | 4           | Address where to store the image / code entry point |
| Image       | 0008h                 | 0000h             | x           | Executable code                                     |

Note: The “Destination” address field stands for both:

- Target address for the image copy from the non-XIP storage to the target XIP location (e.g., internal RAM or SDRAM)
- Entry point for image code

In other words the user must take care to locate the code entry point to the target address for image copy.

#### 26.1.10 Code Execution

##### 26.1.10.1 Overview

One of the early steps of the Public ROM Code execution is to search for a boot image from the requested medium (configured by the SYSBOOT pins) and copy it to RAM if needed. If the device is of GP type and boot interface is non-XIP then the image is simply copied to the provided destination address (internal or external RAM) and then executed. If the boot interface is of XIP type then the image copy is not needed and execution is directly given to the XIP memory.

##### 26.1.10.2 Execution

The image is executed at the time the ROM Code performs the branch to the first executable instruction inside the Initial SW. For a GP Device in non-XIP, the execution address is the first word after the GP header. The branch is performed in public ARM supervisor mode. The R0 register points to the Booting Parameters structure which contains various information about the booting execution. [Table 26-39](#) details this structure.

**Table 26-39. Booting Parameters Structure**

| Offset | Field                                    | Size[bytes] | Description  |
|--------|--|-------------|--|
| 00h    | Reserved                                 | 4           | Reserved   |
| 04h    | Memory booting device descriptor address | 4           | Pointer to the memory device descriptor that has been used during the memory booting process. <sup>(1)</sup> |

<sup>(1)</sup> More detailed information is found in each memory's booting description.

**Table 26-39. Booting Parameters Structure (continued)**

| Offset | Field                  | Size[bytes] | Description   |
|--------|------------------------|-------------|---|
| 08h    | Current Booting Device | 1           | Code of device used for booting<br>00h – void, no device<br>01h – XIP MUX1 memory<br>02h – XIPWAIT MUX 1 (wait monitoring on)<br>03h – XIP MUX2 memory<br>04h – XIPWAIT MUX 2 (wait monitoring on)<br>05h – NAND<br>06h – NAND with I2C<br>08h – MMC/SD port 0<br>09h – MMC/SD port 1<br>15h – SPI<br>41h – UART0<br>44h – USB<br>46h – CPGMAC0 |
| 09h    | Reset Reason           | 1           | Current reset reason bit mask (bit=1-event present)<br>[0] – Power-on (cold) reset<br>[1] – Global warm software reset<br>[2] – Reserved<br>[3] – Reserved<br>[4] – WDT1 timer reset<br>[5] – Global external warm reset<br>Other bits – Reserved<br>Note: ROM code does not clear any of these bits.   |
| 0Ah    | Reserved               | 1           | Reserved  |

## 26.1.11 Wakeup

### 26.1.11.1 Overview

The device supports a comprehensive set of low-powered states to meet aggressive power budget requirements. The different low-power modes supported, from the lowest power consumption state to the highest power consumption state, include:

- RTC Only
- Deep Sleep 0
- Deep Sleep 1
- Deep Sleep 2
- Standby
- Active

Of these states, the ROM code is involved only in the first three, as the A8 registers (particularly the PC) are preserved in all the other states. So for these modes, on wake up, execution will resume from the next instruction following the WFI.

In the RTC only mode, the ROM code involvement is trivial. A wakeup from the RTC only mode, from the ROM perspective, is indistinguishable from a power on reset. In Deep Sleep 0 and Deep Sleep 1 state, the ROM code should detect whether wakeup has occurred and branch to a user-defined return address rather than perform a full boot.

### 26.1.11.2 Wakeup Booting by ROM Code

In this device, when the A8 is in OFF mode, execution begins from the reset vector on wakeup. As the reset vector lies in the ROM code, the ROM code is the first software entity that takes control of the A8 on a wakeup.

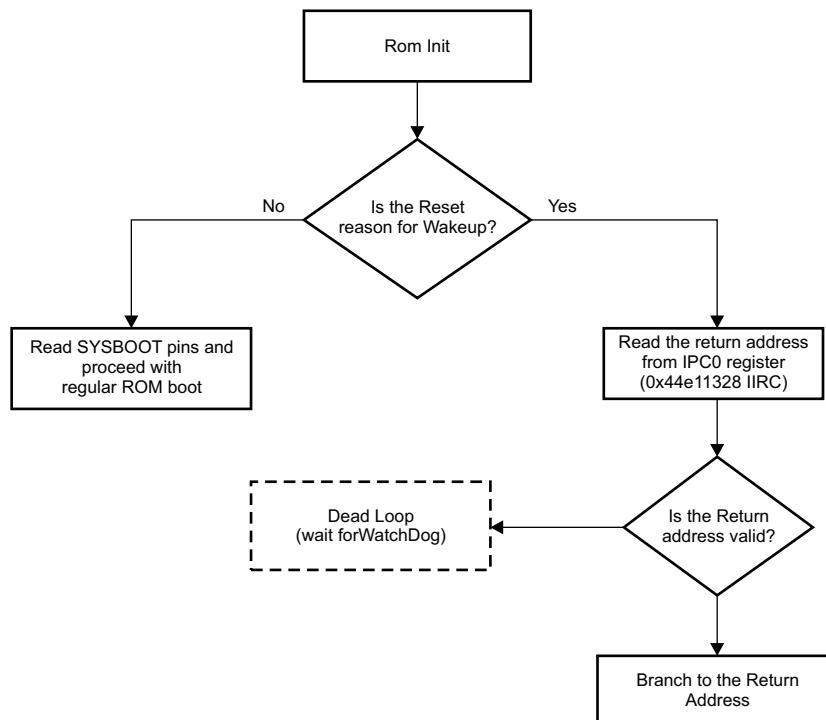
It must be noted here that in the device, in all the modes other than RTC only, the A8 internal SRAM and the L3 OCMC RAM are held in retention. This is a fundamental assumption of the ROM code. As the contents of these RAMs are not lost in the Deep Sleep modes, it is possible to return to a location in these memories.

This does away with the need of the ROM having to restore PLL and EMIF settings, which would have been needed if the ROM had to return to a DRAM address, as DRAM is held in self-refresh. The job of dialing up the PLLs and restoring EMIF and other peripheral register values is the responsibility of the user code. The recommendation is that such restoration code be placed in the OCMC RAM.

In the past, having complex PLL and EMIF restore code in the ROM has made wakeup debugging very complicated. This has also traditionally been a source of many bugs, as the wakeup procedures are complicated and difficult to test exhaustively in pre-silicon.

The current architecture does away with these problems. The flow in the ROM on wakeup is kept simple and minimal. It only involves identifying that the reset reason is a Wakeup, and then branching to a return address, rather than proceeding with a full boot. This also helps in improving the time required to wakeup a system that is in Deep Sleep.

**Figure 26-27. Wakeup Booting by ROM**



### 26.1.12 Tracing

Tracing in the Public ROM Code includes three 32-bit vectors for which each bit corresponds to a particular “way point” in the ROM Code execution sequence (see [Table 26-4](#)). Tracing vectors are initialized at the very beginning of the startup phase and updated all along the boot process.

There are two sets of tracing vectors. The first set is the current trace information (after cold or warm reset). The second set holds a copy of trace vectors collected at the first ROM Code run after cold reset. As a consequence after a warm reset it is possible to have visibility on the boot scenario that occurred during cold reset.

**Table 26-40. Tracing Vectors**

| Trace vector | Bit No. | Group                | Meaning                                |
|--------------|---------|----------------------|--|
| 1            | 0       | General              | Passed the public reset vector         |
| 1            | 1       | General              | Entered main function                  |
| 1            | 2       | General              | Running after the cold reset           |
| 1            | 3       | Boot                 | Main booting routine entered           |
| 1            | 4       | Memory Boot          | Memory booting started                 |
| 1            | 5       | Peripheral Boot      | Peripheral booting started             |
| 1            | 6       | Boot                 | Booting loop reached last device       |
| 1            | 7       | Boot                 | GP header found                        |
| 1            | 8       | Boot                 | Reserved                               |
| 1            | 9       | Boot                 | Reserved                               |
| 1            | 10      | Peripheral Boot      | Reserved                               |
| 1            | 11      | Peripheral Boot      | Reserved                               |
| 1            | 12      | Peripheral Boot      | Device initialized                     |
| 1            | 13      | Peripheral Boot      | Asic Id sent                           |
| 1            | 14      | Peripheral Boot      | Image received                         |
| 1            | 15      | Peripheral Boot      | Peripheral booting failed              |
| 1            | 16      | Peripheral Boot      | Booting Message not received (timeout) |
| 1            | 17      | Peripheral Boot      | Image size not received (timeout)      |
| 1            | 18      | Peripheral Boot      | Image not received (timeout)           |
| 1            | 19      | Reserved             | Reserved                               |
| 1            | 20      | Configuration Header | CHSETTINGS found                       |
| 1            | 21      | Configuration Header | CHSETTINGS executed                    |
| 1            | 22      | Configuration Header | CHRAM executed                         |
| 1            | 23      | Configuration Header | CHFLASH executed                       |
| 1            | 24      | Configuration Header | CHMMCSD clocks executed                |
| 1            | 25      | Configuration Header | CHMMCSD bus width executed             |
| 1            | 26      | Reserved             | Reserved                               |
| 1            | 27      | Reserved             | Reserved                               |
| 1            | 28      | Reserved             | Reserved                               |
| 1            | 29      | Reserved             | Reserved                               |
| 1            | 30      | Reserved             | Reserved                               |
| 1            | 31      | Reserved             | Reserved                               |
| 2            | 0       | Companion chip       | Reserved                               |
| 2            | 1       | Companion chip       | Reserved                               |
| 2            | 2       | Companion chip       | Reserved                               |
| 2            | 3       | Companion chip       | Reserved                               |
| 2            | 4       | USB                  | USB connect                            |
| 2            | 5       | USB                  | USB configured state                   |
| 2            | 6       | USB                  | USB VBUS valid                         |
| 2            | 7       | USB                  | USB session valid                      |
| 2            | 8       | Reserved             | Reserved                               |
| 2            | 9       | Reserved             | Reserved                               |
| 2            | 10      | Reserved             | Reserved                               |
| 2            | 11      | Reserved             | Reserved                               |
| 2            | 12      | Memory Boot          | Memory booting trial 0                 |

**Table 26-40. Tracing Vectors (continued)**

| Trace vector | Bit No. | Group                    | Meaning                                       |
|--------------|---------|--------------------------|---|
| 2            | 13      | Memory Boot              | Memory booting trial 1                        |
| 2            | 14      | Memory Boot              | Memory booting trial 2                        |
| 2            | 15      | Memory Boot              | Memory booting trial 3                        |
| 2            | 16      | Memory Boot              | Execute GP image                              |
| 2            | 17      | Peripheral Boot          | Start authentication of peripheral boot image |
| 2            | 18      | Memory & Peripheral Boot | Jumping to Initial SW                         |
| 2            | 19      | Memory & Peripheral Boot | Reserved                                      |
| 2            | 20      | Memory & Peripheral Boot | Start image authentication                    |
| 2            | 21      | Memory & Peripheral Boot | Image authentication failed                   |
| 2            | 22      | Memory & Peripheral Boot | Analyzing SpeedUp                             |
| 2            | 23      | Memory & Peripheral Boot | Speedup failed                                |
| 2            | 24      | Memory & Peripheral Boot | Reserved                                      |
| 2            | 25      | Memory & Peripheral Boot | Reserved                                      |
| 2            | 26      | Memory & Peripheral Boot | Reserved                                      |
| 2            | 27      | Memory & Peripheral Boot | Reserved                                      |
| 2            | 28      | Memory & Peripheral Boot | Authentication procedure failed               |
| 2            | 29      | Reserved                 | Reserved                                      |
| 2            | 30      | Reserved                 | Reserved                                      |
| 2            | 31      | Reserved                 | Reserved                                      |
| 3            | 0       | Memory Boot              | Memory booting device NULL                    |
| 3            | 1       | Memory Boot              | Memory booting device XIP                     |
| 3            | 2       | Memory Boot              | Memory booting device XIPWAIT                 |
| 3            | 3       | Memory Boot              | Memory booting device NAND                    |
| 3            | 4       | Memory Boot              | Reserved                                      |
| 3            | 5       | Memory Boot              | Memory booting device MMCSD0                  |
| 3            | 6       | Reserved                 | Reserved                                      |
| 3            | 7       | Memory Boot              | Memory booting device MMCSD1                  |
| 3            | 8       | Reserved                 | Reserved                                      |
| 3            | 9       | Reserved                 | Reserved                                      |
| 3            | 10      | Memory Boot              | Reserved                                      |
| 3            | 11      | Reserved                 | Reserved                                      |
| 3            | 12      | Reserved                 | Memory booting device SPI                     |
| 3            | 13      | Reserved                 | Reserved                                      |
| 3            | 14      | Reserved                 | Reserved                                      |
| 3            | 15      | Reserved                 | Reserved                                      |
| 3            | 16      | Reserved                 | Peripheral booting device UART0               |
| 3            | 17      | Reserved                 | Reserved                                      |
| 3            | 18      | Peripheral Boot          | Reserved                                      |
| 3            | 19      | Reserved                 | Reserved                                      |
| 3            | 20      | Peripheral Boot          | Peripheral booting device USB                 |
| 3            | 21      | Peripheral Boot          | Reserved                                      |
| 3            | 22      | Peripheral Boot          | Peripheral booting device GPGMAC0             |
| 3            | 23      | Reserved                 | Reserved                                      |

**Table 26-40. Tracing Vectors (continued)**

| <b>Trace vector</b> | <b>Bit No.</b> | <b>Group</b> | <b>Meaning</b>                 |
|---------------------|----------------|--------------|--------------------------------|
| 3                   | 24             | Reserved     | Peripheral booting device NULL |
| 3                   | 25             | Reserved     | Reserved                       |
| 3                   | 26             | Reserved     | Reserved                       |
| 3                   | 27             | Reserved     | Reserved                       |
| 3                   | 28             | Reserved     | Reserved                       |
| 3                   | 29             | Reserved     | Reserved                       |
| 3                   | 30             | Reserved     | Reserved                       |
| 3                   | 31             | Reserved     | Reserved                       |

## Revision History

**Table A-1** highlights the technical changes made to the SPRUH73F technical reference manual to make it an SPRUH73G revision.

**Table A-1. Document Revision History**

| Reference   | Additions/Modifications/Deletions  |
|---|--|
| <a href="#">Chapter 1<br/>Introduction</a>            | Changed DEVREV description in <a href="#">Table 1-2</a><br>Deleted <i>Device ID Register Values</i> table.   |
| <a href="#">Chapter 2<br/>Memory Map</a>              | Changed size of USB registers in <a href="#">Table 2-1, L3 Memory Map</a> .<br>Added Reserved space from 0x4740_5000 to 0x477F_FFFF in <a href="#">Table 2-1, L3 Memory Map</a> .<br>Added ADC_TSC DMA (0x54C0_0000–0x54FF_FFFF) to <a href="#">Table 2-1, L3 Memory Map</a> .<br>Updated L4_WKUP entry in <a href="#">Table 2-5, M3 Processor Memory Map</a> .  |
| <a href="#">Section 3.1<br/>ARM MPU<br/>Subsystem</a> | Updated <a href="#">Section 3.1.7, ARM Programming Model</a> .<br>Added <a href="#">Section 3.1.4.2.2, Secure Monitor Calls to Access CP15 Registers</a> .   |
| <a href="#">Chapter 6<br/>Interrupts</a>              | Added note to <a href="#">Section 6.1, Functional Description</a> , stating FIQ is not available on general-purpose devices.<br>Added note to and updated list of registers in <a href="#">Section 6.6, Interrupt Controller Registers</a> .   |
| <a href="#">Chapter 7<br/>Memory Subsystem</a>        | <b>OCMC-RAM</b><br>Updated <a href="#">Section 7.2.1.1, OCMC-RAM Features</a> .<br><b>GPMC</b><br>Updated <a href="#">Figure 7-2, GPMC Integration</a> .<br>Updated clocks in <a href="#">Table 7-7, GPMC_CONFIG1_i Configuration</a> .<br>Updated <a href="#">Table 7-5, GPMC Pin Multiplexing Options</a> .<br>Changed gpmc_a[26:16] to gpmc_[11:1] in <a href="#">Figure 7-3, GPMC to 16-Bit Address/Data-Multiplexed Memory</a> .<br>Updated description for DEVICESIZE bit (bit 13–12) in <a href="#">Table 7-65, GPMC_CONFIG1_i Field Descriptions</a> .<br>Updated description for LIMITEDADDRESS in <a href="#">Table 7-63, GPMC_CONFIG Field Descriptions</a> .<br><b>EMIF</b><br>Changed references to Peripheral Bus Burst Priority Register with Interface Configuration Register.<br>Updated maximum frequency in <a href="#">Table 7-95, EMIF Clock Signals</a> .<br>Updated <a href="#">Section 7.3.6, DDR2/3/mDDR PHY Registers</a> .<br>Added offsets 0x54, 0x58, and 0x5C to <a href="#">Section 7.3.5, EMIF4D Registers</a> . |

**Table A-1. Document Revision History (continued)**

| Reference                                   | Additions/Modifications/Deletions  |
|---|--|
| Chapter 8<br><i>PRCM</i>                    | <p>Updated description for Full idle in <a href="#">Table 8-4, Idle States for a Slave Module</a>.</p> <p>Changed selected mode to NO_SLEEP for value 0x0 in <a href="#">Table 8-9, Clock Transition Mode Settings</a>.</p> <p>Updated description of <a href="#">Section 8.1.5.1.9, Targets Modules Interface</a>.</p> <p>Changed DIV value for OPP100 to 4 in <a href="#">Table 8-20, Core PLL Typical Frequencies (MHz)</a>.</p> <p>Updated <a href="#">Section 8.1.6.3, ADPLLS</a>.</p> <p>Updated <a href="#">Section 8.1.6.4, ADPLLJ (Low Jitter DPLL)</a>.</p> <p>Changed SYS_CLK to CLK_M_OSC and PRM_DEVICE_RSTTIME1 to PRM_RSTTIME.RSTTIME1 in <a href="#">Section 8.1.7, Reset Management</a>.</p> <p>Updated <a href="#">Figure 8-18, PORz</a>, <a href="#">Figure 8-19, External System Reset</a>, <a href="#">Figure 8-20, nRESETIN_OUT Waveform as Warm Reset Source</a>, and <a href="#">Figure 8-21, nRESETIN_OUT Waveform Not as Warm Reset Source</a>.</p> <p>Added <a href="#">Section 8.1.6.6, Spread Spectrum Clocking (SSC)</a>.</p> <p>Updated <a href="#">Figure 8-12, Display PLL Structure</a>.</p> <p>Updated <a href="#">Figure 8-14, CLKOUT Signals</a>.</p> |
| Chapter 9<br><i>Control Module</i>          | <p>Updated <a href="#">Table 9-1, Pad Control Register Field Descriptions</a>.</p> <p>Updated <a href="#">Table 9-3, Pull Selection</a>.</p> <p>Added <a href="#">Section 9.2.2.3, RX Active</a>.</p> <p>Updated description for <a href="#">Section 9.2.4.1, Control and Boot Status</a>.</p> <p>Added notes to <a href="#">Table 9-7, DDR Slew Rate Control Settings</a>, and <a href="#">Table 9-8, DDR Impedance Control Settings</a>.</p> <p>Updated description to disable charger after successful detection <a href="#">Section 9.2.4.4.2.2, Operation</a>.</p> <p>Added <a href="#">Section 9.3.6, control_emif_sdram_config register</a>.</p> <p>Updated description of rgmii2_idmode and rgmii1_idmode in <a href="#">Table 9-41, gmii_sel Register Field Descriptions</a>.</p> <p>Updated <a href="#">Table 9-59, usb_vid_pid Register Field Descriptions</a>.</p> <p>Updated offset for <a href="#">Section 9.3.50, conf_&lt;module&gt;_&lt;pin&gt; Register (offset = 800h–A34h)</a>.</p> <p>Added register description to ipc_msg_reg0–7 registers.</p>   |
| Chapter 11<br><i>EDMA</i>                   | <p>Changed number of PaRAM sets to 256 from 512.</p> <p>Changed Pong Set to 66 from 65 in <a href="#">Figure 11-36, Ping-Pong Buffering for McASP Example PaRAM Configuration</a>.</p> <p>Updated reset value to 2 for NUM_EVQUE in <a href="#">Figure 11-43, EDMA3CC Configuration Register (CCCFG)</a>.</p> <p>Updated description of IDLEMODE bit in <a href="#">Table 11-28, EDMA3CC System Configuration Register (SYSCONFIG) Field Descriptions</a>.</p>   |
| Chapter 12<br><i>Touchscreen Controller</i> | <p>Updated <a href="#">Section 12.3.5, Interrupts</a>.</p> <p>Updated <a href="#">Figure 12-2, Functional Block Diagram</a>.</p> <p>Updated CTRL, ADC_MISC, and IDLECONFIG registers in <a href="#">Section 12.5.1, Touchscreen Controller Registers</a>.</p>  |
| Chapter 13<br><i>LCD Controller</i>         | <p>Changed LCD_D[15:0] to LCD_D[23:0] in <a href="#">Figure 13-1, LCD Controller</a>.</p> <p>Updated <a href="#">Section 13.2, LCD Controller Integration</a>.</p> <p>Changed LCD_PCLK to <u>LCD_AC_BIAS_EN</u> for Character Display in <a href="#">Table 13-6, LIDD I/O Name Map</a>.</p> <p>Changed (Pix9)B to (Pix8)B in <a href="#">Figure 13-14, Monochrome and Color Output</a>.</p> <p>Updated values in <a href="#">Table 13-10, Color/Grayscale Intensities and Modulation Rates</a>.</p> <p>Updated registers CTRL, SYSCONFIG, IRQSTATUS_RAW, IRQSTATUS, IRQENABLE_SET, IRQENABLE_CLEAR in <a href="#">Section 13.5, LCD Registers</a>.</p>   |
| Chapter 14<br><i>Ethernet Subsystem</i>     | <p>Added Internal delay mode for RGMII to <a href="#">Table 14-1, Unsupported CPGMAC Features</a>.</p> <p>Updated the steps in the interrupt service routine in <a href="#">Section 14.3.7.3, Interrupt Handling</a>.</p> <p>Updated description for <a href="#">Section 14.3.6.4, RGMII Transmit (TX)</a>.</p> <p>Updated <a href="#">Section 14.5.6, CPSW_PORT Registers</a>.</p>  |

**Table A-1. Document Revision History (continued)**

| Reference                                  | Additions/Modifications/Deletions   |
|--|---|
| Chapter 15<br><i>PWMSS</i>                 | Updated <a href="#">Section 15.1.1.2</a> , Unsupported Features.  |
| Chapter 16<br><i>USB</i>                   | Updated <i>Return policy</i> description and bit range for <i>Packet return queue mgr #</i> in <a href="#">Table 16-11</a> , <i>Packet Descriptor Word 2 (PD2) Bit Field Descriptions</i> .<br>Updated <a href="#">Section 16.3.9.2.3</a> , <i>Teardown Descriptor</i> .<br>Updated range of queues in <a href="#">Section 16.3.9.3.4</a> , <i>Transmit Completion (Return) Queue</i> .<br>Removed initialization code from <a href="#">Section 16.3.4</a> , <i>Clock, PLL, and PHY Initialization</i> .<br>Updated <a href="#">Section 16.5.1</a> , <i>USBSS Registers</i> .<br>Updated <a href="#">Section 16.5.2</a> , <i>USB0_CTRL Registers</i> .<br>Updated <a href="#">Section 16.5.3</a> , <i>USB1_CTRL Registers</i> . |
| Chapter 18<br><i>Multimedia Card (MMC)</i> | Updated features to include SD Physical Layer specification v2.00.<br>Updated unsupported features to include Dual Data Rate (DDR) mode. See <i>SD_CON</i> register.<br>Updated <a href="#">Figure 18-1</a> , <i>MMCHS Module SDIO Application</i> .<br>Updated functional clock frequency in <a href="#">Section 18.2.2</a> , <i>MMCHS Clock and Reset Management</i> .<br>Changed description of 3h to Reserved in <a href="#">Section 18.3.3.2</a> , <i>Idle Mode</i> .<br>Updated <i>SD_SYSCONFIG</i> and <i>SD_SYSCTL</i> registers.   |
| Chapter 19<br><i>UART</i>                  | Updated RXSTOPIT description in <a href="#">Table 19-34</a> , <i>CIR Interrupt Enable Register (IER) Field Descriptions</i> .<br>Updated <a href="#">Table 19-62</a> , <i>System Configuration Register (SYSC) Field Descriptions</i> .<br>Updated <a href="#">Table 19-71</a> , <i>XON2/ADDR2 Register Field Descriptions</i> .<br>Updated value for XOFFWORD1 in <a href="#">Table 19-72</a> , <i>XOFF1 Register Field Descriptions</i> .   |
| Chapter 20<br><i>Timers</i>                | <b>DMTimer</b><br>Changed wake-up enable feature to clarify only for Timer0, listed in <a href="#">Section 20.1.1.2</a> , <i>Features</i> .<br>Changed <a href="#">Table 20-3</a> , <i>Timer[2-7] Connectivity Attributes</i> , to clarify no wake-up capabilities.<br>Updated <i>TIOCPCFG Register</i> and <i>RTC_SYSCONFIG</i> .<br><b>Watchdog Timer</b><br>Updated steps in <a href="#">Section 20.4.3.8</a> , <i>Start/Stop Sequence for Watchdog Timers (Using the WDT_WSPR Register)</i> .<br>Added bits 4:3 to <i>WDT_WDSC</i> register.  |
| Chapter 21<br><i>I2C</i>                   | Updated FIFO in <a href="#">Section 21.1.1</a> , <i>I2C Features</i> .<br>Added note to <a href="#">Section 21.3.8</a> , <i>I2C Clock Generation and I2C Clock Synchronization</i> .<br>Updated <i>I2C_SYS</i> register.  |
| Chapter 22<br><i>McASP</i>                 | Updated <a href="#">Section 22.3.5</a> , <i>Clock and Frame Sync Generators</i> .<br>Changed reference to PWRDEMU with PWRIDLESYSCONFIG in <a href="#">Section 22.3.12.2</a> , <i>Transmit/Receive Section Initialization</i> .<br>Added <a href="#">Section 22.4.1.2</a> , <i>Power Idle SYSCONFIG Register</i> .  |
| Chapter 24<br><i>McSPI</i>                 | Changed FFNBYTE to 64 byte.<br>Removed from <i>Buffer Almost Full</i> and <i>Buffer Almost Empty</i> whether register width depends on the generic parameter 64 value.<br>Updated <a href="#">Section 24.3.2.10</a> , <i>FIFO Buffer Management</i> .<br>Updated <a href="#">Section 24.4.1.13</a> , <i>McSPI Transfer Levels Register (MC SPI_XFERLEVEL)</i> .   |
| Chapter 23<br><i>CAN</i>                   | Changed references to OSC0 with CLK_M_OSC for Dual Clock Source.  |
| Chapter 25<br><i>GPIO</i>                  | Updated <a href="#">Section 25.4.1</a> , <i>GPIO Registers</i> .  |

**Table A-1. Document Revision History (continued)**

| Reference                                 | Additions/Modifications/Deletions  |
|---|--|
| <a href="#">Chapter 26 Initialization</a> | <p>Updated <a href="#">Table 26-7, SYSBOOT Configuration Pins</a>.</p> <p>Updated <a href="#">Table 26-9, Pins Used for NOR Boot</a>.</p> <p>Updated GPMC Initialization and Reading NAND Geometry From I2C EEPROM in <a href="#">Section 26.1.7.4.2, Initialization and Detection</a>.</p> <p>Added note to <a href="#">Section 26.1.7.4.2.2, Pins Used for NAND boot</a>.</p> <p>Updated description of <a href="#">Section 26.1.7.4.2.3, SYSBOOT Pins</a>, for NAND boot.</p> <p>Updated booting restrictions in <a href="#">Section 26.1.7.5.2, System Interconnection</a>.</p> <p>Deleted SLC and MLC from NAND features.</p> <p>Added <a href="#">Section 26.1.7.5.5.1, Configuration Header</a>.</p> <p>Updated tables on pins used for EMAC boot: <a href="#">Table 26-31</a>, <a href="#">Table 26-32</a>, and <a href="#">Table 26-33</a>.</p> <p>Updated <a href="#">Table 26-37, Pins Used for USB Boot</a>.</p> <p>Updated <a href="#">Section 26.1.8.4 EMAC Boot Procedure</a>.</p> <p>Changed 10b to Reserved in <a href="#">Table 26-34, Ethernet PHY Mode Selection</a>, for EMAC boot.</p> <p>Changed even parity to no parity in device initialization for <a href="#">Section 26.1.8.5, UART Boot Procedure</a>.</p> <p>Added sixth bullet to device initialization in <a href="#">Section 26.1.8.6, USB Boot Procedure</a>.</p> <p>Updated <a href="#">Table 26-39, Booting Parameters Structure</a>.</p> <p>Updated Memory booting device SPI in <a href="#">Table 26-40, Tracing Vectors</a>.</p> |

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components which meet ISO/TS16949 requirements, mainly for automotive use. Components which have not been so designated are neither designed nor intended for automotive use; and TI will not be responsible for any failure of such components to meet such requirements.

### Products

|                              |  |
|------------------------------|--|
| Audio                        | <a href="http://www.ti.com/audio">www.ti.com/audio</a>                               |
| Amplifiers                   | <a href="http://amplifier.ti.com">amplifier.ti.com</a>                               |
| Data Converters              | <a href="http://dataconverter.ti.com">dataconverter.ti.com</a>                       |
| DLP® Products                | <a href="http://www.dlp.com">www.dlp.com</a>   |
| DSP                          | <a href="http://dsp.ti.com">dsp.ti.com</a>   |
| Clocks and Timers            | <a href="http://www.ti.com/clocks">www.ti.com/clocks</a>                             |
| Interface                    | <a href="http://interface.ti.com">interface.ti.com</a>                               |
| Logic                        | <a href="http://logic.ti.com">logic.ti.com</a>                                       |
| Power Mgmt                   | <a href="http://power.ti.com">power.ti.com</a>                                       |
| Microcontrollers             | <a href="http://microcontroller.ti.com">microcontroller.ti.com</a>                   |
| RFID                         | <a href="http://www.ti-rfid.com">www.ti-rfid.com</a>                                 |
| OMAP Applications Processors | <a href="http://www.ti.com/omap">www.ti.com/omap</a>                                 |
| Wireless Connectivity        | <a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a> |

### Applications

|                               |  |
|-------------------------------|--|
| Automotive and Transportation | <a href="http://www.ti.com/automotive">www.ti.com/automotive</a>                         |
| Communications and Telecom    | <a href="http://www.ti.com/communications">www.ti.com/communications</a>                 |
| Computers and Peripherals     | <a href="http://www.ti.com/computers">www.ti.com/computers</a>                           |
| Consumer Electronics          | <a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>                   |
| Energy and Lighting           | <a href="http://www.ti.com/energy">www.ti.com/energy</a>                                 |
| Industrial                    | <a href="http://www.ti.com/industrial">www.ti.com/industrial</a>                         |
| Medical                       | <a href="http://www.ti.com/medical">www.ti.com/medical</a>                               |
| Security                      | <a href="http://www.ti.com/security">www.ti.com/security</a>                             |
| Space, Avionics and Defense   | <a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a> |
| Video and Imaging             | <a href="http://www.ti.com/video">www.ti.com/video</a>                                   |
| <b>TI E2E Community</b>       | <a href="http://e2e.ti.com">e2e.ti.com</a>   |